Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Planeación de sistemas de software

Gpo 105

Arquitectura y Estándares

Fernando Morán Fougerat A01284623

Imanol Armando González Solís A00835759

Ramiro Alejandro Garza Villarreal A01178167

Rogelio Garza Rendón A01571384

Campus Monterrey

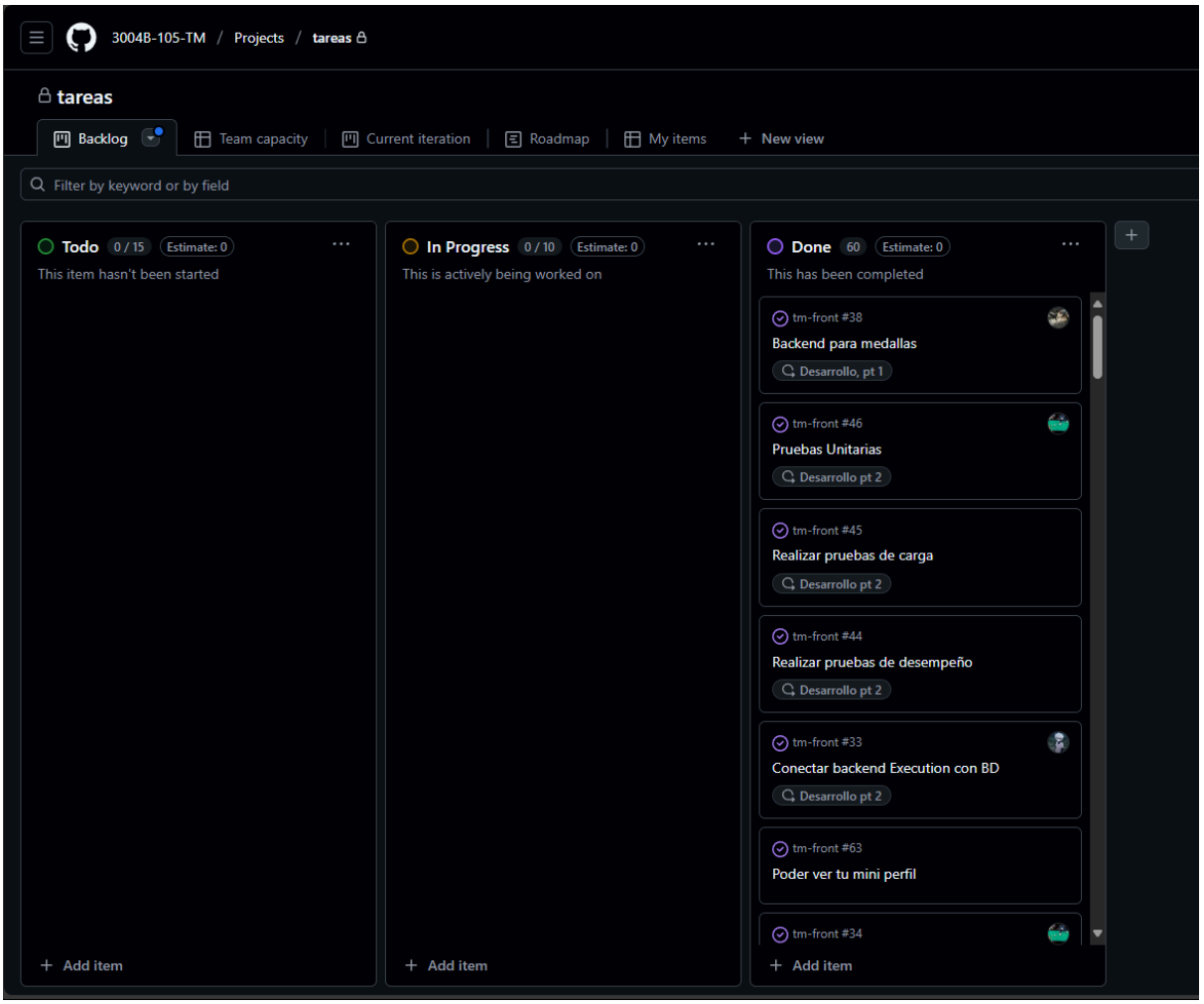
2 jun 2025

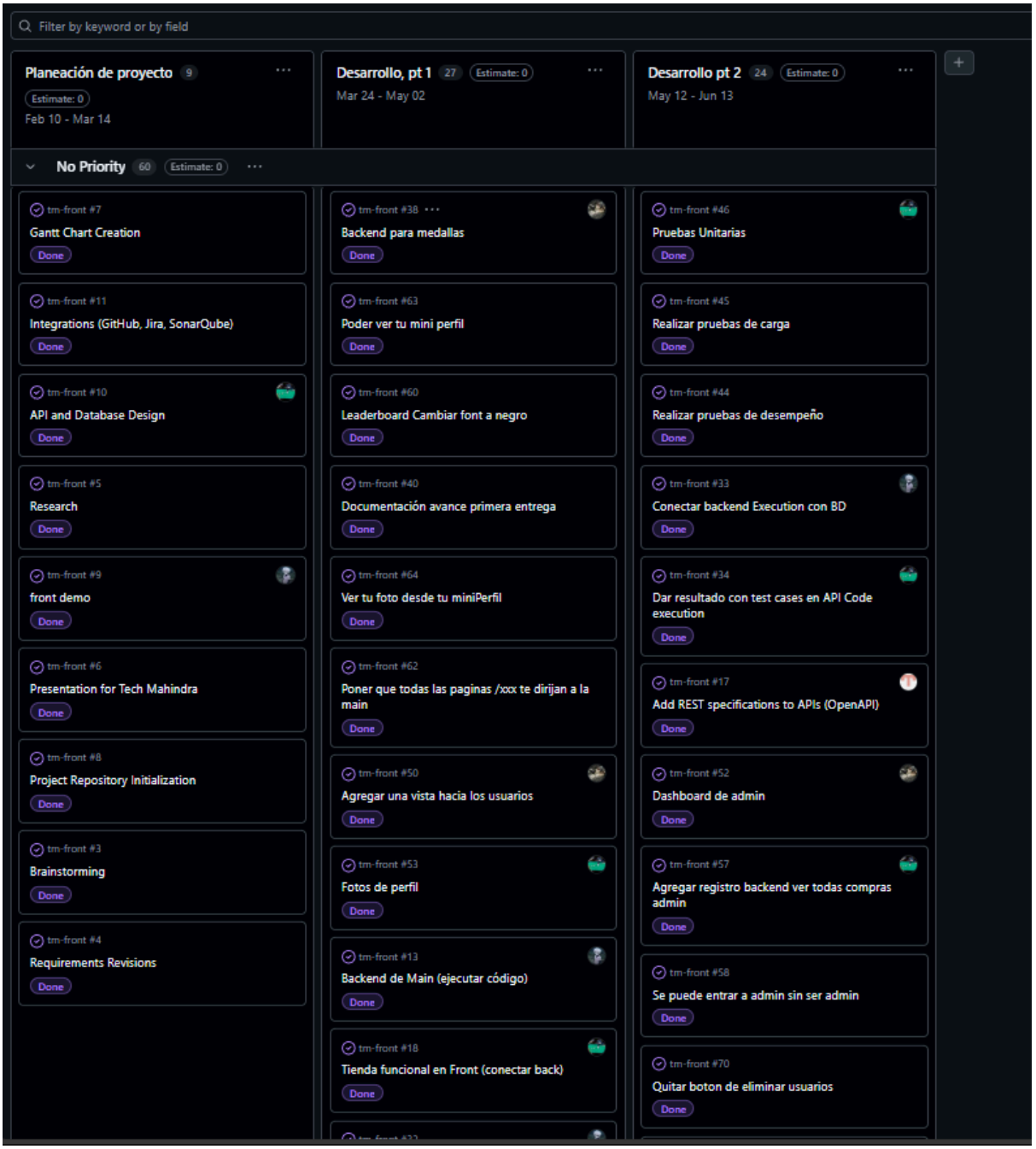
Actualización al software de administración utilizada y el Excel (con formato solicitado). 3
Actualizar el Product Backlog 5
Plan de proyecto 7
Product Backlog (versión final) 8
Sprint´s Backlogs (de los 3 Sprints) 10
Minutas de Planeación de los 3 Sprints (3 en total) 12
Diagramas Burn Down Chart de los 3 Sprints (3 diagramas en total) 14

Actualización al software de administración utilizada y el Excel (con formato solicitado).

Para la gestión y administración del proyecto Code Courses, utilizamos GitHub Projects como herramienta principal de planificación. A través de esta herramienta, organizamos nuestras tareas por historias de usuario, asignamos responsables, controlamos el avance por sprint.

- **Tableros de trabajo**: Se implementaron tableros tipo *Kanban* en GitHub Projects para dividir las tareas por columna (To Do, In Progress, Review, Done) y organizarlas por sprint.





También usamos un excel para llevar a cabo 2 seguimientos, el primero era un cronograma en donde al final se implementó en Github, mientras que usamos otro excel para hacer un registro de nuestras reuniones diarias en donde seguimos el formato de “Qué hice ayer, qué haré hoy y mis impedimentos.”

[Roadmap · tareas]

Daily standup meetings

Actualizar el Product Backlog

Durante el desarrollo de Code Courses, el Product Backlog fue actualizado continuamente para reflejar el estado real del proyecto, considerando las funcionalidades priorizadas, su nivel de complejidad y el avance logrado en cada sprint.

El backlog se construyó en base a historias de usuario, cada una con la siguiente información:

- **ID de historia**
- **Descripción funcional**
- **Prioridad (XS, S, M, L, XL)**
- **Estado actual** (Pendiente, En progreso, Completado)
- **Sprint en el que se trabajó**

Usamos Github para manejar el backlog Team capacity · tareas

Durante los 2 sprints y la planeación:

- Se agregaron nuevas historias a partir del feedback del equipo y el análisis funcional.
- Se ajustaron prioridades conforme evolucionó el alcance del proyecto.

Actualizar el software de administración utilizado

El seguimiento del proyecto Code Courses se gestionó en su totalidad desde GitHub Projects. A lo largo del desarrollo, el software se mantuvo actualizado en tiempo real, permitiendo al equipo tener visibilidad clara sobre las tareas activas, los avances por sprint y las entregas completadas.

A lo largo del proyecto fuimos agregando tareas necesarias para completar las historias de usuario, en donde estas tareas eran etiquetadas según su prioridad, sprint y quien las debe hacer, y Github nos ayudó creando varios diagramas y tablas que nos facilitaron la planeación.

Plan de proyecto

Desarrollamos una aplicación web donde los usuarios puedan resolver retos de programación, obtener recompensas, ver su progreso y competir en un sistema de leaderboard, utilizando una arquitectura de microservicios, base de datos relacional y flujos automatizados de CI/CD para pruebas.

Roles del equipo

- **Rogelio** – Desarrollador y Product Owner
- **Ramiro** – Desarrollador
- **Fernando** – Desarrollador.
- **Imanol** – Desarrollador y Scrum Master.

Entregables por Sprint

- **Sprint 1**:
    - Configuración de entorno de desarrollo.
    - Estructura del backend en Go.
    - Modelado inicial de base de datos.
    - Primeros endpoints funcionales.

- **Sprint 2**:
    - Implementación del frontend en React.
    - Integración entre microservicios.
    - Desarrollo de funcionalidades principales (retos, envíos, leaderboard).
    - Pruebas funcionales y documentación.
    - Preparación del despliegue.

- **Gestión de tareas**: GitHub Projects + GitHub Actions
- **Diseño**: Figma
- **Tecnologías**: React, Go, PostgreSQL, Docker, SonarQube, Clerk ,GeminiAI

Este plan fue actualizado al cierre de cada sprint para reflejar el estado real del avance del proyecto y facilitar la toma de decisiones dentro del equipo.

Product Backlog (versión final)

El Product Backlog final del proyecto incluye todas las historias de usuario definidas y desarrolladas durante los dos sprints del proyecto.

Este backlog refleja:

- El estado real de cada historia (todas completadas y visibles en el video demo).
- Los criterios de aceptación definidos y validados.
- El desglose funcional del sistema desde el login hasta el dashboard de usuario.

Historias de Usuario incluidas:

1. **HU-001 – Login de Usuario**
   Estado: Implementado
   Permite el acceso con correo, contraseña o cuenta Google. Incluye validaciones y mensajes de error.

2. **HU-002 – Resolver Problemas de Programación**
   Estado: Implementado
   El usuario puede seleccionar retos, resolverlos y recibir puntos al acertar.

3. **HU-003 – Filtrar Problemas**
   Estado: Implementado
   Filtros activos por dificultad y etiquetas, con posibilidad de removerlos.

4. **HU-004 – Ver Detalles del Problema**
   Estado: Implementado
   Acceso a la descripción completa del problema, nivel de dificultad y tags.

5. **HU-005 – Escribir y Ejecutar Código**
   Estado: Implementado
   Código ejecutado en contenedor desde el editor integrado. Resultados visibles en la misma interfaz.

6. **HU-006 – Sistema de Recompensas**
   Estado: Implementado
   Al resolver problemas correctamente, se otorgan XP y puntos que se reflejan en el perfil y habilitan compras.

7. **HU-007 – Dashboard de Avances**
   Estado: Implementado
   Vista general del progreso personal y de otros usuarios, incluyendo XP, problemas resueltos y ranking.

Sprint´s Backlogs (de los 3 Sprints)

Durante el desarrollo, organizamos el trabajo en 2 sprints principales y uno de planeación, cada uno con objetivos específicos y un conjunto de historias de usuario priorizadas. A continuación, se detalla el contenido del backlog de cada sprint, incluyendo las tareas técnicas derivadas, responsables y estado de avance.

Sprint 1 – Fundamentos del sistema

Objetivo: Implementar la base técnica del proyecto, incluyendo login, estructura backend, modelo de datos y primeras integraciones.

Historias trabajadas:

- HU-001 – Login de Usuario
- HU-002 – Resolver Problemas de Programación (inicio)
- HU-004 – Ver Detalles del Problema (estructura)
- HU-005 – Escribir y Ejecutar Código (infraestructura)
- HU-007 – Dashboard de Avances (estructura inicial)

Tareas técnicas del sprint:

- Setup del repositorio, ramas y estructura de carpetas.
- Diseño del modelo de usuarios, retos y envíos.
- Implementación del login y autenticación con Clerk.
- Primeros endpoints REST en Go.
- Integración de Docker y PostgreSQL..
- Diseño del layout base en React.
- Inicio del sistema de ejecución de código en contenedor.

Sprint 2 – Funcionalidades completas y experiencia de usuario

Objetivo: Finalizar las funcionalidades, conectar módulos, mejorar la experiencia de usuario y preparar la entrega final.

Historias trabajadas:

- HU-002 – Resolver Problemas de Programación (finalización)
- HU-003 – Filtrar Problemas
- HU-004 – Ver Detalles del Problema (finalización)
- HU-005 – Escribir y Ejecutar Código (finalización)
- HU-006 – Sistema de Recompensas
- HU-007 – Dashboard de Avances (completado)

Tareas técnicas del sprint:

- Integración frontend-backend de todos los módulos.
- Filtro dinámico de problemas por dificultad y etiquetas.
- Renderizado completo de la vista de detalles del reto.
- Ejecución de código mediante contenedor y retorno de resultados.
- Lógica de recompensas y actualización de XP.
- Dashboard con ranking y progreso de usuarios.
- Pruebas funcionales y ajustes finales.
- Actualización de documentación y despliegue.

Resultado: Todas las funcionalidades quedaron implementadas y operativas en el entorno final.

Minutas de Planeación de los 3 Sprints (3 en total)

Durante el desarrollo, se realizaron tres reuniones de planeación para definir objetivos, asignar tareas y priorizar historias de usuario. A continuación se presentan las minutas correspondientes:

Minuta – Sprint 0: Planeación General del Proyecto

Participantes: Roger, Ramiro, Fer, Imanol
Objetivo: Definir el alcance general del proyecto, herramientas a utilizar y estructura de trabajo.

Temas tratados:

- Revisión del enunciado y requerimientos del reto.
- Definición del objetivo principal: plataforma gamificada de retos de programación.
- Selección de tecnologías: Go, React, PostgreSQL, Docker, GitHub Actions.
- División preliminar de roles por especialidad.
- Elección de herramientas de gestión (GitHub Projects).
- Planificación de 2 sprints de desarrollo.

Acuerdos:

- Documentar avance desde el inicio en el repositorio.
- Utilizar GitHub Issues para gestión de tareas.
- Priorizar historias críticas para MVP en Sprint 1.

Minuta – Sprint 1: Infraestructura y Módulo de Login

Participantes: Roger, Ramiro, Fer, Imanol, Diego
Objetivo: Planificar tareas técnicas para el desarrollo de la base del sistema.

Temas tratados:

- Implementar login de usuario (Clerk).
- Construcción del modelo de base de datos.
- Inicio del módulo de problemas.
- Contenedores Docker para backend y frontend.

Acuerdos:

- Definir los endpoints y casos de prueba iniciales.
- Tener MVP funcional de login y lectura de problemas al final del sprint.

Minuta – Sprint 2: Integración, Funcionalidades Avanzadas y Entrega Final

Participantes: Roger, Ramiro, Fer, Imanol
Objetivo: Finalizar el desarrollo completo e integrar todos los módulos.

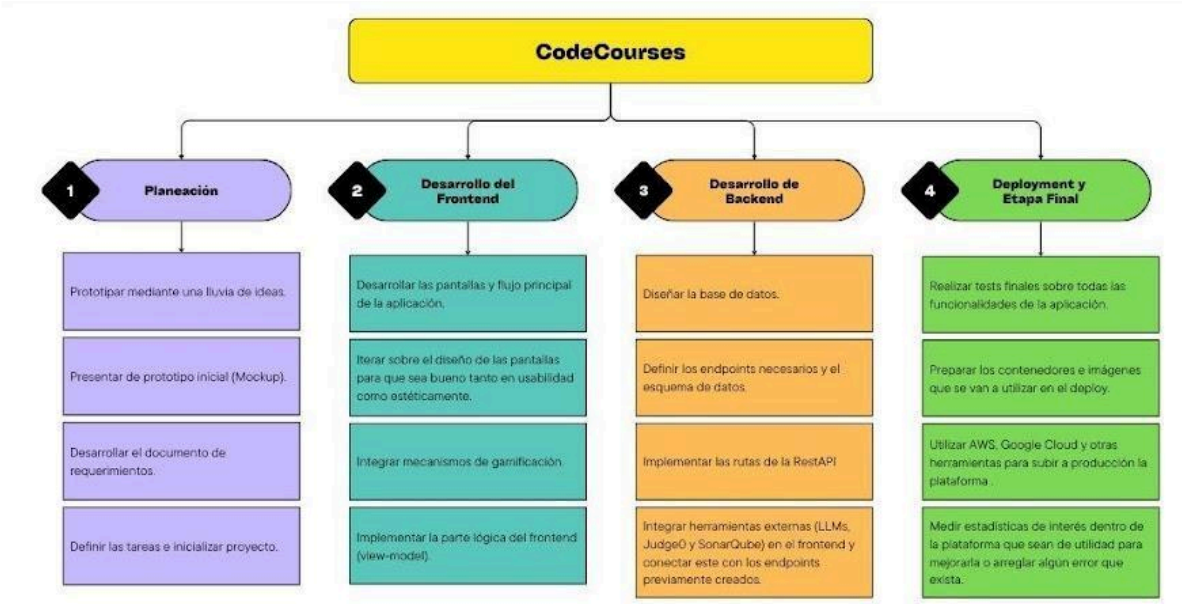
Temas tratados:

- Finalizar ejecución de código en contenedor.
- Habilitar sistema de recompensas y dashboard.
- Implementar filtros dinámicos y vista detallada.
- Preparar presentación final y documentación.
- Revisar despliegue en entorno productivo.

Acuerdos:

- Tener ambiente listo para grabar el video demo.
- Realizar pruebas funcionales al finalizar cada módulo.
- Dividir trabajo de documentación por sección.

Diagramas Burn Down Chart de los 3 Sprints (3 diagramas en total)



Sprint 0 – Planeación General

- **Objetivo del gráfico**: Visualizar tareas relacionadas a la preparación del entorno, definición del stack tecnológico, asignación de roles, diseño preliminar y configuración de herramientas (CI/CD, Docker, GitHub).

- **Observación**: Se logró cumplir con el plan inicial en el tiempo previsto, con algunas tareas administrativas resueltas más rápido de lo proyectado.

Sprint 1 – Infraestructura, Login y Primeros Módulos

- **Tareas planificadas**: Setup de backend, base de datos, login, estructura de problemas y despliegue inicial.
- **Progreso**: El equipo mantuvo un ritmo constante. Algunas tareas técnicas complejas (como integración de Clerk y Docker) se resolvieron antes de lo previsto, lo que permitió adelantar parte del módulo de problemas.
- **Resultado**: El gráfico muestra una caída gradual del esfuerzo restante, cumpliendo con lo estimado.

Sprint 2 – Funcionalidades Completas e Integración

- **Tareas planificadas**: Finalizar filtros, ejecución de código, recompensas, dashboard, documentación y video demo.

- **Progreso**: Se presentaron picos de esfuerzo intermedio por validaciones de ejecución de código y pruebas de integración, pero el equipo logró retomar el ritmo.

- **Resultado**: El gráfico refleja una curva real más irregular pero alineada al objetivo final, con todas las tareas cerradas a tiempo.

Burn up

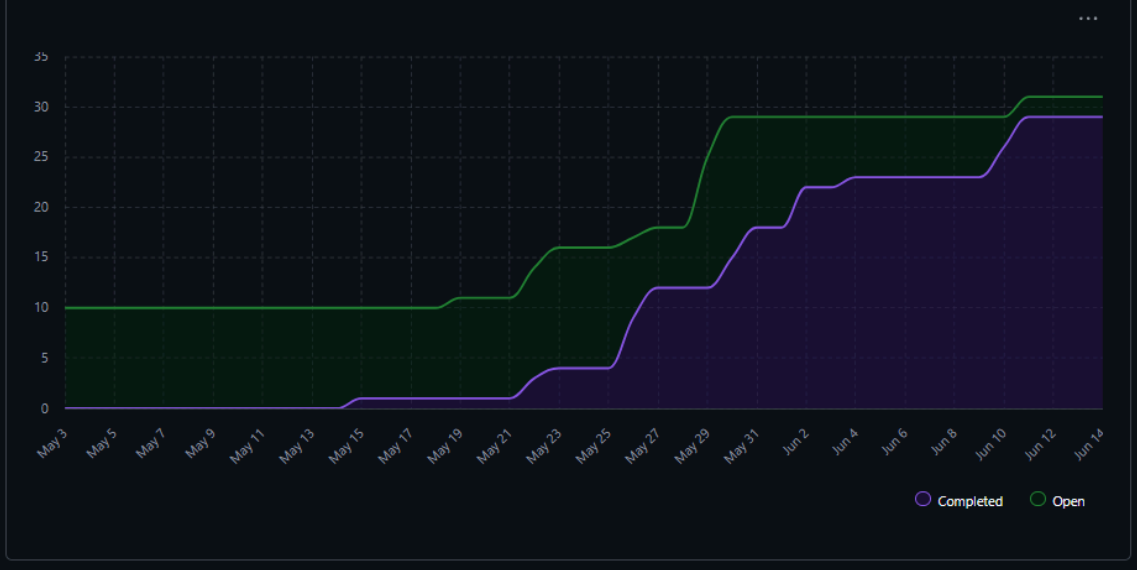
Configure

The Burn up chart shows the progress of your project items over time, showing how much work has been completed and how much is left to do. Use this chart to view progress, spot trends, and identify bottlenecks to help move the project forward.

is:issue 60 Discard

2W 1M 3M Max

May 3 - Jun 14



Al final el diagrama se termino viendo asi: