|  |
| --- |
| Especificación de requisitos de software |
| *Proyecto: EduTech Innovators SPA* |
|  |
|  |
|  |

| Especificación de Requisitos según estándar de IEEE 830 para un Sistema Biblioteca. |
| --- |

Estudiante : Rodrigo Toro

Profesor : Viviana Poblete

**Contenido**

[Ficha del documento 4](#_heading=h.n9tgj9h7i40h)

[1. Introducción 5](#_heading=h.bt123k85uko9)

[1.1. Propósito 5](#_heading=h.o93qmqwwlhuu)

[1.2. Ámbito del Sistema 5](#_heading=h.mhh677354c4g)

[1.3. Definiciones, Acrónimos y Abreviaturas 5](#_heading=h.f92l01c3yxq7)

[1.4. Referencias 5](#_heading=h.5fb1wsdiwnzu)

[1.5. Visión General del Documento 5](#_heading=h.rc4us6jtdufj)

[2. Descripción General 6](#_heading=h.53lr08jtb88k)

[2.1. Perspectiva del Producto 6](#_heading=h.96cmu53ubf41)

[2.2. Funciones del Producto 6](#_heading=h.ttnhrq4m9h9g)

[2.3. Características de los Usuarios 6](#_heading=h.c2fbfwhbye8h)

[2.4. Restricciones 6](#_heading=h.gsamsiu51lr8)

[2.5. Suposiciones y Dependencias 7](#_heading=h.sn7s4ihrfria)

[2.6. Requisitos Futuros 7](#_heading=h.bh84rn5mf72t)

[3. Requisitos Específicos 8](#_heading=h.wii28yya8uwh)

[3.1 Requisitos comunes de los interfaces 9](#_heading=h.w4v2ayl108t7)

[3.1.1 Interfaces de usuario 9](#_heading=h.aeqt2qrvvd6q)

[3.1.2 Interfaces de hardware 9](#_heading=h.f23l1fup3xds)

[3.1.3 Interfaces de software 9](#_heading=h.dy95kzmky409)

[3.1.4 Interfaces de comunicación 9](#_heading=h.4cwv7xpj7dvd)

[3.2 Requisitos funcionales 9](#_heading=h.shqo9d9pnwo9)

[3.3 Requisitos no funcionales 10](#_heading=h.yi4zb22wyhdb)

[3.3.1 Requisitos de rendimiento 10](#_heading=h.507dr8mnsex5)

[3.3.2 Seguridad 10](#_heading=h.g1xa7xa5zwwt)

[3.3.3 Fiabilidad 10](#_heading=h.mnqh533onbc8)

[3.3.4 Disponibilidad 11](#_heading=h.u0rtucu1dilj)

[3.3.5 Mantenibilidad 11](#_heading=h.q2bpp7dwjgwj)

[3.3.6 Portabilidad 11](#_heading=h.imxss1i6t98e)

[3.4 Otros Requisitos 11](#_heading=h.7i34tmwijp9x)

**Ficha del documento**

| **Fecha** | **Revisión** | **Autor** | **Modificación** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Documento validado por las partes en fecha:

| Por el cliente |  | Por la empresa suministradora |
| --- | --- | --- |
| [Firma] |  | [Firma] |
| Sr./Sra. |  | Sr./Sra. |

**1. Introducción**

Esta Especificación de Requisitos de Software (ERS) define de forma detallada las funcionalidades, restricciones, interfaces y características técnicas del nuevo sistema para EduTech Innovators SPA, en el marco del proyecto de Transformación Digital y Migración a Microservicios.

El documento ha sido elaborado , con el propósito de servir como base para el diseño, implementación, validación y mantenimiento del sistema, y como referencia compartida entre los interesados del proyecto (stakeholders), el equipo de desarrollo y la dirección técnica de EduTech.

El sistema a desarrollar reemplazará una arquitectura monolítica por una estructura modular distribuida basada en microservicios, desplegados en la nube mediante tecnologías AWS ECS + Fargate, API Gateway, Amazon RDS, y otros servicios de Amazon Web Services. Esta transformación responde a necesidades estratégicas detectadas por la empresa: mejorar la escalabilidad, disponibilidad, rendimiento, seguridad y mantenibilidad del sistema de educación en línea.

El presente documento contempla los aspectos funcionales y no funcionales del sistema, especifica los interfaces, y describe las dependencias tecnológicas, restricciones, características del usuario objetivo y consideraciones futuras.

**1.1. Propósito**

El presente documento tiene como propósito definir detalladamente los requisitos del sistema para la migración del sistema monolítico actual de EduTech Innovators SPA hacia una arquitectura basada en microservicios. Este cambio busca mejorar la escalabilidad, disponibilidad, rendimiento y mantenibilidad del sistema para responder a la creciente demanda nacional y garantizar la continuidad operativa de la empresa.

**1.2. Ámbito del Sistema**

El sistema abarca todas las funcionalidades críticas de EduTech Innovators SPA, incluyendo la gestión de usuarios, cursos, contenidos, soporte técnico, administración del sistema, monitoreo, gestión de incidencias y relación con los clientes vía web. La solución propuesta considerará la interoperabilidad entre servicios, el uso de una base de datos MySQL, y el despliegue en entornos distribuidos basados en contenedores(Buckets) y servicios en la nube.

**1.3. Definiciones, Acrónimos y Abreviaturas**

* SPA: Sociedad por Acciones
* API: Interfaz de Programación de Aplicaciones (Application Programming Interface)
* BD: Base de Datos
* UI: Interfaz de Usuario
* DevOps: Desarrollo y Operaciones
* CI/CD: Integración Continua / Entrega Continua
* SSO: Inicio de sesión único (Single Sign-On)
* REST: Transferencia de Estado Representacional (Representational State Transfer)

**1.4. Referencias**

* IEEE Std 830-1998 – Recomendación para la especificación de requisitos de software.
* Documentación oficial de Spring Boot, Docker, AWS ECS + Fargate y MySQL.
* Normativa nacional de protección de datos (Ley N° 19.628).
* Casos de éxito de migración a microservicios (Netflix, Amazon, Spotify).

**1.5. Visión General del Documento**

Este documento está estructurado de la siguiente manera:

* Sección 1: Introducción del proyecto y sus objetivos.
* Sección 2: Descripción general del producto.
* Sección 3: Requisitos específicos del sistema, incluyendo funcionales, no funcionales, y de interfaces.

**2. Descripción General**

El sistema representa la evolución arquitectónica de la plataforma tecnológica de EduTech Innovators SPA, migrando de un modelo monolítico a una arquitectura basada en **microservicios desacoplados**. Cada servicio será desplegado de forma independiente mediante contenedores Docker orquestados con **AWS ECS + Fargate**, permitiendo un escalado horizontal eficiente y un mantenimiento granular.

La solución contempla una **separación por dominios funcionales**, los cuales incluyen: usuarios, cursos, evaluaciones, reportes, soporte técnico y notificaciones. La capa de entrada estará gestionada mediante **Amazon API Gateway**, el cual centraliza el control del tráfico, aplica políticas de autenticación mediante **JWT** y enruta solicitudes a los servicios correspondientes.

Cada microservicio mantendrá su propia base de datos relacional aislada, gestionada por **Amazon RDS (MySQL 8.x)**, y se comunicará con otros servicios mediante REST APIs internas o colas de eventos asincrónicas, utilizando **Amazon SQS o EventBridge**. El sistema contará con herramientas de integración continua y entrega continua (**GitHub Actions**) y será monitoreado en tiempo real mediante **AWS CloudWatch**.

**2.1. Perspectiva del Producto**

El sistema representa una evolución tecnológica de una plataforma educativa monolítica hacia una arquitectura distribuida basada en microservicios. Esta solución busca descentralizar las responsabilidades del sistema, permitiendo una mejor escalabilidad, rendimiento y mantenimiento modular de los componentes.

**2.2. Funciones del Producto**

* Registro e inicio de sesión de usuarios.
* Gestión de perfiles de usuario y permisos.
* Administración de cursos, contenido y evaluaciones.
* Soporte técnico y gestión de incidencias.
* Generación de reportes e informes.
* Interacción con usuarios vía foros y mensajería.
* Monitoreo del sistema y respaldos automáticos.

**2.3. Características de los Usuarios**

Esta subsección describirá las características generales de los usuarios del producto, incluyendo nivel educacional, experiencia y experiencia técnica.

* Administrador del sistema: Control total sobre usuarios, permisos y configuración técnica.
  + Nivel educacional superior, Mas de 4 años acordes en cargos similares, y debe tener conocimientos en linux, Sql y Windows nivel avanzado
* Gerentes de cursos I y II: Control y supervisión de contenido, instructores y reportes.
  + Nivel educacional superior, Debe tener conocimientos en administración de empresas, auditorías, carreras similares, más de 4 años en cargos similares, y debe tener cursos de liderazgo.
* Personal de logística de soporte: Gestión de incidencias y recursos tecnológicos.
  + Carrera técnico profesional, más de 2 años de experiencia en cargos similares, Conocimiento nivel intermedio en linux, windows y Sql.
* Clientes/Estudiantes: Acceso a cursos, evaluaciones, soporte y perfil personal.
  + Debe ser mayor de 18 años y tener la educación media completa.

**2.4. Restricciones**

* Comunicación interna mediante REST y colas asincrónicas AWS.
* Cumplimiento obligatorio de la Ley N° 19.628 de Protección de Datos Personales.
* Recursos humanos limitados por calendario académico del equipo de desarrollo.
* El sistema debe ser desplegado utilizando servicios de nube compatibles con contenedores (como AWS, AWS).
* La base de datos principal será MySQL.
* Se requiere disponibilidad 24/7 con redundancia para asegurar la continuidad.
* Los microservicios deben comunicarse mediante API REST o gRPC.

**2.5. Suposiciones y Dependencias**

#### 2.5 Suposiciones y Dependencias

* Los usuarios disponen de acceso estable a internet.
* El sistema se desplegará sobre servicios administrados de AWS.
* El equipo cuenta con conocimientos previos en Java, Python y React.
* Se asumirá el uso de herramientas de versionamiento, documentación y despliegue continuo.

**2.6. Requisitos Futuros**

* Integración con sistemas de pago nacionales e internacionales.
* Soporte multilingüe (español, inglés y portugués).
* Aplicación móvil complementaria para estudiantes.
* Incorporación de analítica avanzada basada en IA para personalización de contenido.
* Integración con sistemas de videoconferencia (como Zoom, Google Meet).
* Posibilidad de habilitar certificados digitales con validez nacional.

**3. Requisitos Específicos**

Esta sección detalla los requerimientos técnicos del sistema, organizados en requisitos de interfaces, funcionales, no funcionales y otros aspectos relevantes. Los requisitos están formulados para ser claros, verificables, rastreables y viables técnicamente en el contexto de una arquitectura de microservicios desplegada en Amazon Web Services.

**3.1 Requisitos comunes de los interfaces**

#### 3.1.1 Interfaces de usuario

* El sistema debe proporcionar una interfaz web responsiva compatible con navegadores modernos (Chrome, Firefox, Edge, Safari).
* La experiencia debe estar optimizada tanto para escritorio como para dispositivos móviles.
* Cada usuario debe acceder a un panel personalizado según su rol (Administrador, Gerente de Cursos, Soporte, Estudiante).
* El sistema debe incluir notificaciones emergentes (toasts) e indicadores visuales para confirmar acciones del usuario.
* La interfaz debe incluir funciones de accesibilidad (contraste alto, etiquetas ARIA, navegación por teclado).
* El sistema debe permitir autenticación multifactor para usuarios con privilegios administrativos.

#### 3.1.2 Interfaces de hardware

* El sistema debe ser compatible con impresoras estándar para emisión de reportes y certificados.
* Debe considerar integración opcional con lectores de códigos QR para validación presencial.
* Los servidores virtuales deben tener especificaciones mínimas: 4 vCPU, 8 GB RAM, SSD, y conectividad de 1 Gbps.
* Todos los servicios deben ejecutarse sobre contenedores Docker administrados con AWS Fargate.

#### 3.1.3 Interfaces de software

* Cada microservicio debe exponer una API RESTful sobre HTTP/HTTPS documentada en Swagger (OpenAPI).
* El frontend se desarrollará con React + Tailwind CSS.
* Los servicios backend se construirán con Spring Boot (Java) para servicios complejos y FastAPI (Python) para servicios ligeros.
* Se utilizará Amazon API Gateway como pasarela de entrada, con autenticación basada en JWT.
* Las bases de datos serán instancias separadas en Amazon RDS (MySQL 8.x).
* Se utilizarán colas asincrónicas como Amazon SQS o EventBridge para eventos críticos.
* El control de versiones se realizará mediante GitHub, incluyendo ramas, pull requests y control de cambios.
* Los despliegues se automatizarán con GitHub Actions, integrados con ECR, ECS y entornos separados de staging y producción.

#### 3.1.4 Interfaces de comunicación

* La comunicación entre microservicios será vía API Gateway y eventos asincrónicos.
* Las conexiones deben estar cifradas con TLS 1.2 o superior.
* Se debe permitir el uso de WebSockets para interacción en tiempo real en el módulo de soporte.
* El sistema debe enviar notificaciones por correo a través de servicios SMTP como Amazon SES o SendGrid.
* Toda la documentación de endpoints debe estar disponible mediante Swagger en cada entorno

**3.2 Requisitos funcionales**

Ver documento excel.

**3.3 Requisitos no funcionales**

Ver documento excel.

**3.3.1 Requisitos de rendimiento**

* El sistema debe garantizar un tiempo de respuesta inferior a 1 segundo para el 90% de las peticiones.
* Debe soportar al menos 500 usuarios concurrentes sin degradación perceptible del rendimiento.
* Las operaciones masivas (como generación de reportes) deben ejecutarse en menos de 10 segundos.
* Cada microservicio debe escalar horizontalmente según demanda (autoescalado).
* Uso de caché con Redis para consultas frecuentes.

**3.3.2 Seguridad**

* El sistema debe cumplir con los principios de seguridad: confidencialidad, integridad y disponibilidad (CIA).
* Los datos personales deben ser cifrados en tránsito (TLS) y en reposo (AES-256).
* Implementación de autenticación OAuth 2.0 y autorización basada en roles (RBAC).
* Integración con firewall de aplicaciones (WAF) y sistema de detección de intrusos.
* Validación exhaustiva de entradas para prevenir ataques como XSS, CSRF e inyección SQL

**3.3.3 Fiabilidad**

El sistema debe mantener una tasa de disponibilidad mínima del 99,9% mensual.  
Cada microservicio debe contar con políticas de retry automático ante fallos.  
Se implementará logging centralizado y monitoreo continuo con AWS CloudWatch.  
Debe existir tolerancia a fallos mediante replicación activa-pasiva.

**3.3.4 Disponibilidad**

Los servicios críticos (autenticación, inscripción, pagos, soporte) deben estar disponibles 24/7.

Backups automáticos diarios y verificación semanal de restauración.  
Uso de balanceadores de carga para distribución de peticiones.  
Se usará una arquitectura de alta disponibilidad con múltiples réplicas y zonas de disponibilidad.

**3.3.5 Mantenibilidad**

Cada microservicio debe seguir el principio de responsabilidad única (SRP).  
Código documentado y versionado siguiendo las buenas prácticas de desarrollo (semver).

Incorporación de pruebas unitarias, de integración y de regresión automatizadas.

Las actualizaciones deben realizarse sin interrupciones utilizando pipelines CI/CD.

Módulos desacoplados que permitan mantenimiento independiente.

Las tareas de mantenimiento las debe realizar el encargado de TI.

**3.3.6 Portabilidad**

El sistema debe ejecutarse sobre contenedores Docker, lo que permite su portabilidad entre distintos entornos de nube compatibles con OCI (Open Container Initiative), incluyendo ambientes on-premise, staging, testing y producción.

La arquitectura debe ser independiente de proveedores específicos de hardware o sistema operativo, permitiendo su ejecución en cualquier infraestructura compatible con servicios de AWS (tales como ECS con Fargate, EC2 o entornos locales con Docker Compose).

Todos los microservicios deben estar empaquetados como imágenes estándar y almacenados en un repositorio Docker (Amazon ECR), permitiendo su despliegue sin modificaciones en distintos entornos.

La configuración del sistema (variables de entorno, secretos, endpoints) debe estar desacoplada del código fuente y ser gestionada mediante sistemas como AWS Parameter Store o Secrets Manager.

El sistema debe ser capaz de exportar datos en formatos estándar como **JSON**, **CSV** y **XML** para asegurar interoperabilidad con herramientas de terceros.

Debe garantizarse que el despliegue completo pueda replicarse automáticamente mediante scripts de infraestructura como código (IaC), utilizando herramientas como **AWS CloudFormation** o **Terraform**, reduciendo el tiempo de portabilidad y evitando configuraciones manuales.

**3.4 Otros Requisitos**

Cumplimiento con la Ley 19.628 sobre protección de la vida privada (Chile) y regulaciones de privacidad de datos.

El sistema debe contar con manuales técnicos y de usuario para cada perfil involucrado.

Debe soportarse la integración con sistemas de facturación electrónica (SII).

Traducción multilingüe (español e inglés) desde el frontend.

Disponibilidad de APIs públicas para terceros con autenticación controlada.

Integración con Google Analytics para métricas de uso.

Tiempos de respuesta y errores deben almacenarse y graficarse.

Se debe realizar una prueba piloto con usuarios reales antes del despliegue final.