Integrantes

Camilo Poblete Gómez
Nicolás Carrasco
Carlos López

Asignatura

Full Stack I
Sección 010D

Profesor

Eduardo Antonio Baeza Escobar

04/04/2025

Indice contenidos

1. Análisis de Requerimientos

1.1 Requerimientos del sistema

1.1.1 Funcionales

1. El sistema debe permitir crear, actualizar, editar y eliminar cuentas de usuarios.
2. El sistema debe permitir crear, actualizar y eliminar cursos del catálogo.
3. El sistema debe pedir iniciar sesión a través de un login para acceder al sistema.
4. El sistema debe permitir la comunicación entre profesores y alumnos.
5. El sistema debe permitir monitorear el progreso de los estudiantes.
6. El sistema debe permitir que los usuarios reporten eventuales fallas en el sistema.
7. El sistema debe permitir al administrador de sistemas gestionar cuentas de usuario.
8. El sistema debe almacenar los datos de los usuarios en una base de datos.
9. El sistema debe gestionar matrículas y pagos.
10. El sistema debe permitir acceder a descuentos y cupones de pago.
11. El sistema debe mostrar en pantalla cursos disponibles en el catálogo.

1.1.2 No funcionales

1. El sistema debe estar disponible las 24 horas del día los 7 días de la semana.
2. El sistema debe garantizar el resguardo de la información de los usuarios con un sistema de cifrado de datos en un 100%.
3. El sistema debe ser capaz de operar con más de 2.000.000 de usuarios simultáneos sin presentar alguna baja sustancial de rendimiento en su operatividad.
4. Las solicitudes y modificaciones de datos en las bases de datos deben ser recibidas y modificadas en menos de 5 segundos.
5. Las copias de seguridad y respaldo de datos se deben hacer cada 24 horas a las 18:00
6. El sistema mostrará el gasto de recursos en la interfaz de monitoreo del sistema con una actualización periódica con 5 segundos de desfase.

1.2 Entrevistas

1.2.1 Rol Administrador del sistema

1. ¿Cómo considera usted que funciona la actual plataforma de EduTech Innovators SPA?

2. ¿Cuáles son las principales desventajas que usted como administrador de sistemas ve en un sistema monolítico como el actual?

3. ¿Ve en un sistema basado en microservicios una solución a las actuales problemáticas que presenta la plataforma?

1.2.2 Rol Gerente de cursos I y II

1. ¿Cómo considera usted que funciona la actual plataforma de EduTech Innovators SPA?

2. ¿La plataforma responde adecuadamente a sus necesidades?

3. ¿Ha tenido que contactar con soporte para resolver alguna problemática relacionada a la plataforma?

4. ¿Qué cosas mejoraría usted del actual sistema?

1.2.3 Rol Alumno

1. ¿Cómo considera usted que funciona la actual plataforma de EduTech Innovators SPA?

2. ¿La plataforma responde adecuadamente a sus necesidades?

3. ¿Ha tenido que contactar con soporte para resolver alguna problemática relacionada a la plataforma?

4. ¿Qué cosas mejoraría usted del actual sistema?

1.2.4 Rol Profesor

1. ¿Cómo considera usted que funciona la actual plataforma de EduTech Innovators SPA?

2. ¿La plataforma responde adecuadamente a sus necesidades?

3. ¿Ha tenido que contactar con soporte para resolver alguna problemática relacionada a la plataforma?

4. ¿Qué cosas mejoraría usted del actual sistema?

2. Análisis del sistema actual

2.1 Análisis del sistema monolítico existente

2.1.1 Consideraciones generales de un sistema monolítico

Una arquitectura monolítica es un modelo de desarrollo de software tradicional que utiliza un código base para realizar varias funciones dentro de un sistema. Las aplicaciones monolíticas suelen constar con una estructura más simple y jerárquica. Los desarrolladores crean todas las partes del software en un único código base.

Es más fácil empezar con aplicaciones monolíticas, ya que no se requiere mucha planificación y recursos iniciales. Sin embargo, a medida que el sistema crece, la aplicación puede volverse compleja y difícil de actualizar o modificar.

2.1.2 Sistema monolítico existente en EduTech Innovators SPA

Para contextualizar este caso, EduTech Innovators SPA es una empresa chilena dedicada a la creación y distribución de plataformas educativas. Recientemente, debido a un aumento de la demanda de sus servicios, han decidido expandir sus operaciones con oficinas en La Serena y Valparaíso.

EduTech Innovators SPA actualmente posee una arquitectura de software de tipo monolítico. Esto quiere decir que todas las dependencias del sistema se encuentran en un único código base. Si bien esto en un principio resultó conveniente ya que un sistema monolítico es más sencillo y económico de implementar, con el paso del tiempo y a medida que el sistema ha ido creciendo y complejizando de acuerdo a las nuevas necesidades esto ha derivado en inconvenientes significativos de funcionamiento.

2.2 Principales causas de fallos y sobrecargas en el sistema

Debido a la creciente expansión de EduTech Innovators SPA su sistema monolítico no ha podido adaptarse de igual manera a la creciente demanda. Los sistemas monolíticos son más complejos de escalar y/o crecer ya que al tener todas sus dependencias en un mismo código base es todo el sistema el que debe escalar y/o crecer en la misma proporción. Lo anterior deriva en un deficiente uso de los recursos. Además, el sistema monolítico actual limita la capacidad del sistema de tener una comunicación fluida entre los distintos componentes que lo componen lo cual resulta perjudicial debido a la actual necesidad de crecimiento que tiene la empresa.

Finalmente, el enfoque monolítico limita la capacidad de la organización a la hora de introducir nuevas funcionalidades y tecnologías lo cual es una clara desventaja ya que el constante avance de la tecnología va dejando obsoletos los sistemas que no son capaces de adaptarse a las nuevas tendencias.

3. Diseño de la Nueva Arquitectura

3.1 Nueva arquitectura basada en microservicios

3.1.1 Arquitectura de microservicios a utilizar

Para nuestro proyecto en EduTech Innovators SPA hemos analizado varias alternativas de arquitecturas basadas en microservicios. En razón de lo anterior consideramos que una arquitectura basada en API Gateway es la mejor opción para nuestro proyecto porque facilita la comunicación, la seguridad y por sobre todo la escalabilidad tan necesaria para las nuevas necesidades de la empresa. La arquitectura basada en API Gateway es un enfoque recurrente en sistemas con microservicios, donde un API Gateway actúa como intermediario entre el usuario y los microservicios internos del sistema. Esta arquitectura es además muy utilizada en instituciones educativas alrededor del mundo (Ejemplo Blackboard Inc.).

3.1.2 Explicación de arquitectura basada en API Gateway

¿Qué es un API Gateway?

Es un servicio centralizado que gestiona todas las solicitudes de los usuarios y las dirige a los microservicios correspondientes. Actúa como un único punto de entrada de datos en la plataforma lo que simplifica la comunicación.

¿Cómo funciona?

1. Un usuario realiza una solicitud al sistema.

2. El API Gateway recibe la solicitud y la valida.
3. El API Gateway envía la solicitud al microservicio correspondiente.
4. El microservicio procesa la solicitud, obtiene la información de la base de datos y responde al API Gateway.
5. El API Gateway devuelve la respuesta al usuario.

¿Beneficios de usar API Gateway?

1. Escalabilidad: Permite aumentar el tráfico de datos sin colapsar el sistema.
2. Centraliza el acceso: Un único punto de entrada para todos los microservicios.
3. Distribución: Mejora la comunicación entre los distintos componentes del sistema.
4. Registro: Facilita auditorías y análisis de datos.

3.1.3 Microservicios a implementar

Los microservicios son pequeños códigos de software que realizan actividades particulares dentro de un sistema más grande. Esto permite dividir las funciones internas del sistema en pequeños bloques mejorando la fluidez de comunicaciones entre los distintos componentes.

1. **Microservicio para login.**

   Microservicio encargado de la gestión de ingreso de los distintos usuarios a la plataforma. Este proceso se lleva a cabo mediante autenticación o verificación de la identidad. Puede ser mecanismo interno del sistema o gestor externo como google authenticator u otro.

2. **Microservicio para gestión de cuentas de usuario.**

   Microservicio encargado de la gestión de cuentas de usuario. Ya sea la creación, eliminación y modificación de cuentas de los distintos usuarios de la plataforma.

3. **Microservicio para la gestión de matrículas, mensualidades y pagos.**

   Microservicio encargado de gestionar pagos como matrícula, mensualidades o anualidades. Puede ser mecanismo interno del sistema o gestor externo como servipag, khipu u otro.

4. **Microservicio para gestión de cursos.**

   Microservicio encargado exclusivamente de todas las operaciones relacionadas con la administración de los cursos disponibles en la plataforma. Ya sea creación, actualización y eliminación de cursos y sus contenidos.

5. **Microservicio para gestión del rendimiento académico.**

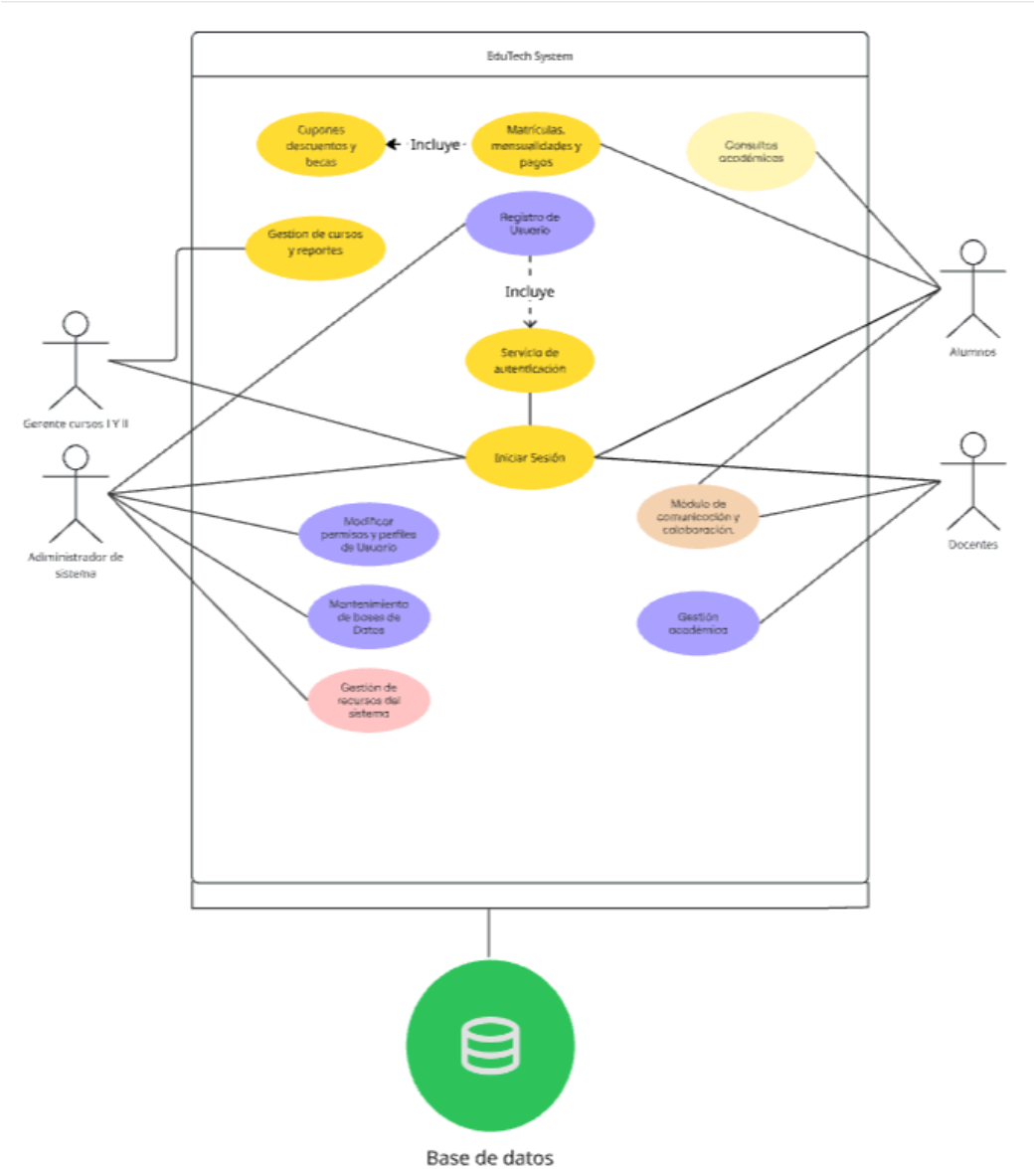
   Microservicio encargado de gestionar el progreso académico de los alumnos generando estadísticas e informaciones generales sobre evaluaciones, certificados, cursos aprobados o en progreso, etc.

3.1.4 Resumen de beneficios de nueva arquitectura

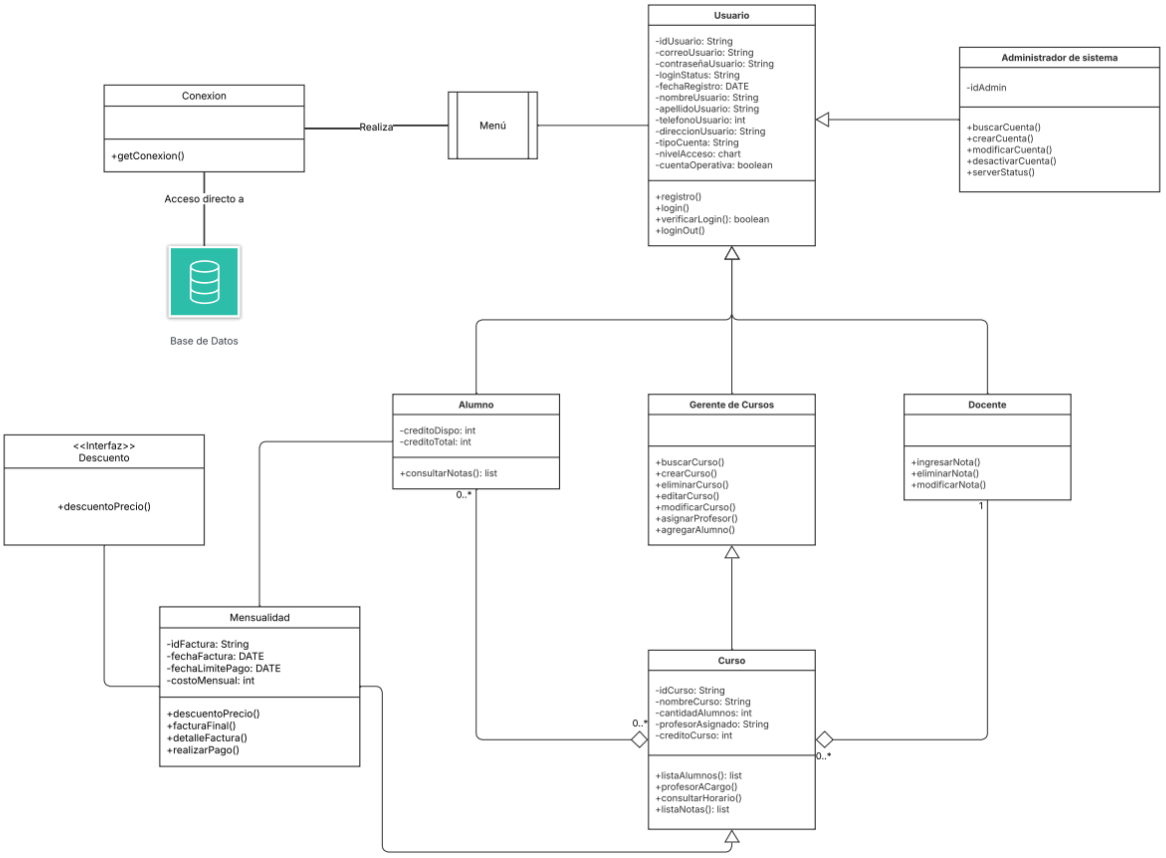
Creemos que una arquitectura de microservicios basada en API Gateway para EduTech Innovators SPA es una buena manera de aumentar la escalabilidad del sistema, centralizar y distribuir de mejor manera las comunicaciones entre los distintos componentes del software y llevar un mejor registro de los datos almacenados. Todo lo anterior resulta beneficioso para las nuevas necesidades y proyecciones de la empresa.

3.2 Diagramas de casos de uso, de clases y de despliegue.

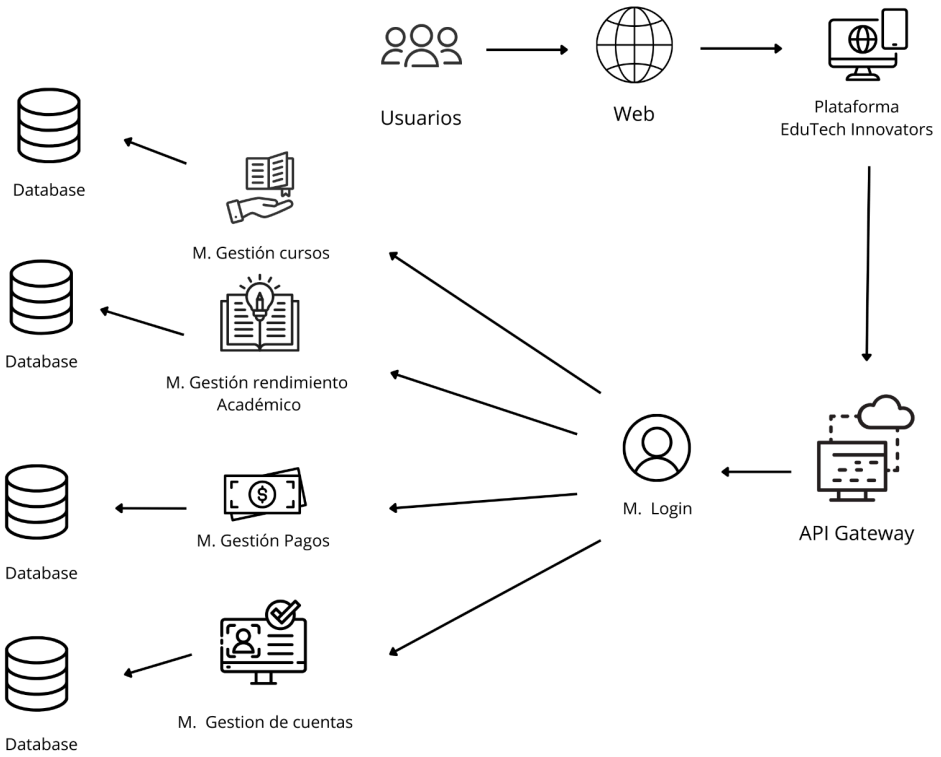
3.2.1 Diagrama casos de uso



3.2.2 Diagrama de clases



3.2.3 Diagrama de despliegue



4. Planificación de la Migración

4.1 Desde sistema monolítico a nueva arquitectura de microservicios

4.1.1 Identificación de Problemas

Para crear un plan de migración hacia la nueva arquitectura de microservicios (Arquitectura basada en Api Gateway) se necesita identificar ciertos problemas que tiene la arquitectura actual (Arquitectura Monolítica), que está dando fallos de rendimiento y no permite escalabilidad. Estas consecuencias negativas son provocadas porque este agrupa todas las funcionalidades en un solo bloque de software. A continuación enumeramos los problemas más destacados.

1. Identificación de Cuellos de Botella

Como se mencionó anteriormente, esta empresa sufrió un crecimiento brusco en sus operaciones tanto así que su infraestructura tecnológica se ve sometida a mayores exigencias en funcionalidades como altas inscripciones o pruebas en línea, el sistema alcanza su límite de procesamiento, resultando en fallos, lentitud en las respuestas o incluso colapsa. En el sistema no existe un balanceo de carga eficaz, lo que implica que todas las solicitudes dependen de una sola aplicación provocando nula tolerancias a fallos ni recuperación modular quedando todo el sistema inactivo, esto dificulta su escalabilidad por partes. Por ejemplo, si solo se necesita más capacidad para la gestión de cursos, hay que escalar todo el sistema completo, lo que es caro e ineficiente.

2. Dependencias Internas entre Módulos

Las dependencias internas entre módulos ocurren cuando un módulo de un sistema necesita usar o comunicarse con otro módulo para cumplir con su función. Es decir, un módulo depende de otro para realizar una tarea específica, acceder a datos o ejecutar una funcionalidad.
En este caso la empresa tiene todos los módulos fuertemente acoplados, lo que significa que uno depende de otro. Por ejemplo, el módulo de “Evaluaciones” puede depender directamente de funciones del módulo de “Usuarios” y si uno de estos dos llega a presentar inconvenientes el otro también los tendrá. Además si se modifica o se gestiona de mala forma una parte del código también podría perjudicar otras partes, ya que todo está centralizado en un mismo bloque, dificultando el mantenimiento, las actualizaciones y la introducción de nuevas funcionalidades.

3. Sobrecarga en la Base de Datos Centralizada

Todas las transacciones y consultas se realizan sobre una sola instancia de base de datos, lo que provoca obstrucción en períodos de alta demanda, consultas poco eficaces y tiempos de respuesta muy largos particularmente en actividades de gran magnitud como generación de informes de evaluaciones o inscripciones masivas. Además conlleva a la falta de escalabilidad ya que el aumento de la plataforma conlleva mayor presión en una única base de datos, restringiendo el rendimiento general.

4.1.2 Planificación de Microservicios

Con los principales inconvenientes ya identificados, podemos proceder al próximo paso. Para avanzar hacia la nueva arquitectura previamente mencionada, resulta crucial establecer adecuadamente los microservicios a implementar. Esto implica segmentar la arquitectura monolítica en elementos funcionales y autónomos que puedan transformarse en microservicios independientes, disminuyendo de esta manera la complejidad del código. Esta separación es muy importante, porque posibilita una división precisa y ordenada de responsabilidades, simplifica el mantenimiento, además potencia la escalabilidad individual de cada servicio y reduce la dependencia entre ambos.

A continuación, se describen a profundidad los servicios más fundamentales en la empresa y sus beneficios:

1. Microservicio de Registro y Login de Usuario.

Uno de los beneficios más significativos de segmentar este módulo (Registro y Login de Usuario) en un microservicio es la concentración de las operaciones vinculadas al inicio de sesión y al control de accesos. Esta centralización optimiza la administración de seguridad al reunir todos los procedimientos de autenticación en un solo lugar, lo que disminuye la complejidad en otros microservicios, dado que no requieren la implementación de sus propios sistemas de autenticación y autorización. En otras palabras, facilita la implementación y administración de un grupo uniforme de políticas de seguridad (tales como requisitos de contraseña, bloqueo de cuentas después de varios intentos fallidos, autenticación de dos factores, etc.) en un único sitio. Además, reduce los peligros de seguridad al tener un solo punto de entrada que puede ser supervisado y auditado con más sencillez.

2. Microservicio de gestion de cuentas de usuario

Uno de los motivos principales por el cual se divide este módulo (Microservicio de gestión de cuentas de usuario) es precisamente por su funcionalidad principal, en la cual se encarga de la administración de la información de los usuarios registrados, lo que incluye datos personales, perfiles y permisos. Al centralizar esta lógica, se asegura que la gestión de cuentas en la plataforma sea consistente y segura en toda la aplicación. Además este enfoque permite una mejor escalabilidad, seguridad y gestión de datos. Esto proporciona un

control robusto sobre los permisos y la seguridad del sistema. Al definir claramente qué puede y qué no puede hacer cada usuario, se minimizan los riesgos de acceso no autorizado y se protege la información sensible.

3. Microservicio para Matrículas, Mensualidades y Pagos

Este microservicio es fundamental en este sistema para tener una sostenibilidad clara de toda la empresa Edutech ya que tiene beneficios como gestionar todo el ciclo financiero relacionado con la participación del estudiante en la plataforma. Este microservicio de pago en el sistema también abarca la inscripción a cursos, el control de pagos recurrentes (mensualidades), y la validación de pagos como matrículas. Además es responsable de emitir comprobantes de pago y mantener actualizados los estados financieros de cada cuenta de usuario. Otro beneficio muy importante es la mejora en la seguridad y trazabilidad de las operaciones económicas, al permitir el monitoreo y auditoría específica de este flujo de datos.

4. Microservicio para la Gestión de Cursos

Este microservicio es el corazón académico de la plataforma educativa. Tiene como objetivo principal centralizar toda la lógica relacionada con la administración del contenido académico. Esto incluye la creación, edición y eliminación de cursos.
La separación de esta funcionalidad en este sistema tiene un beneficio muy importante que es el ajuste específico a la demanda de los estudiantes accediendo a cursos, sin consumir recursos innecesarios en otras áreas, mejorando el rendimiento general en el sistema ya que las consultas y operaciones relacionadas con cursos se manejan de forma aislada reduciendo la complejidad del código.

5. Microservicio para la Gestión del Rendimiento Académico

Este microservicio importante tiene como finalidad principal administrar y analizar el progreso educativo de los estudiantes dentro de la plataforma EduTech. Este mismo se encarga de generar información clave sobre sobre evaluaciones, cursos completados o en progreso, emisión de certificados y estadísticas generales relacionadas con el desempeño académico de cada usuario. Cada estudiante podrá tener una visión clara de toda esta información personalizada a tiempo real en la plataforma educativa.
Uno de los principales beneficios de haber separado esta funcionalidad del sistema monolítico y haberla implementado en un microservicio radica en la posibilidad de procesar grandes volúmenes de información sin afectar el rendimiento de otros módulos del sistema.

4.1.3 Implementación de Microservicios y Api Gateway

Con los microservicios ya planificados y definidos tras la división del sistema monolítico, el siguiente paso es su implementación gradual y preparación del entorno. Este proceso implica desarrollar, desplegar e integrar cada microservicio de forma independiente, asegurando una transición ordenada y sin interrupciones en el servicio. Estos microservicios serán desarrollados como una unidad autónoma, la mayoría con su propia base de datos y con su Api Rest (interfaz de programación de aplicaciones). Esta Api Rest permite la comunicación entre microservicios (o clientes externos) en una arquitectura distribuida. Al implementar los microservicios de esta forma permite que puedan ser escalados, mantenidos y actualizados de manera individual sin afectar a los demás servicios del sistema.
Cada microservicio se alojará como contenedor Docker, orquestado mediante Kubernetes, facilitando así la escalabilidad, la tolerancia a fallos y la implementación en distintos entornos.

Una vez implementados, los microservicios serán expuestos únicamente a través del API Gateway que es un componente clave, el cual actúa como una puerta de entrada única para todas las solicitudes que llegan desde los usuarios hacia los distintos microservicios que componen el sistema. Este simplifica y controla cómo se accede a estos servicios, funcionando como un intermediario centralizado. Además verifica si un usuario tiene acceso permitido al sistema y decidirá qué partes del sistema puede utilizar, basándose en su rol.
De este modo, el sistema mantiene una única interfaz de acceso para el cliente, manteniendo la simplicidad y la seguridad.
Durante la implementación del API Gateway, se establecerán pipelines de Integración Continua (CI) y Despliegue Continuo (CD), que consisten en una serie de pasos automatizados que se ejecutan cada vez que se realizan cambios en el código del sistema. Estos pasos pueden incluir: compilar el código, ejecutar pruebas automáticas, verificar aspectos de seguridad y desplegar en entornos de pruebas o producción. El objetivo principal es prevenir errores, detectar fallos de forma temprana, y asegurar que los nuevos cambios no rompan funcionalidades existentes, permitiendo despliegues seguros, rápidos y confiables.
Además no es estrictamente obligatorio pero si es muy recomendable, el implementar monitoreo y logging centralizado, este proceso tiene como finalidad supervisar el comportamiento y rendimiento del sistema en tiempo real. Esto no solo permite una gestión eficiente del Gateway y los microservicios, sino que además garantiza un mejor control, mayor seguridad operativa y facilidad en el mantenimiento del sistema.

4.1.4 RETIRO FINAL DEL SISTEMA MONOLITICO

Lo más importante de todo, para el lanzamiento hacia el nuevo sistema de micro servicio (Api Gateway) es hacerlo de poco a poco y no todo de una vez. Por eso mismo se ha optado por aplicar la estrategia que se mencionó anteriormente. Esta metodología propone una transición progresiva y controlada, evitando interrupciones en el funcionamiento del sistema mientras se moderniza su infraestructura tecnológica. Una vez que todos los microservicios han sido implementados, probados e integrados al API Gateway, se procede a retirar completamente el sistema monolítico. A partir de este momento, el sistema opera exclusivamente con una arquitectura distribuida basada en microservicios. Con todo esto ya

aclarado, la migración permite a EduTech Innovators SPA contar con un sistema más escalable, seguro y mantenible.

4.2 Riesgos y plan de mitigación

4.2.1 Riesgos de implementación

Al momento de migrar de un sistema monolítico a microservicios basada en Api Gateway pueden surgir diferentes riesgos para implementar estos servicios, Estos pueden ser críticos si no se tiene una prevención correcta y podría impactar negativamente el funcionamiento de la plataforma educativa EduTech Innovators SPA. A continuación, se detallarán los tres riesgos más importantes junto con sus respectivas estrategias de mitigación:

1. Interrupción del Servicio

Existe una gran posibilidad de riesgo de que, durante la migración o implementación de los microservicios, se produzcan caídas o interrupciones del sistema que afecten la disponibilidad para los usuarios finales.

Una mitigación fundamental para esto se puede adoptar una estrategia de migración progresiva como se había mencionado anteriormente, que permita introducir microservicios gradualmente mientras el sistema monolítico sigue operativo. Además se puede establecer un entorno de pruebas, idéntico al entorno de producción para verificar el comportamiento antes del despliegue final y así también poner a prueba la pérdida de datos que puede tener. Se recomienda tener un respaldo completo de datos antes de realizar el entorno

2. Fallas en la Comunicación entre Microservicios

Al momento de hacer la división del sistema monolítico para tener servicios autónomos pueden surgir ciertos riesgos como problemas de comunicación, incompatibilidad de APIs, o tiempos de respuesta muy prolongados.
Una mitigación recomendable para estos problemas es utilizar estándares de comunicación como REST o mensajería asíncrona con control de errores. También se puede aplicar técnicas de versionado de APIs para garantizar compatibilidad entre servicios.

3. Problemas de Seguridad y Control de Acceso

Al momento de distribuir el sistema monolítico en múltiples microservicios este se expone a más superficie de ataques como por ejemplo el de Denegación de Servicio (DoS), en el cual El API Gateway y los microservicios pueden ser bombardeados con solicitudes masivas para intentar sobrecargar el sistema y dejarlo fuera de línea. Esto puede dificultar el control unificado de la autenticación y autorización.
Para mitigar este riesgo se puede configurar el API Gateway con políticas de seguridad estrictas, incluyendo validación de tokens, límite de peticiones y monitoreo de tráfico.