Proyecto EduTech Innovators Spa.

Equipo: Tiare Acuña

Ricardo Rodríguez

Maximiliano Rubilar

Fecha: 08 de abril de 2025.

Contenido

[Introducción 3](#_Toc195013681)

[Análisis de requerimientos 4](#_Toc195013682)

[Requisitos funcionales 4](#_Toc195013683)

[Requisitos no funcionales 5](#_Toc195013684)

[Perfiles de usuarios 6](#_Toc195013685)

[Cliente 6](#_Toc195013686)

[Administrador 6](#_Toc195013687)

[Vendedor 7](#_Toc195013688)

[Análisis de sistema actual 9](#_Toc195013689)

[Sistema monolítico 9](#_Toc195013690)

[Diseño nueva arquitectura 11](#_Toc195013691)

[Estrategia de microservicios 11](#_Toc195013692)

[Plan de migración 12](#_Toc195013693)

[Riesgos 14](#_Toc195013694)

[Impacto 16](#_Toc195013695)

[Modelo relacional 17](#_Toc195013696)

[Diagrama de casos de uso 19](#_Toc195013697)

# Introducción

En la actualidad, la educación en línea se ha convertido en un pilar fundamental para el aprendizaje a nivel global. La posibilidad de acceder a cursos y materiales educativos desde cualquier lugar y en cualquier momento ha democratizado el conocimiento, permitiendo que miles de personas amplíen sus habilidades y conocimientos sin las barreras tradicionales del aprendizaje presencial. En este contexto, EduTech Innovators SPA ha desempeñado un papel crucial en la transformación digital del sector educativo en Chile, ofreciendo plataformas de aprendizaje innovadoras y accesibles para una amplia comunidad de estudiantes y profesionales.

Desde su fundación, EduTech Innovators SPA ha experimentado un crecimiento exponencial, expandiendo sus operaciones desde su sede en Providencia, Santiago, hacia nuevas oficinas en Valparaíso y La Serena. Este rápido crecimiento es un testimonio del impacto positivo que sus soluciones educativas han tenido en el mercado. Sin embargo, con la expansión y el aumento del número de usuarios, también han surgido desafíos técnicos que amenazan con comprometer la calidad del servicio ofrecido.

El sistema actual de EduTech Innovators SPA está basado en una arquitectura monolítica, la cual ha comenzado a mostrar signos de deficiencia. Las fallas en el rendimiento, la dificultad para escalar sus servicios y la reducida disponibilidad del sistema han impactado negativamente en la experiencia de los usuarios. Problemas como tiempos de respuesta elevados, caídas inesperadas del sistema y dificultades en la administración de cursos y usuarios han generado una necesidad urgente de modernización tecnológica.

Ante este panorama, la empresa ha decidido embarcarse en un proceso de transformación digital que le permita superar las limitaciones del sistema actual y asegurar su crecimiento futuro. La solución propuesta es la migración hacia una arquitectura basada en microservicios, utilizando una base de datos MySQL para mejorar el rendimiento y la disponibilidad del sistema. Esta nueva estructura permitirá a EduTech Innovators SPA ofrecer un servicio más eficiente, escalable y confiable, garantizando una experiencia de usuario de alta calidad.

Este documento de Especificación de Requisitos del Sistema (ERS) tiene como propósito definir de manera clara y detallada los requisitos funcionales y no funcionales que deberá cumplir la nueva plataforma. Se abordarán los desafíos actuales, los objetivos del proyecto, el alcance de la solución, las restricciones y las consideraciones clave para el éxito de la implementación. A través de esta modernización tecnológica, EduTech Innovators SPA reafirma su compromiso con la educación de calidad y su visión de ser un referente en la innovación educativa en Chile y Latinoamérica.

# Análisis de requerimientos

## Requisitos funcionales

* **RF01** Autenticación de usuarios
* **RF02** Validación de credenciales
* **RF03** Recuperación de contraseña
* **RF04** Creación de cuentas de usuario
* **RF05** Recordar sesión en dispositivos de confianza
* **RF06** Bloqueo por intentos fallidos de login
* **RF07** Gestión de accesos y restablecimientos
* **RF08** Registro de intentos de inicio de sesión
* **RF09** Notificaciones de seguridad
* **RF10** Integración con sistemas de pago
* **RF11** Interfaz multilingüe
* **RF12** Registro de usuarios
* **RF13** Gestión de perfiles
* **RF14** Creación de cursos
* **RF15** Edición y eliminación de cursos
* **RF16** Asignación de instructores
* **RF17** Subida de material
* **RF18** Descarga de material
* **RF19** Organización por módulos
* **RF20** Evaluaciones automáticas
* **RF21** Evaluaciones manuales
* **RF22** Feedback de evaluaciones
* **RF23** Calificación y registro de notas
* **RF24** Foros de discusión
* **RF25** Mensajería interna
* **RF26** Clases en vivo
* **RF27** Chat en tiempo real
* **RF28** Seguimiento del proceso
* **RF29** Certificación automática
* **RF30** Historial de cursos completados
* **RF31** Suscripción a cursos
* **RF32** Accesibilidad web
* **RF33** Exportación de reportes
* **RF34** Creación de encuestas y cuestionarios
* **RF35** Respuesta a encuestas
* **RF36** Análisis de encuestas
* **RF37** Gamificación
* **RF38** Ranking de usuarios
* **RF39** Biblioteca de recursos
* **RF40** Configuración de privacidad
* **RF41** Control de versiones de documentos
* **RF42** Notificaciones automáticas
* **RF43** Configuración de frecuencia de notificaciones
* **RF44** Búsqueda avanzada
* **RF45** Integración con redes sociales
* **RF46** Recuperación de sesiones en vivo
* **RF47** Evaluación de instructores
* **RF48** Administración de cupos
* **RF49** Integración con calendarios
* **RF50** Modo oscuro
* **RF51** Mantenimiento de la plataforma

## Requisitos no funcionales

* **RNF01** Diseño
* **RNF02** Seguridad
* **RNF03** Back up y recuperación
* **RNF04** Accesibilidad
* **RNF05** Adaptabilidad
* **RNF06** Rendimiento
* **RNF07** Usabilidad
* **RNF08** Control de acceso
* **RNF09** Disponibilidad
* **RNF10** Cumplimiento estándares de calidad
* **RNF11** Escalabilidad

## Perfiles de usuarios

### Cliente

* Nombre: Juan Pérez
* Edad: 32
* Ocupación: Maestro de historia en un colegio de Santiago.
* Objetivo: Querer ampliar sus conocimientos y habilidades pedagógicas mediante cursos en línea y así poder integrar herramientas tecnológicas en sus clases.
* Motivación: Poder mejorar la experiencia de aprendizaje de los estudiantes manteniendo actualizado las nuevas tendencias educativas y tener un ascenso profesional.

**Retos:**

* Tiene tiempo limitado por su horario laboral
* Necesidad de acceder cursos económicos y de fácil uso
* Frustración con la lentitud y caídas de sistemas actual en la plataforma

**Entrevista:**

El entrevistado comentó que conoció la página a través de un colega que la usaba para poder capacitarse en habilidades tecnológicas, ya que, su colega le había comentado que los cursos eran más accesibles que en otras páginas web y a la vez eran de mejor relevancia para su labor como profesores.

También, a Juan le gusto la variedad de contenido que ofrece la plataforma pero que el mayor problema era que justo a veces cuando descargaba el material para los estudios esta se caía impidiendo que pueda acceder a este lo cual lo termina por frustrar. Por lo tanto, Juan planteó que deberían mejorar la velocidad del sistema y evitar las caídas en el servidor y que se pudiera agregar recomendaciones según la materia y metas académicas en la cual el docente necesita basarse y así poder mejorar la experiencia en la aplicación.

### Administrador

* Nombre: María Gonzales
* Edad: 40 años
* Posición: Administradora de operaciones en la oficina de Providencia, Santiago.
* Objetivo: Garantizar el uso correcto del funcionamiento del sistema educativo de la empresa, mejorando a la vez la infraestructura tecnológica para soportar el crecimiento de la empresa en el mercado nacional.
* Motivación: Evitar las interrupciones que tiene el sistema actual, satisfacer las necesidades del cliente y mantener la reputación de la empresa en el mercado nacional.

**Retos:**

* Resolver problemas técnicos recurrentes en el sistema monolítico.
* Planificar una expansión en nuevas oficinas y gestionar un buen equipo en diferentes zonas geográficas.
* Mantener la confianza de los clientes mientras se trabaja en la mejora del sistema.

**Entrevista:**

La entrevistada mencionó que ha sido un desafío bien interesante, porque la empresa al tener un crecimiento tan exponencial se ha tenido que ajustar las operaciones en el sistema para poder satisfacer la demanda que se exige. Ya que, como se menciona anteriormente, el rendimiento y la disponibilidad ha tenido caídas que afectan directamente las operaciones digitales afectando a la vez a los clientes. Además, se puede mencionar que la escalabilidad de tipo monolito es un obstáculo para la expansión que requiere el sistema software.

María también mencionó que las interrupciones constantes obligan a dedicar mayor soporte técnico, dejando menos tiempo y campo para desarrollar una estrategia de crecimiento.

Al final, la entrevistada menciona que la mejora que se deberá implementar es invertir en una transición más moderna ya que la monolítica es muy anticuada sobre todo para la velocidad de uso y constantes actualizaciones que requiere la aplicación software.

### Vendedor

* Nombre: Carlos Rojas
* Edad: 29 años
* Posición: Ejecutivo en ventas en la oficina de Valparaíso.
* Objetivo: Promover la plataforma educativa de la empresa, captando así mayor clientela para el negocio del software de la empresa “EDUTECH INNOVATORS SPA”.
* Motivaciones: Ayudar a las instituciones educativas a mejorar sus metodologías de enseñanza con herramientas modernas contribuyendo a su vez el crecimiento de EDUTECH INNOVATORS SPA.

**Retos:**

* Abordar sobre las precauciones de los clientes sobre el rendimiento y la estabilidad del sistema software.
* Diferenciarse en un mercado competitivo en lo relacionado en plataformas educativas.
* Adaptarse a las demandas específicas de las diferentes instituciones de la empresa y sus ubicaciones.

**Entrevista:**

El entrevistado Carlos mencionó que para él era bastante gratificante trabajar en la empresa ya que el proyecto de la empresa tiene un impacto bastante positivo en la educación. Sin embargo, le preocupa los problemas técnicos de la empresa los cuales le ha costado mucho el poder concretar negociaciones con clientes importantes. Estos problemas técnicos se resumen en la falta de rendimiento en el programa software debido al sistema de monolito el cual dificulta la adaptación de nuevas necesidades para la empresa como una mejor optimización ya que al ser una empresa en crecimiento requiere de una constante actualización. Por lo que, Carlos menciona que podría ser útil el mostrar un sistema más robusto y seguro a la hora de desarrollar actividades sin perder su estabilidad. Y, que se debería trabajar en una comunicación más proactiva con los clientes para así poder identificar los errores que tiene el sistema software para poder lograr la satisfacción del cliente y así generar mayor confianza.

**En resumen:**

* Se deberá cambiar el sistema de Monolitos a Microservicios para así poder optimizar de manera correcta el software.
* Tener mejor comunicación con el cliente para detectar mejor los errores que pueda tener el programa.
* Expandir el software para que este se pueda perfeccionar para su uso.
* Evitar las caídas del servidor que tiene el sistema software para mejorar la calidad de la experiencia.

# Análisis de sistema actual

## Sistema monolítico

Actualmente, el sistema que se utiliza en la oficina de Providencia es de tipo monolítico, lo cual significa que todo el software está construido como una sola unidad donde las funcionalidades están estrechamente integradas. Aunque hasta ahora ha funcionado bien en una sola sede, la organización tiene como objetivo expandirse hacia otras ciudades como Valparaíso y La Serena. Esta expansión se vuelve especialmente compleja considerando que se trata de una plataforma educativa en línea que busca llegar a nivel nacional. Sin embargo, el hecho de que el sistema sea monolítico presenta varias limitaciones importantes.

Una de las principales dificultades es la falta de flexibilidad en el desarrollo. Debido a la naturaleza de los sistemas monolíticos, cualquier cambio en una funcionalidad puede requerir la modificación del sistema completo. Esto no solo ralentiza el desarrollo, sino que también complica la implementación de nuevas características. Además, resulta casi imposible trabajar de forma independiente en distintas partes del sistema sin interferir con otras áreas, lo que frena la colaboración eficiente entre los equipos de desarrollo.

Por otro lado, el mantenimiento del sistema también se vuelve cada vez más problemático. A medida que se agrega más código con el tiempo, este se vuelve más complejo y difícil de manejar, al punto de convertirse en lo que se conoce como un "monolito incontrolable". Esto hace que la depuración de errores y el mantenimiento general del sistema sean tareas lentas y complicadas. Además, las múltiples dependencias entre los componentes del sistema son difíciles de rastrear, lo que agrega una capa adicional de complejidad.

Otro gran inconveniente es la dependencia de tecnologías antiguas. Muchos sistemas monolíticos están construidos sobre herramientas y lenguajes que hoy en día se consideran obsoletos. Esto impide que se aprovechen soluciones más modernas, eficientes y seguras. Si bien migrar a una arquitectura más actual sería lo ideal, este proceso implica un costo considerable en términos de tiempo y recursos.

La escalabilidad también se ve afectada. En un sistema monolítico, escalar significa replicar toda la aplicación, incluso si solo se necesita más capacidad en una parte específica del sistema. Esto no es eficiente y se convierte en un problema cuando se intenta crecer geográficamente, ya que la apertura de nuevas oficinas añade cargas adicionales que el sistema simplemente no puede manejar bien.

Desde la perspectiva del usuario final, estas limitaciones técnicas tienen un impacto directo. Problemas de rendimiento, lentitud y errores frecuentes afectan negativamente la experiencia del usuario. Esto genera frustración, pérdida de clientes y daña la reputación de la empresa.

Además, los sistemas monolíticos presentan lo que se conoce como “puntos únicos de fallo”. Si una parte del sistema, como el módulo de pagos, deja de funcionar, todo el sistema se cae, lo que no solo paraliza la operación, sino que también dificulta la innovación.

La integración de nuevos servicios también se vuelve un desafío. A medida que se buscan incorporar nuevas funcionalidades o conectarse con servicios externos, el sistema monolítico no permite hacerlo de manera flexible sin afectar otras partes del sistema. Esto genera mayores costos operativos al tener que reorganizar el sistema cada vez que se quiere incorporar algo nuevo.

También es importante mencionar el alto costo de infraestructura. Ejecutar un sistema monolítico, ya sea en servidores físicos o en la nube, puede ser mucho más costoso, ya que todos los módulos, incluso los menos críticos, consumen recursos constantemente. Una arquitectura modular permitiría optimizar mucho mejor estos recursos.

En cuanto a la detección de errores, los diagnósticos suelen ser lentos. Encontrar la causa de una falla implica revisar grandes bloques de código, lo que ralentiza la solución de problemas y genera tiempos de inactividad, afectando la continuidad del servicio y elevando los costos de soporte técnico.

Por último, este tipo de arquitectura limita la renovación y el crecimiento de la empresa. Al ser tan rígido, el sistema impide implementar nuevas ideas o estrategias de manera ágil, lo que frena la evolución del producto y reduce su competitividad en el mercado.

# Diseño nueva arquitectura

## Estrategia de microservicios

**Problemas de rendimiento:**

* División de los servicios a más pequeños: Al poder descomponer los monolitos en microservicios, donde cada uno deberá de encargarse de de una funcionalidad que sea específica. Ya que esto permitirá que el servicio pueda escalar de manera independiente según las necesidades, facilitando la velocidad de escalabilidad que necesita la empresa.
* Balancear la carga: Al implementar los microservicios pueden distribuirse entre múltiples servidores y utilizar herramientas como kubernetes para así poder balancear la carga para gestionar los contenedores de forma eficiente.

**Tiempo de despliegue:**

* Despliegue independiente: Cada servicio que se puede convertir en microservicio puede desarrollarse, probarse y desplegarse de una manera autónoma. Logrando reducir la inactividad y los tiempos muertos.
* Automatización: Implementar pipelines de integración (procesos automatizados diseñados para gestionar cambios en la base de datos de forma eficiente, segura y organizada) y despliegue continuo (CI/CD) para poder agilizar el proceso de entrega de actualizaciones.

**Dependencia de los datos acoplados:**

* Base de datos distribuidas: Separar la base de datos en datos de tipo monolítica ayuda a facilitar la sincronización y permite manejar los datos de manera específica para cada oficina. Como la oficina de Valparaíso o de la Serena.
* Apis y comunicaciones desacopladas: El uso de APIs REST o otros tipos de mensajería como (RabbitMQ o Kafka) para poder comunicar los microservicios necesarios y así poder evitar dependencias directas.

**Restricciones tecnológicas:**

* Tecnologías específicas para cada módulo: Con microservicios, los puede usar las herramientas y lenguajes más adecuado para cada servicio. Esto permite flexibilidad tecnológica y aprovecha las fortalezas de otras tecnologías.
* Equipo capacitado: Formar o contratar nuevo personal con experiencia en desarrollo y en mantenimiento de microservicios para poder garantizar una transición de datos exitosa.

**Consideraciones adicionales:**

* Planificación estratégica: Porque para poder emigrar de un tipo de sistema como el monolítica a uno de microservicios se necesita una planificación cuidadosa. Por lo que aplicaría una estrategia gradual incluyendo los módulos más importantes y críticos para convertirlos en microservicios primero.
* Equipo capacitado: Esto es importante ya que para poder lograr una transacción de información de manera exitosa se necesita personal con conocimiento en mantenimiento de microservicios.

## Plan de migración

1. **Evaluación inicial y planificación:**

**Auditoría del sistema actual**

* Se deberá identificar los módulos críticos que presenta el sistema de monolitos (Gestión de usuarios, pagos, cursos, etc).
* Evaluar las dependencias que poseen cada módulo.
* Analizar el rendimiento de la infraestructura del sistema y sus limitaciones actuales.

**Definición de objetivos**

* Mejorar la escalabilidad, flexibilidad y velocidad de respuesta.
* Garantizar mínima interrupción a los usuarios durante la transición a microservicios.

**Diseño de la arquitectura**

* Crear un diagrama que pueda reflejar los microservicios necesarios para sus interacciones mediante el uso de APis o colas de mensajes.
* Establecer estrategias de programación entre los servicios (RESTful APis, GRPC, eventos, etc.)

1. **Infraestructura y reparación:**

**Implementación de contenedores**

* Usar dockers para poder empaquetar los servicios y Kubernetes para la organización del equipo de programación

**Configurar entornos**

* Crear los entornos separados para el desarrollo, pruebas y producción
* Garantizar que los entornos soporten la integración o despliegue continuos (CI/CD)

**Base de datos**  

* Decidir entre bases de datos centralizada o distribuida. E incluso cada servicio puede tener su propia base de datos de ser necesario.

1. **Estrategia de migración gradual:**

**División de monolito**

* Seleccionar un módulo menos crítico como prueba de piloto para convertirlo en microservicios
* Implementar y probar el servicio antes de pasar a otros módulos del sistema

**Abordar servicios independientes**

* Identificar componentes que puedan ser separados sin afectar la operación diaria (por ejemplo, autentificación de usuarios)

**Creación de apis**

* Diseñar APIs que puedan permitir a los nuevos servicios comunicarse con otros componentes que aún permanecen en el monolito lo que facilita el cambio el sistema de monolito a microservicio.

1. **Monitoreos y pruebas**

**Implementación de herramientas de monitoreo:**

* Usar herramientas como Prometheus y Grafana para poder rastrear métricas del rendimiento de los microservicios y el sistema completo.

**Pruebas exhaustivas:**

* Realizar pruebas de carga, estrés y regresión para asegurar que los nuevos microservicios funcionen correctamente
* Verificar que los microservicios no impacten negativamente en los usuarios

1. **Despliegue progresivo**

**Implementación gradual:**

* Migrar un módulo a la vez para minimizar riesgos.
* Usar técnicas como “strangler fig pattern” (modernizar una aplicación gradual), donde los microservicios reemplazan gradualmente las funciones del monolito.

**Soporte paralelo:**

* Mantener el sistema monolito y los microservicios activos en paralelo durante la transición para asegurar su continuidad.

1. **Optimización y escalabilidad**

**Análisis post despliegue**

* Identificar cuellos de botella en los microservicios desplegados.
* Ajustar los recursos e infraestructura para poder optimizar costos y rendimiento.

**Automatización**

* Implementar procesos de CI/CD para poder facilitar actualizaciones futuras

**Escalado independiente**

* Escalar servicios individuales según la demanda, lo que no era posible en el monolito.

1. **Educación y soporte al equipo**

**Capacitación del equipo**

* Asegurar que los desarrolladores y administración dominen las nuevas tecnologías (contenedores, APIs, etc.)

**Documentación**

* Crear documentación clara sobre la arquitectura, flujos y procedimientos para futuros desarrollos y mantenimiento.
  1. **Cierre de transacciones**

**Desactivación de monolitos**

* Una vez que todos los módulos hayan sido migrados, el sistema monolítico ya puede ser eliminado para poder avanzar con el sistema de microservicios.

**Revisión final**

* Evaluar el impacto general de la migración, documentar aprendizajes claves y planificar mejoras continuas en el nuevo sistema software.

## Riesgos

1. **Complejidad al realizar el proyecto:**

**Impacto:** La arquitectura de microservicios es el doble de compleja de de realizar que el de monolitos ya que esta requiere gestión de cada servicio de forma independiente lo que requiere una planificación más exhaustiva.

**Mitigación:** Implementar un enfoque que gradual a la hora de realizar el cambio del sistema de monolito al de microservicios que pueda facilitar la comprensión del sistema en el equipo de trabajo.

1. **Interrupción del servicio**

**Impacto:** Durante la transición del sistema, pueden surgir interrupciones o inestabilidad en la plataforma que afecten en la experiencia del cliente.

**Mitigación:** Mantener a ambos sistemas (monolito, microservicios) en paralelo para evitar las interrupciones del uso de la aplicación en los usuarios y realizar pruebas de uso antes de desconectar el sistema de monolito.

1. **Dificultades en la comunicación entre los servicios**

**Impacto:** Los microservicios necesitan poder comunicarse entre sí, con el uso de APIs o colas de mensajes. Lo que podría introducir retrasos o fallos si no se diseñan correctamente.

**Mitigación:** Diseñar protocolos de comunicación robustos para establecer estrategias de monitoreo que ayuden a identificar y resolver problemas de manera rápida.

1. **Sobrecarga operacional**

**Impacto:** La gestión de múltiples servicios pequeños podría significar una mayor carga para el trabajo en equipo de operaciones, en especial para los de la etapa inicial del proyecto.

**Mitigación:** Usar herramientas de orquestación como Kubernetes para automatizar tareas en la operación para facilitar el avance del proyecto, como también el capacitar a los trabajadores en su uso para evitar retrasos en el proyecto

1. **Curva de aprendizaje para el equipo de técnicos**

**Impacto:** El adoptar nuevas tecnologías y herramientas de la infraestructura y herramientas para soportar microservicios puede ser más costoso de aprender al principio.

**Mitigación:** Planificar una capacitación intensiva con talleres para que el equipo pueda entender los funcionamientos de las nuevas herramientas tecnológicas.

1. **Costos iniciales elevados**

**Impacto:** La configuración de infraestructura y herramientas para soportar los microservicios puede ser costoso al principio.

**Mitigación:** Planificar un presupuesto detallado y mostrar los beneficiados e inversionista que es una inversión a largo plazo para el desarrollo del proyecto software.

1. **Seguridad y gobernanza**

**Impacto:** Cada microservicio tiene sus vulneraciones, lo que puede ocasionar el aumento de ciberataques o robo de información de los clientes.

**Mitigación:** Implementación de estrategias de gestión de datos, como bases de datos específicas de cada servicio.

## Impacto

Unos de los mayores impactos que tendrá el proyecto es la mejora del rendimiento y disponibilidad gracias a la reducción de los tiempos de carga y fluidez de la aplicación, la mayor escalabilidad gracias a las nuevas oficinas, servicios.

Facilidad de mantenimiento gracias a la actualización y correcciones de errores de manera más rápida. En el caso de los clientes generará una mayor satisfacción por las mejoras siendo a su vez más llamativo para los clientes.

Al mejorar el sistema software de la empresa esto creará una mejor estrategia y posición competitiva sobre otra empresa de uso académico.

## Modelo relacional

Diagrama

El contenido generado por IA puede ser incorrecto.

**1**. Gestion\_cuentas

Administra cuentas de usuario.

* Campos importantes:
  + Nombre\_usuario, Rut: claves primarias.
  + Config\_seguridad: configuración relacionada con la seguridad.
  + Registros, Respaldo: fechas de registro y respaldo.

2. Crear\_cuenta

Registra la creación de una cuenta.

* Campos clave:
  + Nombre\_usuario, Correo\_usuario, Rut: información del nuevo usuario.
  + Borrar\_usuario: indica quién creó o puede eliminar la cuenta.
* Relación con Gestion\_cuentas y Borrar\_cuenta.

3. Borrar\_cuenta

Registra eliminación de cuentas.

* Campos clave:
  + Correo\_usuario, Rut, Nombre\_usuario: datos del usuario eliminado.
  + Borrar\_usuario: quién realizó la eliminación.

4. Otorgar\_permisos

Permite asignar permisos a usuarios.

* Campos:
  + Correo\_usuario, Rut: usuario que recibe permisos.
  + Confirmacion\_acceso: tipo de permiso otorgado.
  + Borrar\_usuario: quién asignó el permiso.

5. Admin

Define parámetros administrativos.

* Claves compuestas: Config\_seguridad, Registros, Respaldo.
* Permisos: tipo de permisos administrativos otorgados.

6. Gestionar\_sistemas

Control del sistema por parte del administrador.

* Relacionado con la tabla Admin por Config\_seguridad, Registros, Respaldo.
* Campos clave:
  + Datos\_sistema: información técnica o configuracional del sistema.
  + Respaldo\_2: fecha relacionada al respaldo del sistema.

7. Respaldo

Contiene la copia de datos del sistema o usuarios.

* PK: Copia\_de\_datos.

8. Restaurar\_copia

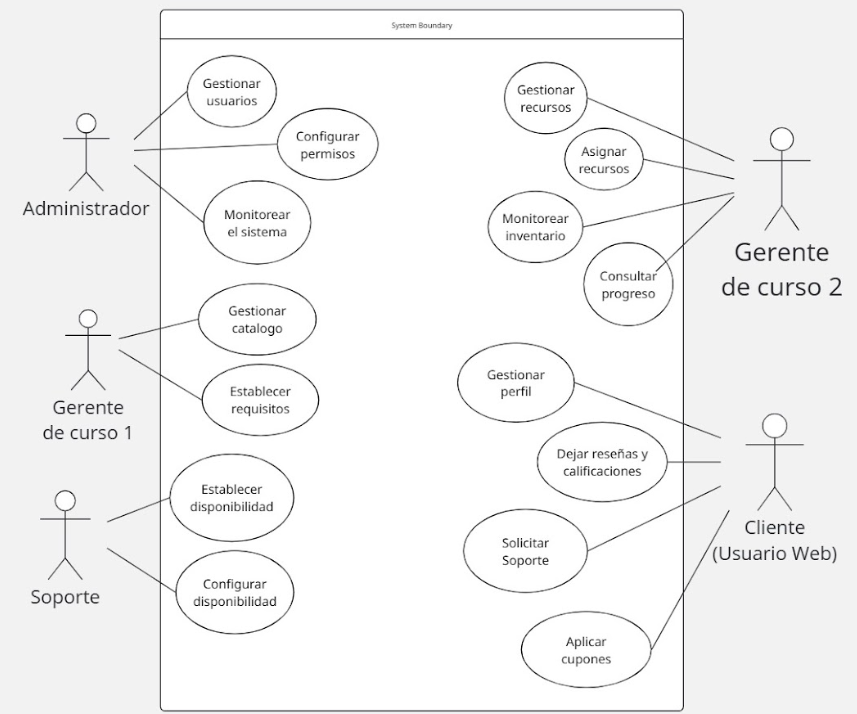
Permite restaurar información desde un respaldo.

* Campos:
  + Datos\_sistema, Respaldo\_2: referencia al respaldo.
  + PK: Restaurar\_copia.

Relaciones importantes

* Gestion\_cuentas está relacionada con Crear\_cuenta, Borrar\_cuenta y Admin.
* Admin se relaciona con Gestionar\_sistemas.
* Gestionar\_sistemas y Respaldo se conectan a través de la restauración (Restaurar\_copia).
* Controlar acciones de administradores sobre el sistema.

## Diagrama de casos de uso



Este diagrama representa un sistema educativo o de gestión de cursos, donde distintos roles tienen permisos diferenciados para administrar usuarios, recursos, cursos, soporte y funciones de cliente final.

**Actores:**

1. **Administrador**
   * Gestiona usuarios
   * Configura permisos
   * Monitorea el sistema
2. **Gerente de curso 1**
   * Gestiona el catálogo de cursos
   * Establece requisitos de los cursos
3. **Gerente de curso 2**
   * Gestiona y asigna recursos
   * Monitorea inventario
   * Consulta progreso de usuarios
4. **Soporte**
   * Establece y configura disponibilidad para soporte
5. **Cliente (Usuario Web)**
   * Gestiona su perfil
   * Deja reseñas y calificaciones
   * Solicita soporte
   * Aplica cupones