Requisitos del Sistema – Plataforma IGRIS

1. Requisitos Funcionales

📌 Módulo de Autenticación

- Los estudiantes y profesores deben poder registrarse e iniciar sesión con correo y contraseña.
- Implementar autenticación con OAuth (Google/Microsoft) como opción alternativa.
- Los usuarios deben poder restablecer su contraseña mediante correo electrónico.
- El sistema debe generar y validar tokens JWT para sesiones seguras.

📌 Módulo de Gestión de Competencias y Avances

- Los profesores pueden crear, modificar y eliminar competencias.
- Los estudiantes pueden visualizar competencias asignadas y registrar sus avances.
- El sistema debe otorgar puntos y medallas según el desempeño del estudiante.
- Se debe generar un gráfico de progreso que refleje la evolución del estudiante.

📌 Banco de Ejercicios y Retos

- Cada competencia debe contar con un conjunto de ejercicios categorizados por dificultad (Básico, Intermedio, Avanzado).
- Los ejercicios pueden ser de opción múltiple, código o respuesta abierta.
- Al responder, la plataforma debe proporcionar retroalimentación automática.
- El sistema debe almacenar el historial de respuestas y avances del usuario.

📌 Sistema de Maratones y Clasificación

- Los estudiantes podrán participar en maratones de programación semanales.
- Los profesores podrán configurar maratones con tiempo límite y lista de problemas.
- El sistema debe evaluar automáticamente las respuestas y asignar puntuaciones.
- Integración con API externas para obtener nuevos problemas de programación.

📌 Sistema de Recompensas y Gamificación

- Implementación de medallas y rangos según el desempeño del usuario.
- Los estudiantes pueden desbloquear logros por participación y rendimiento.
- Generación de tablas de clasificación y rankings según desempeño.
- Las recompensas deben motivar la constancia y el aprendizaje progresivo.

---

2. Requisitos No Funcionales

📌 Usabilidad y Diseño

- La interfaz debe ser intuitiva, minimalista y accesible en dispositivos móviles y escritorio.
- Aplicación de principios UX/UI para mejorar la experiencia del usuario.
- Implementación de un sistema de notificaciones sobre avances y nuevos desafíos.

📌 Desempeño y Escalabilidad

- La plataforma debe manejar múltiples usuarios simultáneamente sin afectar el rendimiento.
- Optimización de consultas con Redis para mejorar tiempos de respuesta.
- Uso de WebSockets para actualización en tiempo real en maratones y ranking.
- Integración con Docker y Kubernetes para despliegue escalable.

📌 Seguridad

- Cifrado de contraseñas mediante bcrypt.
- Implementación de JWT para autenticación segura.
- Protección contra ataques de fuerza bruta y SQL Injection.
- Control de acceso basado en roles (estudiante, profesor, administrador).

📌 Disponibilidad y Conectividad

- El sistema debe estar disponible 24/7 en la nube con un uptime del 99.9%.
- Implementación de backups automáticos para evitar pérdida de información.
- Integración con herramientas de monitoreo para detectar fallos y optimizar rendimiento.

---

3. Restricciones y Suposiciones

- Se asume que los problemas de programación pueden ser obtenidos de una API externa.
- El desarrollo del sistema utilizará **FastAPI (Backend), React.js (Frontend) y PostgreSQL (Base de Datos).**
- La clasificación en el ranking dependerá de la cantidad de ejercicios y maratones completados.

- Los estudiantes y profesores solo pueden visualizar la información relacionada con su rol y actividades asignadas.