



Consolidación del Proyecto API en C++

Proyecto Prograthon: sistema integral de gestión académica con trazabilidad de competencias y concursos de programación

Rodolfo Ángel Gilart Pulido

Jhonatan Camilo Garrido Gomez

Laura Catalina Riveros Flórez

Julián Andrés Cuadros Muñoz

Universidad católica de Colombia

Desarrollo de software

Facultad de ingeniería

Bogotá D.C.

2025

Tabla de Contenido

1. Descripción Del Proyecto	6
1.0. Control De Cambios	6
1.1. Introducción	7
1.2. Planteamiento Del Problema	7
1.3. Pregunta Problema	7
1.4. Funcionalidad API	8
1.5. Objetivos	9
1.5.1. Objetivo general:	9
1.5.2. Objetivos específicos:	9
2. Tecnologías Utilizadas	10
2.1. Pila MERN	10
2.1.1. Mongo DB.	10
2.1.2. Express.	11
2.1.3. React.	11
2.1.4. Node.	11
2.2. JWT	12
2.3. API	12
2.4. Control De Versiones	13
3. Metodologías	16
3.1. Explicación de la metodología	16
3.2. Fases	17
3.3. Asignación De Roles	18
3.4. Requerimientos	19
3.4.1. Requerimientos Funcionales.	19
3.4.1. Requerimientos No Funcionales.	22
3.4.1.1 Rendimiento:	22
3.4.1.2 Mantenibilidad:	23
3.4.1.3 Usabilidad:	23
3.4.1.4 Fiabilidad:	24
3.4.1.5 Seguridad	24
3.4.1.6 Compatibilidad	25
3.5. Arquitectura	26
3.5.1. Casos De Uso.	26
3.5.2. Diagrama De Vista De Procesos.	26

3.5.2. Diagrama De Vista Logica.	27
3.5.2. Diagrama De Implementación.	28
3.5.2. Diagrama De Vista De Despliegue.	29
3.5.2. Diagrama De Paquetes.	30
4. Errores Y Soluciones	30
5. Objetivos Logrados	31
5.1. Requerimientos Funcionales logrados	31
5.2. Checklist	32
6. Bibliografía	32

Tabla de tablas

<i>Tabla 1 Control de cambios</i>	6
<i>Tabla 2 RF001</i>	19
<i>Tabla 3 RF002</i>	19
<i>Tabla 4 RF003</i>	19
<i>Tabla 5 RF004</i>	20
<i>Tabla 6 RF005</i>	20
<i>Tabla 7 RF006</i>	21
<i>Tabla 8 RF007</i>	21
<i>Tabla 9 RF008</i>	21
<i>Tabla 10 RF009</i>	22
<i>Tabla 11 RNF001</i>	22
<i>Tabla 12 RNF002</i>	23
<i>Tabla 13 RNF003</i>	23
<i>Tabla 14 RNF004</i>	24
<i>Tabla 15 RNF005</i>	24
<i>Tabla 16 RNF006</i>	25

Tabla de ilustraciones

<i>Ilustración 1 Roadmap_1</i>	13
<i>Ilustración 2 Roadmap_2</i>	13
<i>Ilustración 3 Backlog iteración 1</i>	14
<i>Ilustración 4 Backlog interacción 2</i>	14
<i>Ilustración 5 Backlog interacción 3</i>	15
<i>Ilustración 6 Burn up proyecto</i>	15
<i>Ilustración 7 Assignees issues</i>	16
<i>Ilustración 8 Fases del proyecto</i>	18
<i>Ilustración 9 Roles del proyecto</i>	18
<i>Ilustración 10 Casos de uso</i>	26
<i>Ilustración 11 Vista de procesos</i>	26
<i>Ilustración 12 Vista lógica</i>	27
<i>Ilustración 13 Implementación</i>	28
<i>Ilustración 14 Vista de despliegue</i>	29
<i>Ilustración 15 Paquetes</i>	30
<i>Ilustración 16 Revisión de requerimientos cumplidos</i>	31
<i>Ilustración 17 Checklist de calificación</i>	32

1. Descripción Del Proyecto

1.0. Control De Cambios

Tabla 1 Control de cambios

Versión del documento	Fecha	Estado del documento	Nombre del archivo	Cambios realizados	Responsable/s de las modificaciones
0.1	24/06/2025	En revisión	Documentación general del proyecto	Se agregan la introducción, planteamiento del problema, requerimientos y diagramas del proyecto	Catalina Julián Rodolfo Jhonatan
0.2	25/06/2025	Finalizado	Documentación general del proyecto	Se modifica el objetivo general y los objetivos específico en base a las recomendaciones del profesor	Catalina
0.3	25/06/2025	Finalizado	Documentación general del proyecto	Se detallan las fases de la metodología a seguir en el proyecto	Catalina Julián Rodolfo Jhonatan
0.4	25/06/2025	Finalizado	Documentación general del proyecto	Se modifican los diagramas del proyecto en base a las recomendaciones del profesor	Catalina Julián Jhonatan
0.5	25/06/2025	Finalizado	Documentación general del proyecto	Se modifica la tabla de los roles de cada integrante del proyecto	Jhonatan
0.6	30/06/2025	Finalizado	Documentación general del proyecto	Se modifican los requerimientos no funcionales de acuerdo con la ISO25010	Catalina

1.1. Introducción

Según el periódico ElColombiano Colombia es el tercer país con más aumento en el campo laboral de programadores en Latinoamérica. En el 2022 se presentó un aumento de programadores del 33%, de acuerdo con el informe el país alcanza un aumento de 663.000 personas. El ministerio de las MinTic (el ministerio colombiano de tecnologías y comunicación) estima que para este año actual (2025) existirá un déficit entre 68.000 y 112.000 desarrolladoras de software en el país.

Bogotá al ser una de las ciudades más importantes del país y con un alto desarrollo en el área tecnológica es la ciudad que lidera la demanda en el campo de la programación, concentrando un 78% de las oportunidades laborales (Clavijo, 2024).

La carrera de ingeniería en Sistemas en la actualidad es una de las más competidas y solicitadas por los estudiantes en la capital del país. Se estima que cerca de 1.257 egresados ingresan anualmente al campo laboral en la ciudad lo que refleja que esta carrera tiene una empleabilidad bastante grande (Tropicana, 2025).

1.2. Planteamiento Del Problema

En Colombia, existe una brecha crítica entre la formación académica en TI y las demandas del sector laboral. Según El Heraldo (2021), "cerca del 90% de los egresados en áreas informáticas ingresan al mercado laboral con habilidades de programación insuficientes" (p. 1A). Esto ha generado:

- Un desempleo del 18.5% en jóvenes del sector (Departamento Administrativo Nacional de Estadística [DANE], 2023, p. 34).
- 83,000 vacantes tecnológicas sin cubrir debido a la falta de competencias prácticas (Ministerio de Tecnologías de la Información y las Comunicaciones [Mintic], 2024, p. 12).

Algunas de las causas que han sido identificadas sobre este problema es:

- Desfase curricular: El 70% de las universidades priorizan contenidos teóricos sobre programación aplicada (Agencia Nacional de Evaluación de la Calidad y Acreditación [ANECA], 2022, p. 45).
- Falta de entrenamiento real: Solo el 20% de los estudiantes resuelven problemas similares a los usados en entrevistas técnicas (LinkedIn, 2023, Tendencias de Contratación TI).

1.3. Pregunta Problema

¿Cómo diseñar e implementar un aplicativo interactivo de programación competitiva, que reduzca la brecha entre las habilidades técnicas de los egresados de la Universidad Católica de Colombia y las exigencias del sector laboral tecnológico, mediante un espacio de entrenamiento práctico, evaluación continua y retroalimentación personalizada?

1.4. Funcionalidad API

API Principal (Backend en Node.js)

Esta es la API principal con la que interactúa directamente la aplicación de React. Su función es ser el centro de operaciones para todo lo que no sea el cálculo intensivo del algoritmo genético.

Sus responsabilidades clave son:

1. Gestión de Usuarios y Autenticación:

- Registro (/api/auth/register): Permite que nuevos usuarios (estudiantes, profesores) se creen una cuenta.
- Login (/api/auth/login): Autentica a los usuarios con su email y contraseña. Si es exitoso, crea un token (JWT) y lo guarda en una cookie para mantener la sesión activa.
- Logout (/api/auth/logout): Cierra la sesión del usuario eliminando la cookie de autenticación.
- Gestión de Perfiles (/api/users): Permite a los usuarios ver su propio perfil y a los administradores ver, modificar y eliminar cualquier perfil.

2. Gestión de Maratones y Problemas (Manual):

- CRUD de Maratones (/api/maratones): Permite a los profesores y administradores crear, ver, actualizar y eliminar maratones de forma manual.
- CRUD de Problemas (/api/problemas): Permite a los profesores y administradores añadir nuevos problemas de programación a la base de datos, especificando su nombre, dificultad y tiempo promedio.
- Inscripción (/api/maratones/:id/inscribir): Permite que los estudiantes se inscriban en una maratón específica.

3. Actuar como Intermediario (Gateway):

Su función más importante en relación con la IA es no ejecutar el algoritmo directamente. En cambio, cuando el usuario quiere generar una maratón optimizada, esta API de Node.js recibe la solicitud y, a su vez, realiza una llamada HTTP al microservicio de C++.

Microservicio de IA (API en C++)

Este es un backend especializado e interno. No es contactado directamente por el frontend de React. Su única misión es realizar la tarea computacionalmente pesada de forma rápida y eficiente.

Sus responsabilidades son:

1. Generación Optimizada de Maratones (POST /generate):

Este es su endpoint principal. Recibe una solicitud de la API de Node.js con parámetros como la dificultad deseada y la cantidad de problemas.

Consulta la base de datos de MongoDB para obtener la lista completa de problemas disponibles.

Ejecuta el algoritmo genético en C++ para seleccionar la combinación óptima de problemas que cumpla con los criterios (por ejemplo, minimizar el tiempo total de resolución).

Una vez que encuentra la mejor combinación, crea un nuevo documento de "maratón" y lo guarda directamente en la colección Maratones de MongoDB.

Finalmente, responde a la API de Node.js con el ID de la nueva maratón que acaba de crear.

2. Gestión de la Configuración del Algoritmo (GET /config, PUT /config):

Expone endpoints que permiten ver o modificar los parámetros internos del algoritmo genético (como el tamaño de la población o la tasa de mutación) sin necesidad de recompilar el código C++.

1.5. Objetivos

1.5.1. Objetivo general:

Desarrollar un sistema web integrado de seguimiento académico y gestión de competencias de programación, con tecnologías escalables, para fortalecer las habilidades técnicas estudiantiles en un entorno educativo colaborativo-competitivo.

1.5.2. Objetivos específicos:

- Garantizar una arquitectura escalable que sustente la integración del seguimiento académico y gestión de competencias.
- Gestionar operaciones de usuarios, competencias y maratones mediante APIs seguras y base de datos en la nube.
- Potenciar la interacción estudiantil con visualización de progreso y retroalimentación en tiempo real.
- Personalizar retos de programación mediante algoritmos de asignación óptima adaptados a habilidades individuales.

2. Tecnologías Utilizadas

El sistema sigue una arquitectura de tres capas basada en el stack MERN (MongoDB, Express.js, React.js, Node.js), diseñada para una clara separación de responsabilidades y escalabilidad. Esta estructura consta de:

- Capa de Presentación: Interfaz de usuario desarrollada con React.js
- Capa de Lógica de Negocio: API RESTful implementada con Node.js y Express.js
- Capa de Persistencia de Datos: Almacenamiento gestionado mediante MongoDB Atlas

La coherencia tecnológica se mantiene mediante el uso de JavaScript/JSON en todas las capas, optimizando el desarrollo y la integración entre componentes.

2.1. Pila MERN

La implementación de la pila MERN (MongoDB, Express.js, React y Node.js) se posiciona como la solución óptima para desarrollar un sistema de competencias de maratones de programación, gracias a su arquitectura unificada, escalabilidad y eficiencia en el manejo de datos en tiempo real.

2.1.1. Mongo DB.

La persistencia de datos se implementa mediante un cluster MongoDB Atlas con configuración gratuita M0 que incluye:

- **Replica Set de 3 Nodos:** Un nodo primario y dos secundarios para alta disponibilidad
- **Almacenamiento de 512MB:** Suficiente para la etapa de desarrollo y prototipado
- **Colecciones Principales:**
 - o usuarios: Almacena perfiles con campos diferenciados por rol (estudiante/profesor)
 - o maratones: Gestiona eventos de programación con referencias a problemas y participantes
 - o problemas: Contiene desafíos técnicos con casos de prueba embebidos

Las relaciones se manejan mediante referencias de ObjectId, con índices optimizados en campos de consulta frecuente como email de usuarios y estado de maratones. Las operaciones complejas utilizan agregaciones MongoDB con etapas como \$lookup para unir datos entre colecciones (Chodorow, 2013).

2.1.2. Express.

Express.js, como framework de backend, agiliza la creación de APIs RESTful para gestionar operaciones críticas como la validación de código, autenticación de usuarios y distribución de problemas (Express.js, s. f.).

2.1.3. React.

El frontend consiste en una aplicación React.js de página única (SPA) que gestiona toda la interacción con el usuario. Su estructura organizativa sigue un enfoque modular con:

- **Componentes Reutilizables:** Botones, formularios y elementos UI comunes
- **Vistas Específicas por Rol:**
 - o Dashboard de Estudiantes: Muestra maratones disponibles, progreso y problemas Asignados.
 - o Dashboard de Profesores: Herramientas para crear maratones, gestionar problemas y revisar soluciones.
 - o Editor de Código: Entorno integrado para resolver problemas con soporte múltiple de lenguajes.

La gestión de estado se realiza mediante Context API para compartir datos globales como la sesión de usuario y el estado de las maratones. React Router maneja la navegación entre vistas, mientras que Axios se encarga de la comunicación con el backend mediante solicitudes HTTP a los endpoints de la API.

2.1.4. Node.

Node.js, con su arquitectura asíncrona y orientada a eventos, maneja eficientemente múltiples solicitudes simultáneas.

El backend implementa una API RESTful con las siguientes características estructurales:

- Sistema de Rutas Organizado:
 - o Rutas de autenticación (/api/auth)
 - o Rutas de gestión de maratones (/api/marathons)
 - o Rutas de operaciones con problemas (/api/problems)
- Controladores de Lógica de Negocio:
 - o AuthController: Gestión de registro, login y renovación de tokens
 - o MarathonController: Creación de maratones, inscripción de estudiantes y gestión de participantes

- o ProblemController: CRUD de problemas y validación de soluciones

- Middleware Especializado:

- o Autenticación JWT: Verificación de tokens en cada solicitud protegida

- o Control de Roles: Restricción de acceso (ej: solo profesores pueden crear maratones)

- o Validación de Datos: Sanitización de entradas para prevenir ataques de inyección

Los modelos de datos se definen mediante Mongoose ODM, estableciendo esquemas, validaciones y relaciones para las entidades principales: Usuarios, Maratones y Problemas.

2.2. JWT

JWT es un componente esencial en sistemas de maratones de programación con MERN, ofreciendo un equilibrio óptimo entre seguridad, rendimiento y escalabilidad. Su integración con el ecosistema JavaScript (desde React hasta Node.js) simplifica la gestión de identidades en entornos de alta demanda, donde la agilidad y confiabilidad son críticas.

2.3. API

API Principal (Node.js)

- **Función:** Intermediario entre el frontend (React) y servicios internos.
- **Responsabilidades:**
 - o Gestión de usuarios (registro, login/logout con JWT en cookies HTTP-only, perfiles).
 - o CRUD manual de maratones y problemas (profesores/administradores).
 - o Inscripción de estudiantes en maratones.
 - o Recibir solicitudes de maratones optimizadas y redirigirlas al microservicio de C++.

Microservicio de IA (C++)

- **Función:** Ejecutar cómputo intensivo (algoritmo genético).
- **Responsabilidades:**
 - o Generar maratones optimizadas: Recibe parámetros (dificultad, n° problemas), consulta MongoDB, ejecuta algoritmo, guarda resultado en BD y devuelve ID a Node.js.
 - o Permitir ajuste de parámetros del algoritmo (sin recompilar código).

Flujo Clave

1. Frontend → Node.js: Solicita maratón optimizada.
2. Node.js → Microservicio C++: Redirige parámetros.
3. C++: Ejecuta algoritmo, guarda maratón en MongoDB, devuelve ID.
4. Node.js → Frontend: Envía datos de la maratón generada.

Aspectos Destacados

- **Separación de roles:** Node.js gestiona interacciones; C++ enfocado en cálculos pesados.
- **Seguridad:** JWT en cookies + microservicio C++ interno (no expuesto al frontend).
- **Eficiencia:** C++ para procesos intensivos; Node.js para operaciones ligeras/concurrentes.
- **BD unificada:** MongoDB como única fuente de datos para ambas APIs.

2.4. Control De Versiones

El control de cambios se realiza para llevar un seguimiento de lo que se ha realizado y lo que se debe realizar, este control de cambios lo llevamos mediante GitHub.

Ilustración 1 Roadmap_1

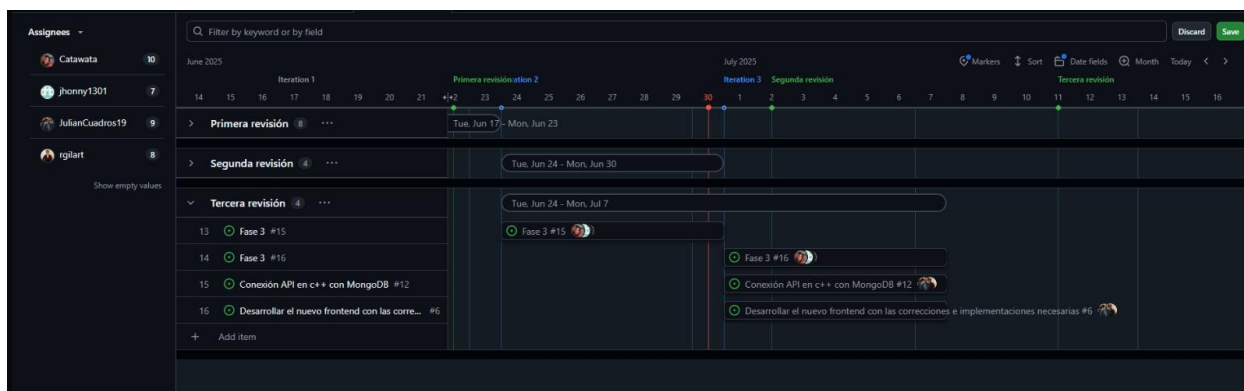


Ilustración 2 Roadmap_2

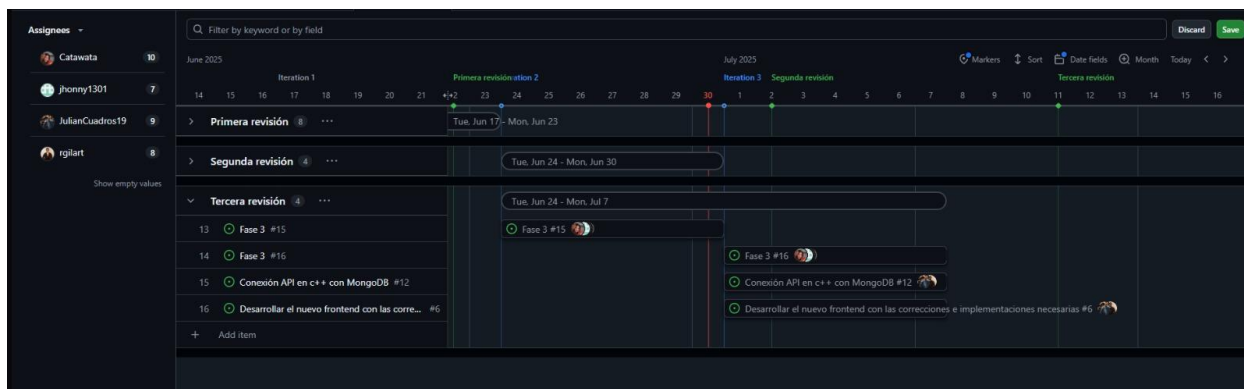


Ilustración 3 Backlog iteración 1

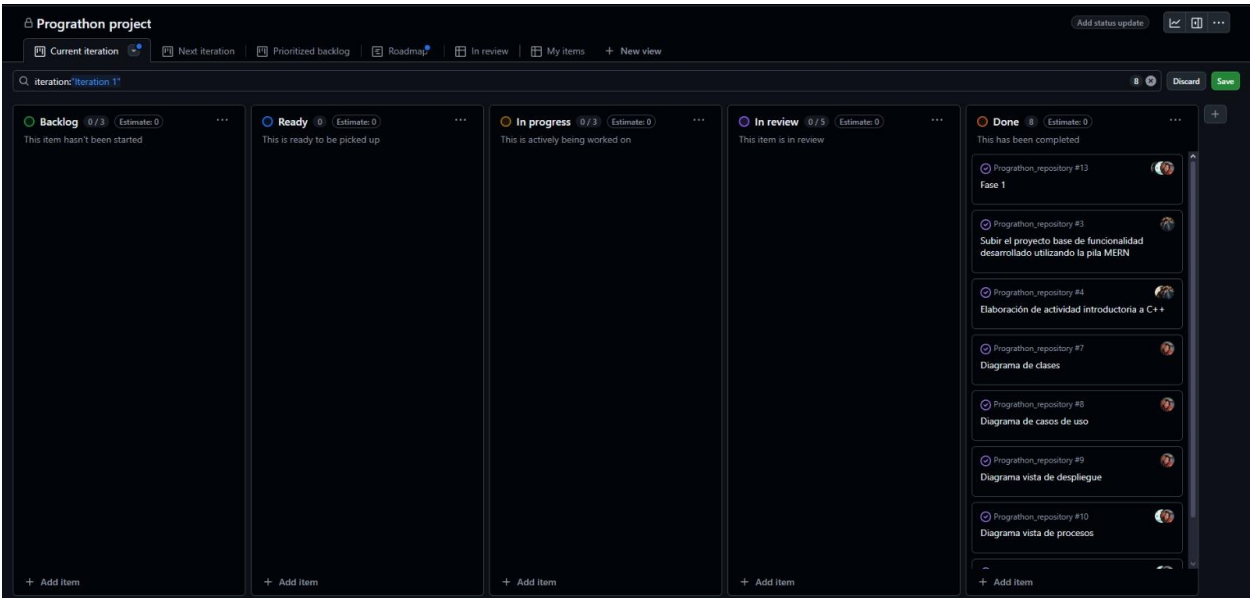


Ilustración 4 Backlog interacción 2

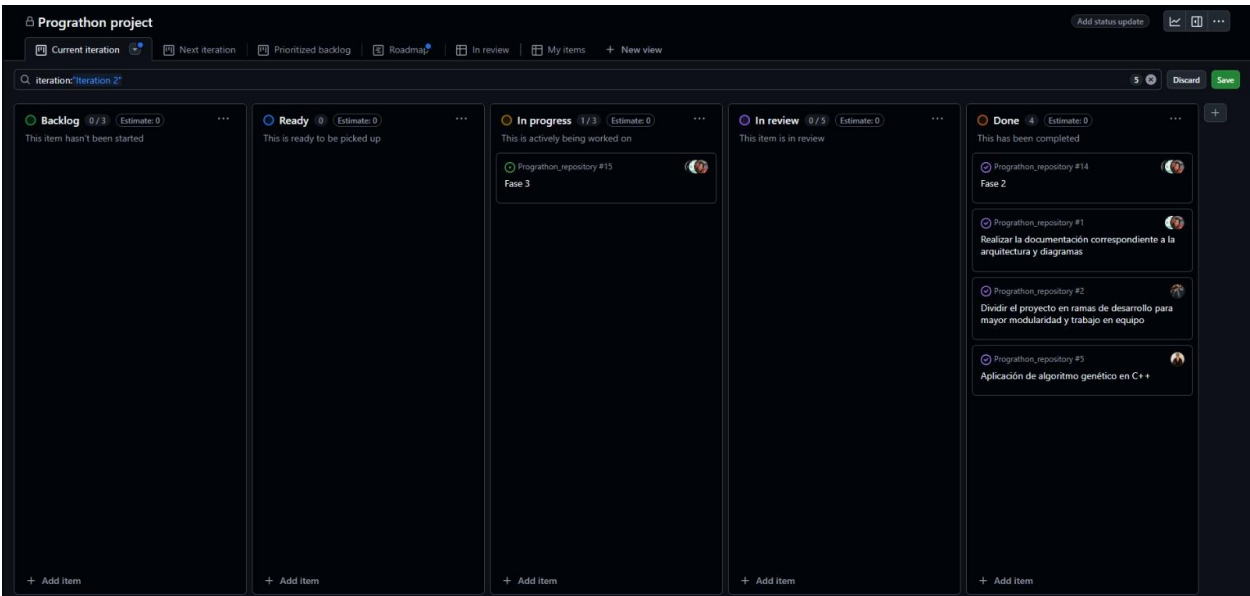


Ilustración 5 Backlog interacción 3

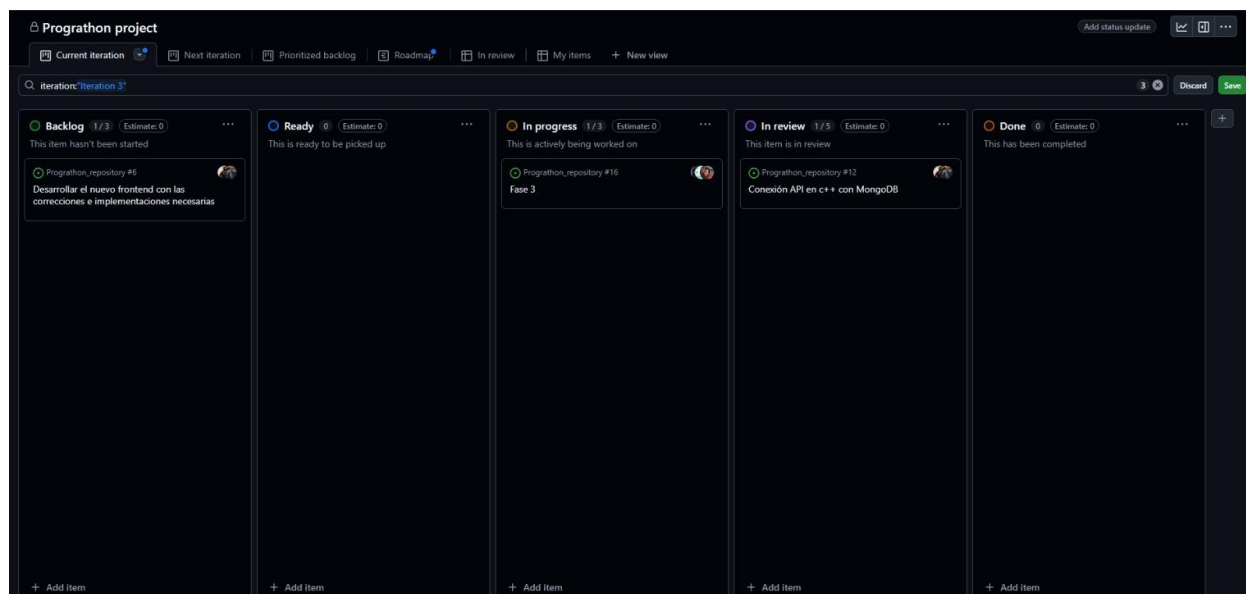
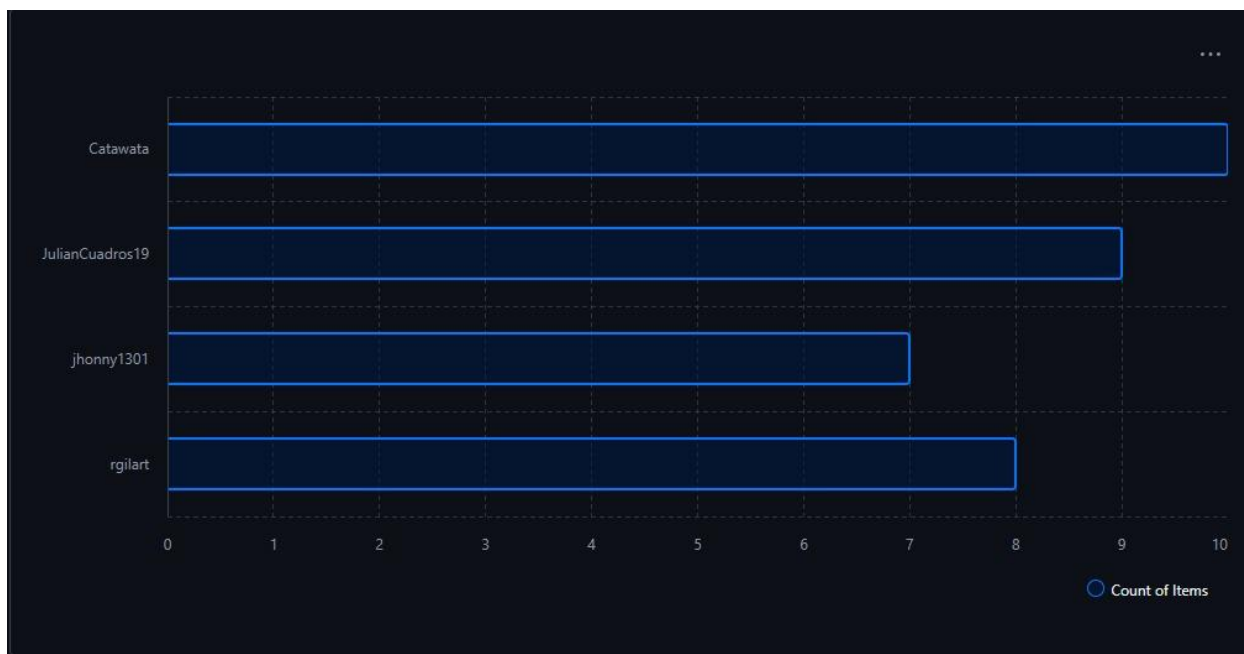


Ilustración 6 Burn up proyecto



Ilustración 7 Assignees issues



3. Metodologías

3.1. Explicación de la metodología

El Proceso Unificado de Rational (RUP) es la metodología ideal para desarrollar un sistema de maraton de programación usando la pila MERN, debido a su enfoque iterativo, gestión de riesgos proactiva y adaptabilidad a proyectos complejos.

RUP enfatiza la definición temprana de la visión y requisitos mediante casos de uso. Esto es crucial en un sistema educativo, donde deben priorizarse necesidades como la integración con currículas académicas y roles de usuarios (estudiantes, profesores).

RUP es la mejor opción porque:

- Maneja complejidad técnica mediante arquitectura temprana (vital para MERN + JWT).
- Alinea iteraciones con valor educativo: Prioriza el dashboard de progreso y retroalimentación en tiempo real.
- Mitiga riesgos proactivamente: Prototipos en Fase 2 evitan fallos en WebSockets o algoritmos.
- Facilita adopción en entornos educativos: Despliegue gradual con beta controlado.

3.2. Fases

Fase 1:

Objetivo: Alinear el sistema con el contexto educativo y definir el alcance integrado.

- Definir la visión del producto.
- Identificar los requisitos clave.
- Análisis de riesgo inicial.

Fase 2:

Objetivo: Establecer la arquitectura escalable y validar la integración.

- Diseñar arquitectura base.
- Prototipar núcleos críticos.
- Pruebas de usabilidad.

Fase 3:

Objetivo: Desarrollo iterativo con entregas funcionales incrementales.

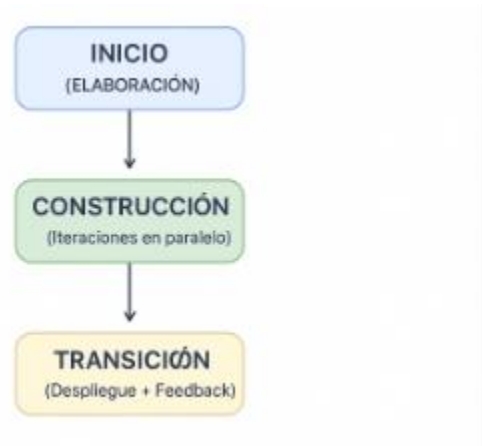
- Iteración 1: Backend (APIs) + Algoritmo
 - API de usuarios/competencias
 - Algoritmo base de asignación de problemas
- Iteración 2: Frontend (Progreso académico)
 - Dashboard de estudiante
 - Gráficas de avance
- Iteración 3: Competencias + Retroalimentación
 - Sistema de maratones
 - Websockets para feedback en tiempo real
- Iteración 4: Integración total + Optimización
 - Personalización completa de retos
 - Pruebas de carga

Fase 4:

Objetivo: Despliegue gradual y adopción en entorno educativo.

- Beta controlado.
- Retroalimentación y ajustes.
- Entrega final.

Ilustración 8 Fases del proyecto



3.3. Asignación De Roles

Ilustración 9 Roles del proyecto

Rol	Participante	Responsabilidades	Tecnologías	Entregables	Fases de participación
Arquitecta y líder técnica	Catalina	- Diseñar la arquitectura del sistema multicapa (frontend, backend, base de datos) - Definir y validar contratos de APIs - Definir y buenas prácticas - Coordinar revisiones técnicas y refactoros	Node.js, NestJS, PostgreSQL, Redis, GitHub.	- Diagrama de arquitectura - Especificaciones de APIs - Documentación técnica (API + Infraestructura)	Fase 1 (planificación) Fase 2 (arquitectura) Fase 3 (supervisión técnica)
Frontend Developer	Jhonatan	- Desarrollar interfaces reactivas y accesibles - Implementar flujo de navegación, formularios, validaciones y manejo de estado - Conectar interfaces con backend vía - Asegurar el cumplimiento de principios de usabilidad y accesibilidad	React.js / Next.js, TailwindCSS.	interfaces funcionales y Pruebas unitarias frontend	Fase 3 - Iteraciones 2 y 3
Backend Developer	Rodolfo	- Gestionar roles y permisos (Auth y RBAC) - Implementar lógica de negocio para maratones, competencias, recompensas y rankings - Modelar y normalizar la base de datos relacional - Implementar pruebas unitarias y de integración	Node.js/NestJS, PostgreSQL, Redis, JWT.	- APIs documentadas y seguras - Scripts de base de datos - Pruebas automatizadas backend	Fase 3 - Iteraciones 1 y 4
DevOps Engineer, QA Funcional	Julian	Configurar canal CI/CD (build, test, deploy) en GitHub Actions - Automatizar Pruebas funcionales y de rendimiento - Implementar monitoreo básico (logs, métricas) - Elaborar Documentación de despliegue para el equipo	GitHub Actions	Pipeline CI/CD funcional - Reportes de Pruebas funcionales	Fase 2 (Infraestructura) Fase 3 (QA y Deploy)

3.4. Requerimientos

3.4.1. Requerimientos Funcionales.

Tabla 2 RF001

Identificación del requerimiento	RF001
Nombre:	Crear Maratón.
Entradas:	Datos del maratón (nombre, descripción, fecha, reglas).
Proceso:	Consulta de maratones según permisos de usuario.
Salidas:	Confirmación de creación.
Descripción:	Los profesores deben poder crear nuevos maratones.
Prioridad:	Alta.
Categoría:	Funcionalidad Principal del Negocio.

Tabla 3 RF002

Identificación del requerimiento	RF002
Nombre:	Registro y Autenticación
Entradas:	Datos del usuario: nombre, apellido, edad, correo, contraseña, rol
Proceso:	El usuario se registra ingresando sus datos personales y rol. El sistema valida la información, encripta la contraseña y guarda el registro. Para iniciar sesión, se verifica la contraseña y se genera un token que permite el acceso seguro según el rol del usuario.
Salidas:	Usuario registrado exitosamente, acceso habilitado para iniciar sesión.
Descripción:	Los usuarios deben poder acceder de manera acertada a su inicio de sesión, dependiendo del rol asignado.
Prioridad:	Alta.
Categoría:	Registro y autenticación.

Tabla 4 RF003

Identificación del requerimiento	RF003
Nombre:	Gestión De Competencias.
Entradas:	Correo, contraseña.
Proceso:	El profesor autenticado accede a una interfaz donde puede crear, editar o eliminar

	competencias académicas. Estas acciones se reflejan directamente en la base de datos y quedan disponibles para los estudiantes.
Salidas:	Acceso permitido dependiendo su rol.
Descripción:	De acuerdo a las credenciales de acceso se permiten a los usuarios asignados las funciones del CRUD en el modulo de maratones.
Prioridad:	Alta.
Categoría:	Gestión de Maratones

Tabla 5 RF004

Identificación del requerimiento	RF004
Nombre:	Registro de avances en competencia.
Entradas:	Datos de la competencia: nombre, descripción, criterios, nivel de dificultad.
Proceso:	El estudiante visualiza las competencias disponibles y registra sus avances en cada una. El sistema guarda esta información y la muestra en su panel de control como parte de su progreso académico.
Salidas:	Confirmación de actualización.
Descripción:	Actualización de los avances en el perfil del estudiante.
Prioridad:	Media.
Categoría:	Gestión de Maratones.

Tabla 6 RF005

Identificación del requerimiento	RF005
Nombre:	Sistema de recompensas.
Entradas:	ID de la competencia, porcentaje o nivel alcanzado, comentario opcional.
Proceso:	El profesor configura una maratón asignando una competencia específica, su nivel o porcentaje requerido, y los retos correspondientes. Los estudiantes, al participar, resuelven los problemas planteados. El sistema evalúa automáticamente sus avances y actualiza el progreso en el perfil de cada estudiante en tiempo real.

Salidas:	Actualización del progreso reflejado en el perfil del estudiante.
Descripción:	Los profesores podrán asignar un premio a los estudiantes que muestren un buen desempeño en la competencia que el mismo haya creado.
Prioridad:	Media.
Categoría:	Gestión de Maratones.

Tabla 7 RF006

Identificación del requerimiento	RF006
Nombre:	Ver Maratones.
Entradas:	Maraton creada previamente.
Proceso:	Consulta de maratones según permisos de usuario.
Salidas:	Lista de maratones.
Descripción:	Los usuarios deben poder visualizar maratones disponibles.
Prioridad:	Alta.
Categoría:	Funcionalidad Principal del Negocio.

Tabla 8 RF007

Identificación del requerimiento	RF007
Nombre:	Entrar a Maratones.
Entradas:	Selección de maratón.
Proceso:	Validación de elegibilidad y registro.
Salidas:	Confirmación de registro o mensaje de error.
Descripción:	Los estudiantes deben poder registrarse/participar en maratones.
Prioridad:	Alta.
Categoría:	Funcionalidad Principal del Negocio.

Tabla 9 RF008

Identificación del requerimiento	RF008
Nombre:	Editar Configuración del Sistema.
Entradas:	Parámetros de configuración.
Proceso:	Validación y aplicación de cambios.
Salidas:	Confirmación de cambios.
Descripción:	Los administradores deben poder modificar configuraciones generales .
Prioridad:	Media.

Categoría:	Administración del Sistema.
------------	-----------------------------

Tabla 10 RF009

Identificación del requerimiento	RF009
Nombre:	Gamificación y Rankings.
Entradas:	Detección de avance significativo o competencia completada.
Proceso:	El sistema detecta avances relevantes o competencias completadas por los estudiantes. En respuesta, desbloquea logros, insignias o envía alertas de reconocimiento que se muestran en su perfil. Además, se actualiza el ranking general, incentivando la participación mediante elementos de gamificación.
Salidas:	Desbloqueo de logros, alertas o insignias en el perfil del estudiante.
Descripción:	Los estudiantes al lograr completar una maratón o lograr un avance realmente significativo, aumentará en el nivel de la maratón en comparación a los demás competidores, y se le reconocerá su esfuerzo mediante algún tipo de incentivo visual.
Prioridad:	Baja.
Categoría:	Gestión de Maratones.

3.4.1. Requerimientos No Funcionales.

3.4.1.1 Rendimiento:

Tabla 11 RNF001

Requerimiento	Descripción	Característica ISO 25010	Sub - Característica
PERF – 001 (respuesta <1 min)	El sistema debe responder a las solicitudes de usuario en menos de 1 minuto bajo condiciones normales	Eficiencia de mantenimiento	Comportamiento temporal
PERF – 002 (100 usuarios concurrentes)	Debe soportar hasta 100 usuarios concurrentes	Eficiencia de mantenimiento	Utilización de recursos
PERF – 003 (80% disponibilidad)	La disponibilidad del sistema debe ser del	Eficiencia de mantenimiento	Capacidad

	80% durante horario hábil		
--	---------------------------	--	--

Justificación:

- PERF-001/PERF-002 miden eficiencia temporal y uso de recursos bajo carga.

3.4.1.2 Mantenibilidad:

Tabla 12 RNF002

Requerimiento	Descripción	Característica ISO 25010	Sub - característica
MANT - 001 (código documento)	El código debe estar documentado	Mantenibilidad	Modularidad
MANT – 002 (Actualizaciones sin interrupción)	Debe permitir actualizaciones sin interrumpir el servicio	Mantenibilidad	Reusabilidad
MANT – 003 (Logs de error detallados)	Debe generar logs de error detallados.	Mantenibilidad	Analizabilidad

Justificación:

- Documentación y logs mejoran analizabilidad (detección de fallos).
- Actualizaciones sin downtime requieren modularidad (componentes independientes).

3.4.1.3 Usabilidad:

Tabla 13 RNF003

Requerimiento	Descripción	Característica ISO 25010	Sub - característica
USA - 001	Usuarios sin formación técnica completarán tareas clave en ≤ 3 intentos	Usabilidad	Aprendibilidad
USA – 002	Debe proporcionar mensajes de error claros	Usabilidad	Tolerancia a errores
USA – 003	100% de las funciones críticas tendrán ayuda contextual visible	Usabilidad	Operabilidad

Justificación:

- Enfoque en usuarios no técnicos → aprendibilidad y operabilidad.
- Mensajes claros son parte de tolerancia a errores (ISO 25010).

3.4.1.4 Fiabilidad:

Tabla 14 RNF004

Requerimiento	Descripción	Característica ISO 25010	Sub - característica
FIAB-001	El sistema debe mantener operativas las funcionalidades esenciales durante fallos en componentes no críticos.	Fiabilidad	Tolerancia a fallos
FIAB-002	Garantizar integridad transaccional incluso tras interrupciones abruptas	Fiabilidad	Recuperabilidad & Madurez
FIAB-003	Simulación mensual de escenarios catastróficos para validar mecanismos de recuperación.	Fiabilidad	Madurez

Justificación:

- Aislamiento de servicios críticos (Bulkheads) → Mantiene funcionalidad esencial durante fallos.
- Circuit breakers → Evita propagación de errores.
- Transacciones Sagas → Garantizan integridad eventual.
- Mecanismos de compensación → Recuperación proactiva de transacciones.
- Simulacros con Chaos Engineering → Eliminan "puntos ciegos" de resiliencia.
- Validación de RTO/RPO → Mejora continua del diseño.

3.4.1.5 Seguridad:

Tabla 15 RNF005

Requerimiento	Descripción	Característica ISO 25010	Sub - característica
SEG-001	Garantizar que solo profesores autorizados puedan crear/modificar maratones. Evita que estudiantes hackeen el sistema para	Seguridad	Autenticidad

	cambiar reglas o ver soluciones ajenas.		
SEG-002	Proteger soluciones de estudiantes y datos personales. Aunque hackers roben la base de datos, no podrán leer la información.	Seguridad	Confidencialidad
SEG-003	Saber quién, cuándo y qué hizo cada profesor. Permite auditar malas prácticas.	Seguridad	Integridad

Justificación:

- Cifrado garantiza confidencialidad.
- Autenticación y logs aseguran autenticidad y no repudio.

3.4.1.6 Compatibilidad:

Tabla 16 RNF006

Requerimiento	Descripción	Característica ISO 25010	Sub - característica
COMP-001	Permitir que cada parte del sistema funcione de forma independiente. Así, si falla un componente, los demás siguen operando.	Compatibilidad	Compatibilidad
COMP-002	Actualizaciones sin Interrupción: Agregar mejoras mientras los estudiantes usan activamente el sistema.	Compatibilidad	Compatibilidad

Justificación:

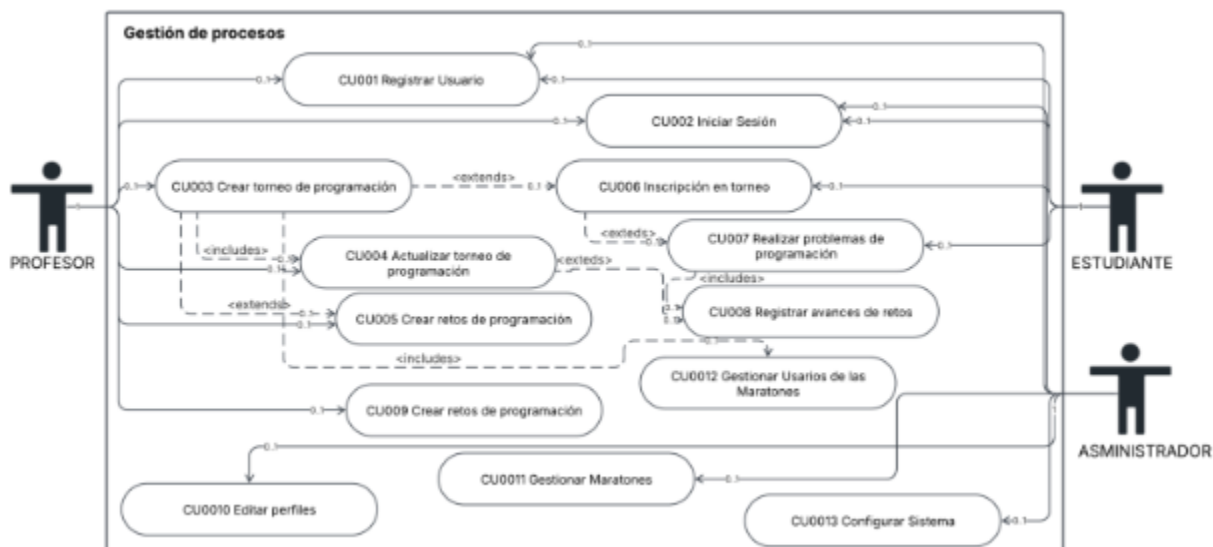
- Modularidad permite coexistencia con futuros componentes (interoperabilidad implícita).
- Actualizaciones sin downtime → compatibilidad con versiones anteriores.

3.5. Arquitectura

Para la realización de este proyecto se manejará una arquitectura 4+1.

3.5.1. Casos De Uso.

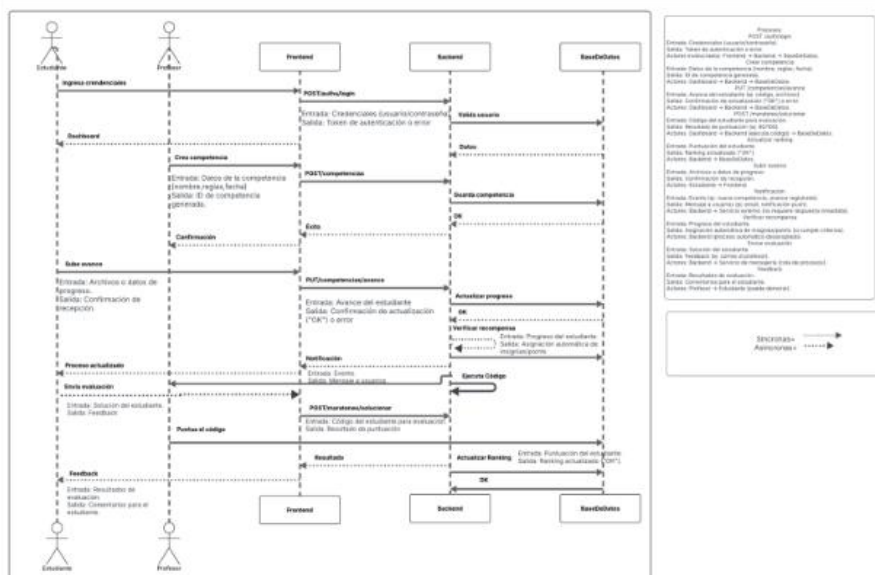
Ilustración 10 Casos de uso



Nota: Muestra la interacción de los usuarios con el sistema, detallando las funcionalidades clave y relaciones entre casos de uso.

3.5.2. Diagrama De Vista De Procesos.

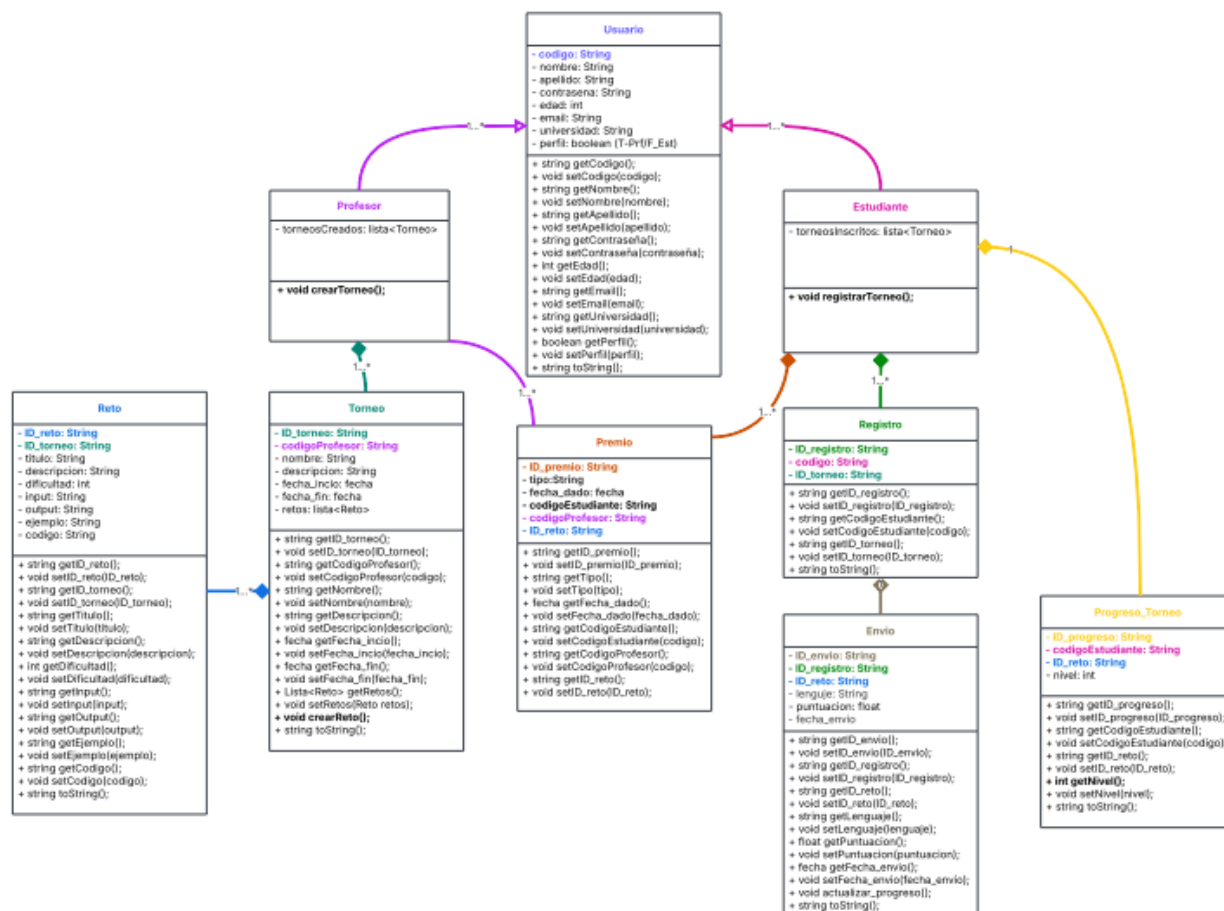
Ilustración 11 Vista de procesos



Nota: Representa el flujo de datos entre frontend, backend y la base de datos, destacando la ejecución de tareas clave como autenticación y evaluación de código.

3.5.2. Diagrama De Vista Logica.

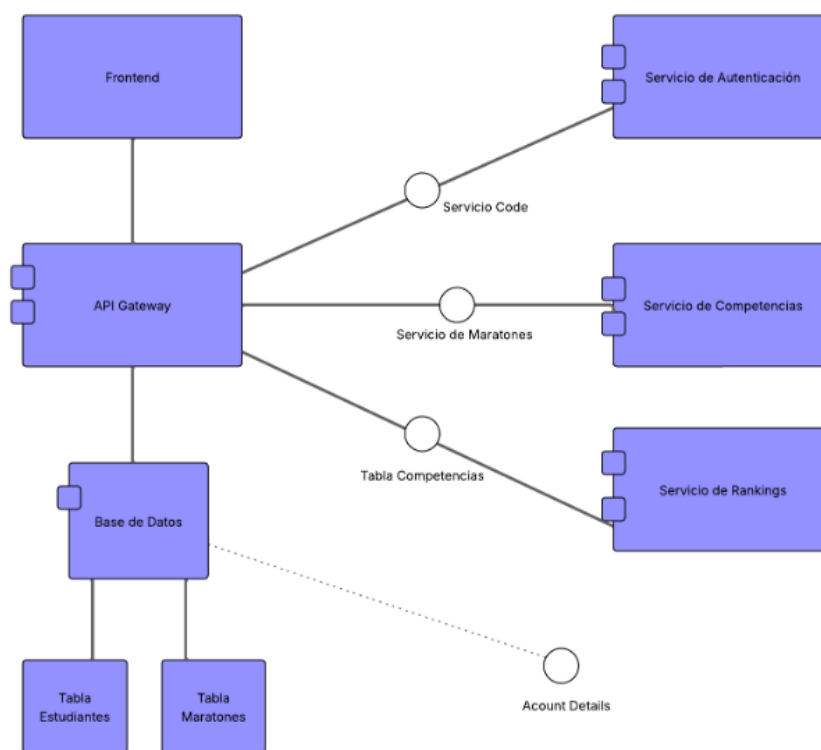
Ilustración 12 Vista lógica



Nota: Define la estructura de datos del sistema, mostrando entidades, atributos y relaciones entre usuarios, retos, torneos y competencias.

3.5.2. Diagrama De Implementación.

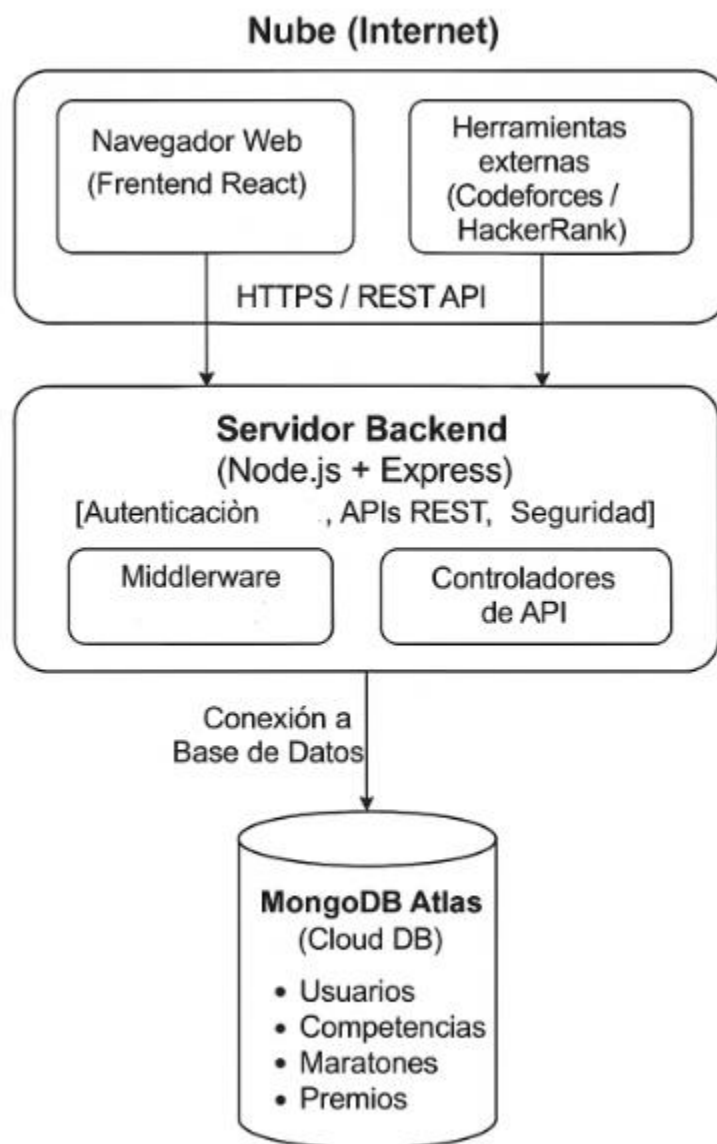
Ilustración 13 Implementación



Nota: Describe cómo se organizan los servicios dentro del API Gateway, separando módulos de autenticación, maratones y evaluación de código.

3.5.2. Diagrama De Vista De Despliegue.

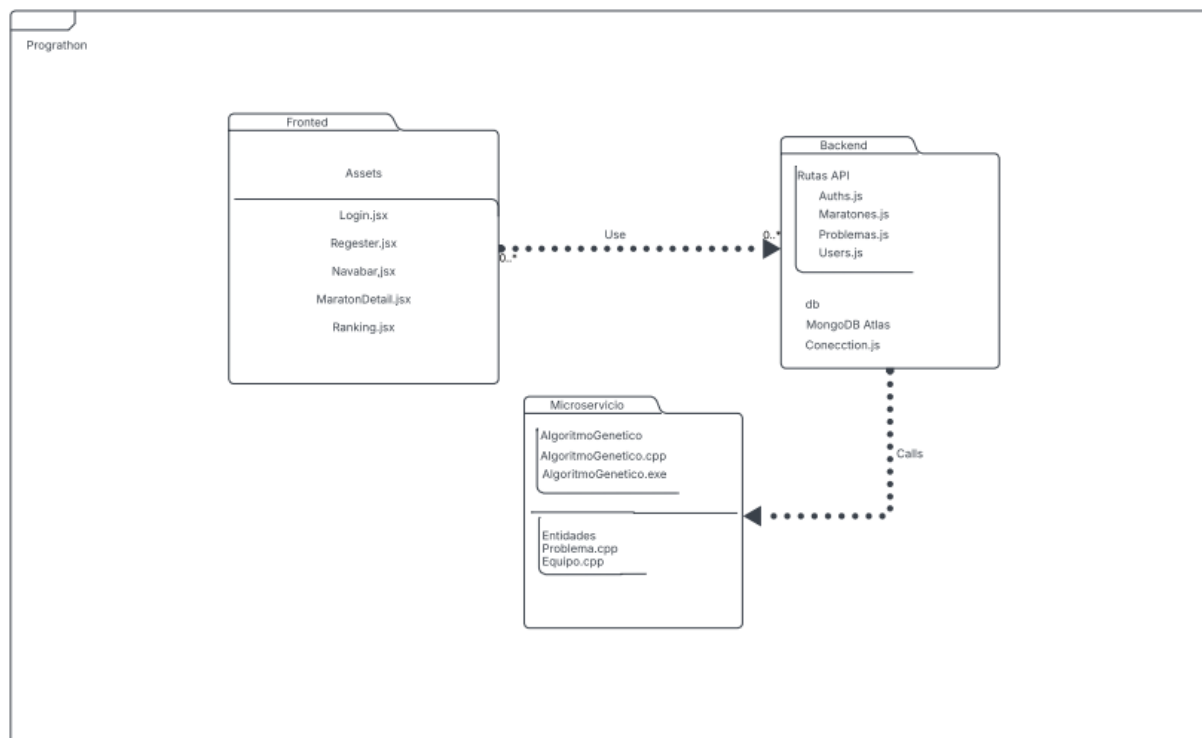
Ilustración 14 Vista de despliegue



Nota: Muestra la infraestructura del sistema en la nube, incluyendo la interacción entre frontend, backend, base de datos y servicios externos.

3.5.2. Diagrama De Paquetes.

Ilustración 15 Paquetes



Nota: Este diagrama de paquetes representa la estructura modular del sistema Prograthon, destacando las dependencias entre frontend, backend y microservicio, según los estándares de UML 2.0.

4. Errores Y Soluciones

1. Integración del microservicio en C++ con Node.js :

Problema: Dificultades al comunicar el microservicio en C++ con el backend en Node.js, debido a diferencias de lenguaje, serialización de datos o errores en el tiempo de respuesta.

Solución: Se definió un protocolo claro de intercambio mediante HTTP y formato JSON. Se empleó Axios desde Node.js para enviar solicitudes, y se manejaron los errores con un sistema de reintentos y logs detallados.

2. Autenticación y gestión de sesiones inseguras

Problema: Inicialmente los tokens JWT eran expuestos en el almacenamiento local del navegador, lo que representaba un riesgo de seguridad.

Solución: Se migró al uso de cookies HTTP-only para almacenar los tokens JWT, lo que evita su acceso por scripts maliciosos (XSS).

3. Dificultades con pruebas del algoritmo genético

Problema: La validación de los resultados del algoritmo genético en C++ era inconsistente, generando maratones no balanceadas o con tiempos estimados erróneos.

Solución: Se incorporaron pruebas automatizadas con datos controlados, y se habilitó el endpoint /config para ajustar parámetros del algoritmo sin recompilar, facilitando la calibración.

4. Complejidad en la gestión de usuarios por rol

Problema: Usuarios podían acceder a vistas o ejecutar acciones no permitidas para su rol (por ejemplo, estudiantes intentando acceder al CRUD de maratones).

Solución: Se implementó un sistema robusto de control de roles mediante middleware en Express.js que valida permisos en cada solicitud protegida.

5. Cambios frecuentes en requerimientos durante desarrollo










Problema: Las recomendaciones del profesor implicaron modificar objetivos, roles y diagramas varias veces, lo cual generaba inconsistencias y retrabajo.

Solución: Se mantuvo un control de versiones claro en el documento y el repositorio GitHub, con histórico de cambios y responsables designados en cada versión.

5. Objetivos Logrados

5.1. Requerimientos Funcionales logrados

Ilustración 16 Revisión de requerimientos cumplidos

#	Requerimineto	Estado	Observaciones	Fecha de finalización
1	RF001	Completado 	Funcionando correctamente	Wednesday, June 18, 2025
2	RF002	Completado 	Funcionando correctamente	Tuesday, June 17, 2025
3	RF003	Completado 	Funcionando correctamente	Wednesday, June 18, 2025
4	RF004	En proceso 	Se encuentra en proceso de desarrollo	
5	RF005	En proceso 	Se encuentra en proceso de desarrollo	
6	RF006	Completado 	Funcionando correctamente	Tuesday, June 17, 2025
7	RF007	Completado 	Funcionando correctamente	Monday, June 23, 2025
8	RF008	Completado 	Funcionando correctamente	Friday, June 20, 2025
9	RF009	En proceso 	Se encuentra en proceso de desarrollo	

5.2. Checklist

Ilustración 17 Checklist de calificación

#	Descripción	Estado
1	La API está desarrollada en C++ utilizando un framework como Crow, Drogon o similar.	Hecho ✓
2	La API responde correctamente a peticiones GET y POST.	Hecho ✓
3	Al menos una ruta realiza operaciones CRUD sobre una base de datos.	Por hacer ⏰
4	El proyecto contiene un archivo CMakeLists.txt funcional.	Hecho ✓
5	Hay una estructura de carpetas clara: src/, include/, build/, etc.	Hecho ✓
6	El equipo ha redactado un informe técnico explicando: objetivos, herramientas, errores y soluciones.	Hecho ✓
7	Se ha descrito la metodología aplicada (xp, rup, lean, etc.).	Hecho ✓
8	Se incluye evidencia de los objetivos alcanzados y funcionalidades implementadas.	Hecho ✓
9	Existe una interfaz mínima que consume la API.	Hecho ✓
10	El frontend puede estar desarrollado en HTML/JS, React, u otro.	Hecho ✓

6. Bibliografía

- Diego Oliveros. (2025). Desarrollo de software. Universidad Católica de Colombia
- Miguel Parada. (21 de octubre, 2020). MERN Stack: Qué es y qué ventajas ofrece.
- OpenWebinars S.L., <https://openwebinars.net/blog/mern-stack-que-es-y-queventajasofrece/>
- Beck, K. et al. (2001). Manifiesto para el desarrollo ágil de software. Agile Alliance.
- Chacon, S. & Straub, B. (2014). Pro Git (2a ed.). Apress.
- Elmasri, R. & Navathe, S. B. (2016). Fundamentals of database systems (7a ed.). Pearson.
- Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. University of California.
- Flanagan, D. (2011). JavaScript: The definitive guide (6a ed.). O'Reilly Media.
- Gamma, E. et al. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.
- Graham, D. & Fewster, M. (2012). Experiences of test automation. Addison-Wesley.
- IEEE Computer Society. (1990). IEEE standard glossary of software engineering terminology (IEEE Std 610.12-1990). IEEE.
- Knuth, D. E. (1997). The art of computer programming: Fundamental algorithms (3a ed., Vol. 1). Addison-Wesley.
- Larman, C. (2004). Agile and iterative development: A manager's guide. Addison-Wesley.

- Loeliger, J. & McCullough, M. (2012). Version control with Git (2a ed.). O'Reilly Media.
- McConnell, S. (2004). Code complete: A practical handbook of software construction (2a ed.). Microsoft Press.
- Myers, G. J. et al. (2012). The art of software testing (3a ed.). Wiley.
- Norman, D. A. (1988). The psychology of everyday things. Basic Books.
- Norman, D. A. (2013). The design of everyday things (Ed. revisada). Basic Books.
- Richardson, L. & Ruby, S. (2007). RESTful web services. O'Reilly Media.
- Schwaber, K. (2004). Agile project management with Scrum. Microsoft Press.
- Schwaber, K. & Sutherland, J. (2020). La guía de Scrum. Scrum Guides.
- Stroustrup, B. (2013). The C++ programming language (4a ed.). Addison Wesley.
- Zeller, A. (2009). Why programs fail: A guide to systematic debugging. Morgan Kaufmann.
- Gutiérrez, E., & Vargas Riaño, D. A. (2023, 30 de septiembre). Colombia, el tercer país de la región donde más aumentó el número de programadores. El Colombiano.
- <https://www.elcolombiano.com/negocios/colombia-tercer-pais-de-america-latina-dondemas-nuevos-programadores-hay-2023-FF22500063>.
- Clavijo, M. A. (2024, 2 de febrero). Mercado laboral de talento TI en Colombia. DB SYSTEM. <https://www.db-system.com/mercado-laboral-de-talento-ti-en-colombia/>
- Redacción ELHERALDO.CO. (2021, 3 de septiembre). La covid-19 agudizó la demanda de talento TI en Colombia. El Herald.
- <https://www.elheraldo.co/tecnologia/2021/09/03/la-covid-19-agudizo-la-demanda-det talento-ti-en-colombia/>
- MongoDB Inc. (2024). MongoDB Documentation. <https://www.mongodb.com/docs>
- Express.js. (2024). Express.js API documentation. <https://expressjs.com/en/api.html>