

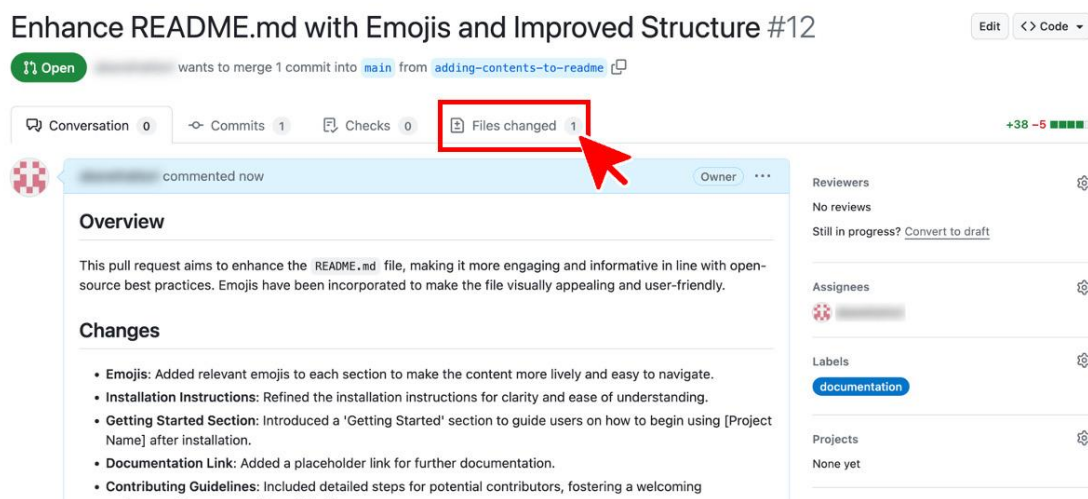
Revisión de pull requests

En primer lugar, es esencial comprender cómo realizar revisiones en GitHub y luego explorar las mejores prácticas que merecen atención. Cuando comienzas a aprender GitHub, es posible que no te involucres de inmediato en revisiones de código completas. Sin embargo, comprender cómo se deben realizar las revisiones puede influir en tus expectativas y en el contenido que incluyes al escribir un pull request.

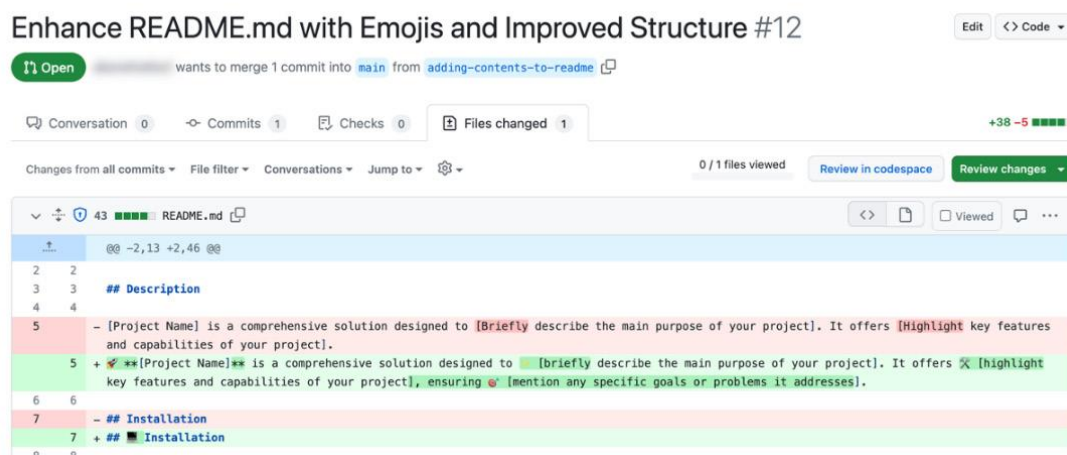
Recuerda: estas son prácticas que se pueden mejorar gradualmente y, en última instancia, deben optimizarse para tu equipo.

Conceptos básicos de la revisión

Acceder a la interfaz de revisión en GitHub es sencillo. Desde la página de un pull request, abre la pestaña "Files changed" para ver cuántos archivos se han modificado:














Una vez que hayas abierto la pestaña "Files changed", examina el contenido. Además de la captura de pantalla de diferencia línea por línea que se muestra a continuación, hay una opción de vista que organiza los cambios lado a lado, lo cual es muy recomendable. Las líneas con el prefijo - indican eliminaciones, mientras que aquellas con + son adiciones:



Para comentar sobre cambios de línea específicos, utiliza el botón + en la izquierda. Esto también permite seleccionar varias líneas, facilitando la iniciación de discusiones de manera eficiente:

5 - [Project Name] is a comprehensive solution designed to [Briefly describe the main purpose of your project]. It offers [Highlight key features and capabilities of your project].

5 + 🚀 ****[Project Name]**** is a comprehensive solution designed to 🌟 [briefly describe the main purpose of your project]. It offers ✨ [highlight key features and capabilities of your project], ensuring 🎯 [mention any specific goals or problems it addresses].












Write Preview  H B I        @   

LGTM!! Thanks :)|

Cancel Add single comment Start a review

Al revisar, es posible que desees proponer cambios específicos. Esto se puede hacer presionando el botón "Suggestion" en la pantalla, que copia el contenido de la línea relevante. Edita este contenido en tu comentario, pero ten en cuenta que, si bien se pueden agregar comentarios a líneas eliminadas, no se pueden hacer sugerencias:

33 + Contributions are what make the open-source community an amazing place to learn, inspire, and create. Any contributions you make are ****greatly appreciated****.

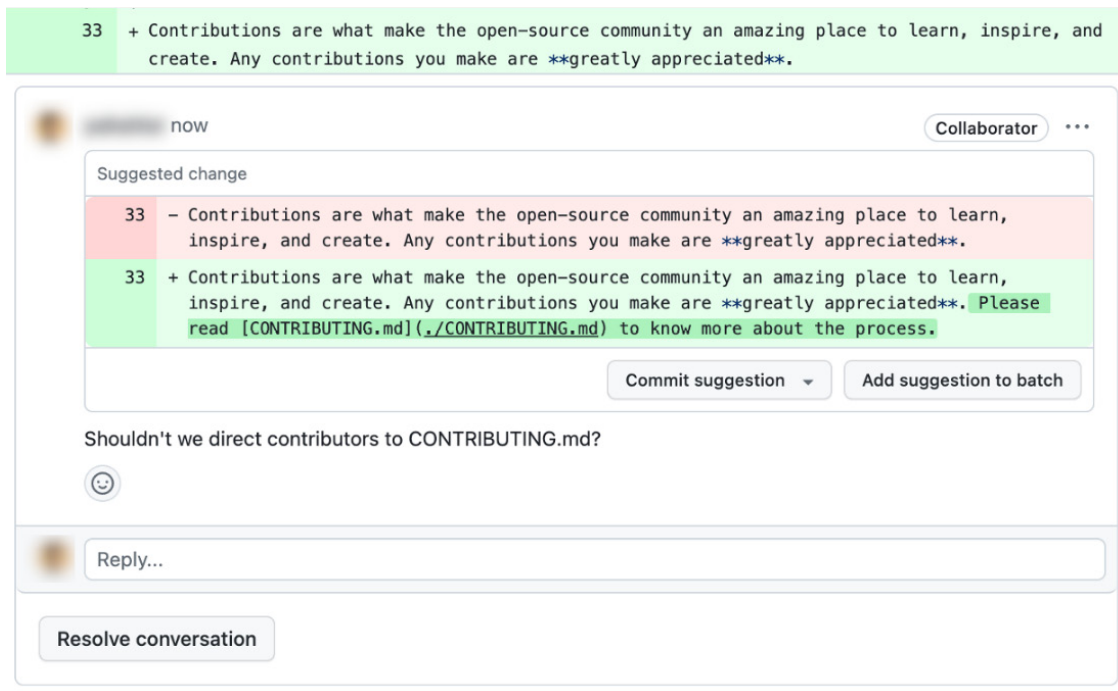
Write Preview  H B I        @   

```suggestion  
Contributions are what make the open-source community an amazing place to learn, inspire, and create. Any contributions you make are **\*\*greatly appreciated\*\***. Please read [CONTRIBUTING.md](./CONTRIBUTING.md) to know more about the process.  
```

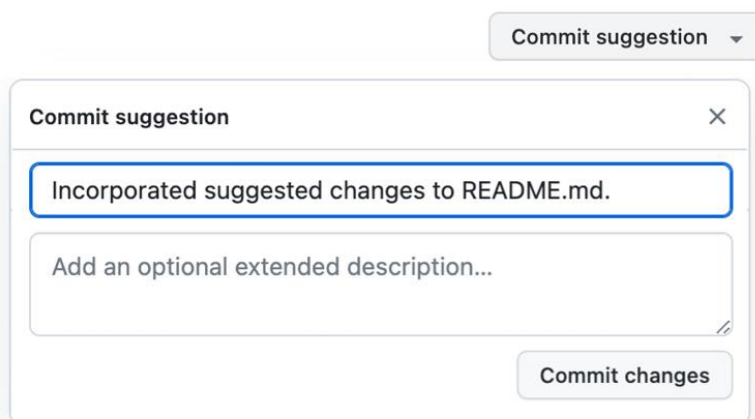
Shouldn't we direct contributors to CONTRIBUTING.md?

Cancel Add single comment Start a review

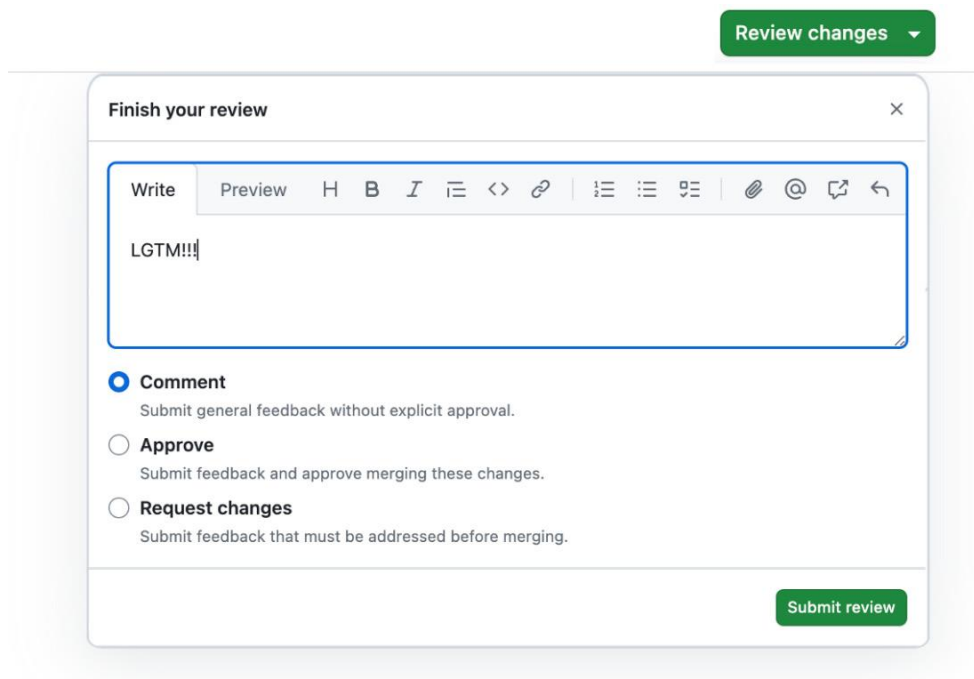
Si consideras que una sugerencia es buena, presiona el botón "Add single comment" para comentar. Para múltiples comentarios de revisión, utiliza el botón "Start a review", que es útil cuando hay muchas ediciones. Los comentarios de revisión, incluidas las sugerencias, aparecen tanto en el hilo del pull request como entre las diferencias de código, lo que facilita la revisión del código:



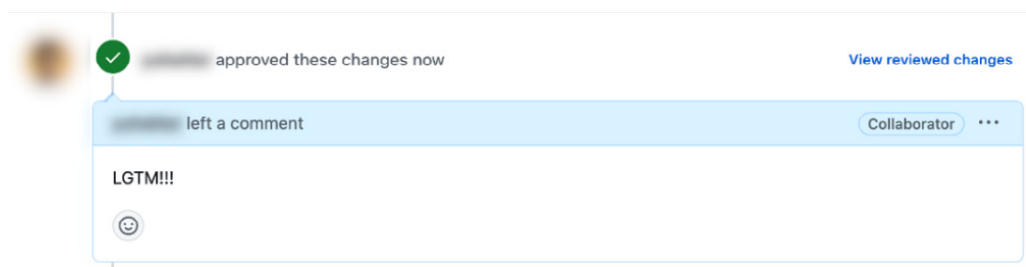
Después de completar la revisión, verás los botones "Commit suggestion" y "Add suggestion to batch". La brillantez de GitHub radica en permitir que estos cambios se confirmen directamente a través de la interfaz de usuario, en lugar de necesitar hacerlo a través de la interfaz de línea de comandos o un editor. "Commit suggestion" confirma un cambio individual, mientras que "Add suggestion to batch" permite registrar múltiples cambios como un solo commit. Esto evita commits innecesarios cuando tus pull requests reciben muchos cambios. Presionar el botón "Commit suggestion" abre una pantalla para ingresar un mensaje de commit y una descripción, después de lo cual el cambio se confirma en la rama del pull request:



Una vez que la revisión esté completa, es hora de realizar una revisión. Ten en cuenta que aprobar una revisión no fusiona automáticamente el pull request, pero lo marca como aprobado por ti. Aquí, puedes elegir entre "Commit", "Approve" y "Request changes". La opción "Comment" es solo para hacer un comentario, "Approve" indica aprobación, y "Request changes" se utiliza para sugerir enmiendas necesarias:



Si apruebas, tu comentario y estado de aprobación se publican en el hilo así:



Así, GitHub simplifica el proceso de revisión y habilita muchas funcionalidades a través de su interfaz gráfica de usuario. La combinación de cambios de código y comentarios relacionados enriquece el contexto, permitiendo que los revisores, los revisados y los miembros actuales o futuros del equipo que buscan contexto histórico comprendan e interactúen con la información de manera efectiva.

Dominar esta función puede llevar a una comunicación altamente transparente dentro de tu equipo.

Echemos un vistazo a lo que debes considerar al revisar o ser revisado.

Mejores prácticas de revisión

En el dinámico panorama del desarrollo de software, las revisiones en GitHub se erigen como un componente crucial para tender un puente entre el esfuerzo individual y la excelencia colectiva. Estas revisiones van más allá de la mera examinación de código para fomentar un entorno colaborativo y rico en aprendizaje. Dominar las revisiones en GitHub es crucial para los equipos que buscan procesos de desarrollo de alta calidad. Aquí hay algunas pautas sobre cómo lograrlo:

- Cultivando la mentalidad del revisor: La efectividad de las revisiones en GitHub comienza con la mentalidad del revisor. Enfatizar la retroalimentación rápida es clave, ya que mantiene el impulso del desarrollo. Sin embargo, la velocidad debe equilibrarse con la

minuciosidad, reconociendo que el código es un activo colectivo del equipo. Esta mentalidad fomenta un enfoque constructivo y libre de egos hacia las revisiones, viéndolas no solo como una tarea, sino como una oportunidad para un diálogo significativo y documentación para los miembros actuales y futuros del equipo.

- Mejorando la comunicación en las revisiones: La comunicación efectiva es fundamental en las revisiones de GitHub. Utilizar Markdown asegura que las revisiones sean accesibles y comprensibles. Ser específico y conciso en los comentarios evita confusiones y acelera el proceso de revisión. El refuerzo positivo y la retroalimentación constructiva, a menudo complementados con emojis, crean un entorno de revisión acogedor y psicológicamente seguro, alentando la adopción rápida y entusiasta de los cambios.
- Promoviendo la diversidad y la autoría en las revisiones: La diversidad en el proceso de revisión introduce perspectivas variadas, enriqueciendo la revisión y protegiendo contra puntos ciegos. Alentar a los desarrolladores a revisar su código basado en retroalimentación fomenta el empoderamiento y el desarrollo de habilidades. Además, preservar la autoría, incluso en prácticas como el squash merging, es crucial para mantener la motivación y la responsabilidad individual.
- Manteniendo estándares en escenarios urgentes: Incluso en escenarios urgentes, como hotfixes, la calidad y minuciosidad de las revisiones no deben comprometerse. Mantener estándares bajo presión es esencial, ya que cada revisión contribuye a la ética de trabajo general del equipo y a la calidad del código.
- Integrando la automatización para la eficiencia: Incorporar la automatización en el proceso de revisión, a través de pruebas automáticas, verificaciones de sintaxis y reglas de linter, agiliza el proceso de revisión. Esto permite que los revisores humanos se concentren en aspectos más complejos del código. El uso de plantillas para revisiones también asegura consistencia y minuciosidad, manteniendo una alta calidad de revisión en todo el equipo.

En conclusión, las revisiones en GitHub son más que un mecanismo para el aseguramiento de la calidad del código; son una parte vital para fomentar el crecimiento del equipo, el aprendizaje y la colaboración. Al adoptar estas mejores prácticas, los equipos elevan no solo su código, sino también su entorno de trabajo, promoviendo el respeto mutuo, la mejora continua y un espíritu colaborativo. Una revisión bien ejecutada es, de hecho, un paso hacia un proceso de desarrollo más cohesivo, eficiente y resistente.

Sacando lo mejor de GitHub

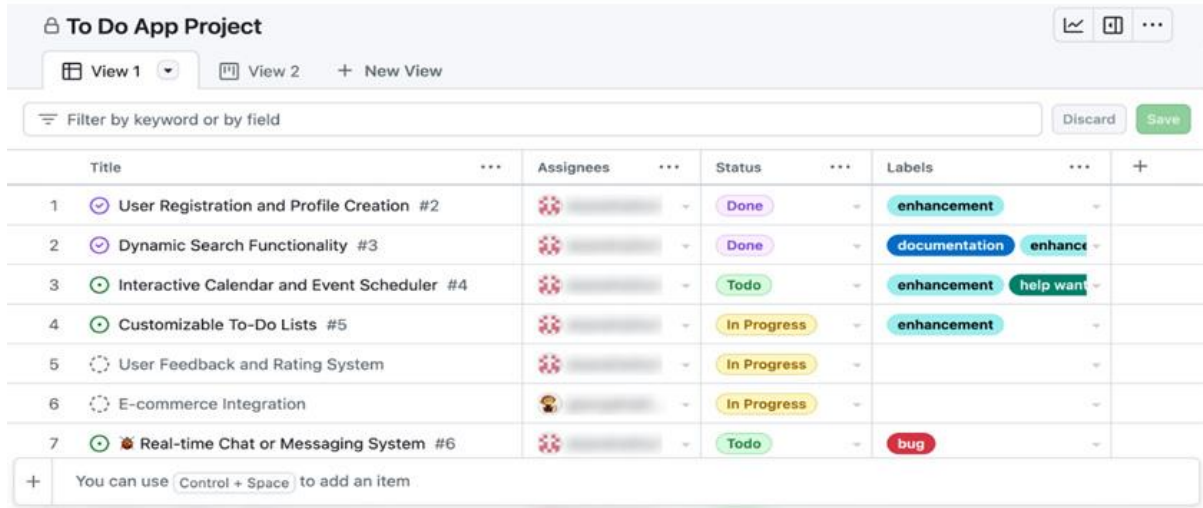
Explorar características adicionales en GitHub puede mejorar enormemente tu experiencia en la gestión de repositorios. Si bien una exploración detallada de cada característica sería extensa, una breve introducción a cada una puede resaltar su importancia.

GitHub Projects: Gestionando tus issues y pull requests en un solo lugar

GitHub Projects es una herramienta que te permite gestionar issues, borradores de issues y pull requests en un solo lugar.

GitHub Projects permitirá una gestión más flexible que solo los issues y pull requests de GitHub. Representa una evolución significativa en cómo los equipos de desarrollo pueden organizar, rastrear y gestionar su trabajo en GitHub. A diferencia del alcance lineal y algo limitado de los issues y pull

requests, GitHub Projects ofrece un enfoque multifacético para la gestión de proyectos, alineándose perfectamente con la naturaleza colaborativa e iterativa del desarrollo de software moderno:



	Title	Assignees	Status	Labels
1	🔗 User Registration and Profile Creation #2	👤	Done	enhancement
2	🔗 Dynamic Search Functionality #3	👤	Done	documentation, enhance
3	🔗 Interactive Calendar and Event Scheduler #4	👤	Todo	enhancement, help wanted
4	🔗 Customizable To-Do Lists #5	👤	In Progress	enhancement
5	🔗 User Feedback and Rating System	👤	In Progress	
6	🔗 E-commerce Integration	👤	In Progress	
7	🔗 Real-time Chat or Messaging System #6	👤	Todo	bug

En el corazón de GitHub Projects se encuentra el tablero Kanban, una herramienta altamente visual que mejora la visualización del flujo de trabajo. Este tablero permite a los equipos crear columnas personalizadas que reflejan sus etapas del flujo de trabajo, desde "To Do" hasta "In Progress" y "Done". La simplicidad de arrastrar y soltar issues y pull requests a través de estas columnas mejora significativamente la transparencia del flujo de trabajo y ayuda a rastrear el progreso de las tareas de manera más intuitiva:

To Do App Project

View 1

View 2

New View

Filter by keyword or by field

Discard

Save

Todo

15

This item hasn't been started

your-repository #4

Interactive Calendar and Event Scheduler

enhancement

help wanted

your-repository #6

Real-time Chat or Messaging System

bug

Add item

In Progress

3

This is actively being worked on

your-repository #5

Customizable To-Do Lists

enhancement

Draft

User Feedback and Rating System

Draft

Add item

Done

2

This has been completed

your-repository #2

User Registration and Profile Creation

enhancement

your-repository #3

Dynamic Search Functionality

documentation

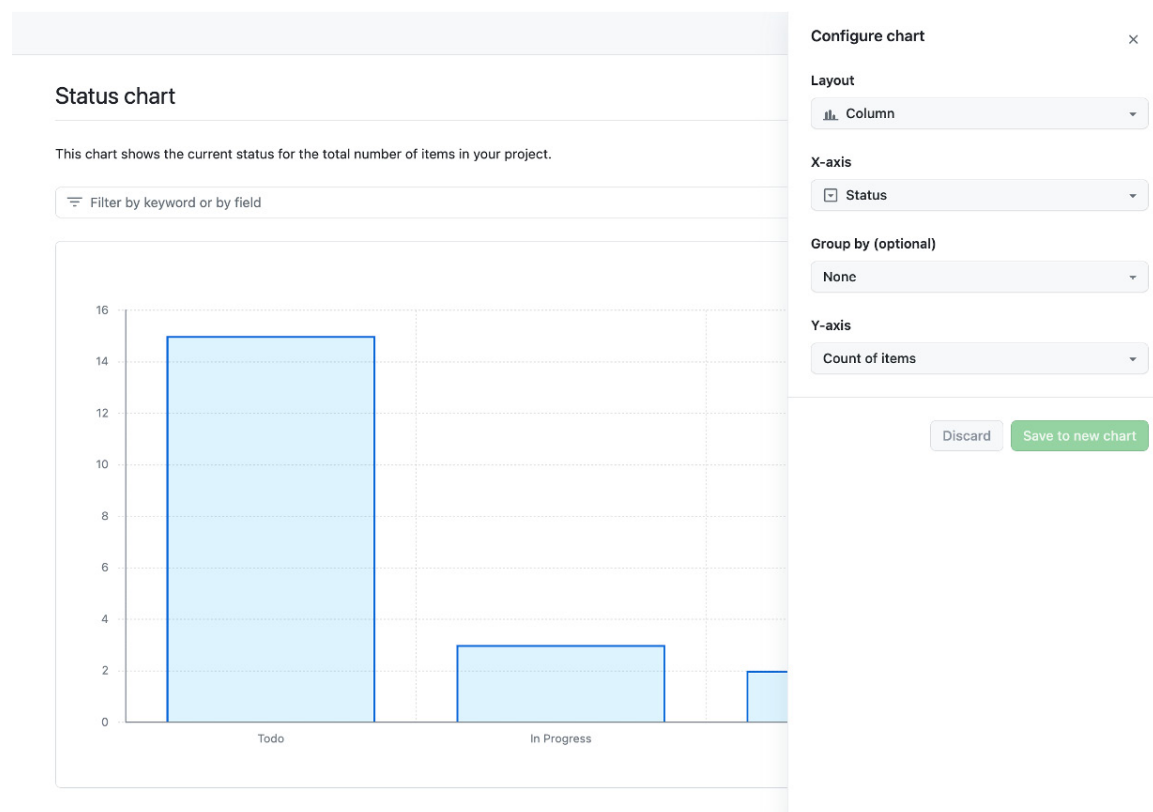
enhancement

Add item

La colaboración se mejora aún más a través de herramientas como **asignatarios** y **hitos**. Los asignatarios aseguran que las responsabilidades estén claramente definidas, mientras que los hitos ayudan a rastrear el progreso hacia objetivos o plazos clave. Este enfoque estructurado para la

colaboración asegura que cada miembro del equipo esté alineado con los objetivos del proyecto y entienda su papel en su consecución.

Los **informes integrados del proyecto** son otro aspecto que distingue a GitHub Projects. Proporcionan una visión integral del progreso del proyecto. Las métricas de DevOps son invaluable para evaluar el rendimiento del equipo e identificar áreas de mejora. Este enfoque basado en datos para la gestión de proyectos permite a los equipos tomar decisiones informadas y optimizar sus flujos de trabajo:



GitHub Projects también ofrece una variedad de plantillas de proyecto, desde las predefinidas para tipos de proyectos comunes, como rastreador de bugs o Kanban, hasta la capacidad de crear plantillas personalizadas adaptadas a flujos de trabajo específicos del equipo. Estas plantillas ahorran tiempo y proporcionan una estructura consistente de la que los equipos pueden depender.

Además, la inclusión de filtros de issues y pull requests dentro del tablero del proyecto mejora la capacidad de gestionar tareas de manera eficiente. Los equipos pueden filtrar por etiquetas, asignatarios, hitos y más, lo que les permite encontrar y enfocarse rápidamente en las tareas que más importan.

Una de las ventajas más significativas de GitHub Projects es su capacidad para gestionar issues y pull requests de múltiples repositorios en un solo proyecto. Esta característica es particularmente beneficiosa para proyectos más grandes que abarcan varios repositorios, proporcionando una vista holística y una experiencia de gestión coherente.

GitHub Projects va más allá de los confines tradicionales del seguimiento de issues y la gestión de pull requests, ofreciendo una plataforma de gestión de proyectos integral, flexible y altamente colaborativa. Al utilizar sus características, los equipos de desarrollo pueden mejorar su productividad, mejorar la colaboración y llevar sus proyectos hacia resultados exitosos, todo dentro del ecosistema familiar de GitHub.

Esta alineación con los principios de la cultura DevOps, que prioriza la colaboración, la eficiencia y la transparencia, convierte a GitHub Projects en una gran herramienta en el panorama moderno del desarrollo de software.

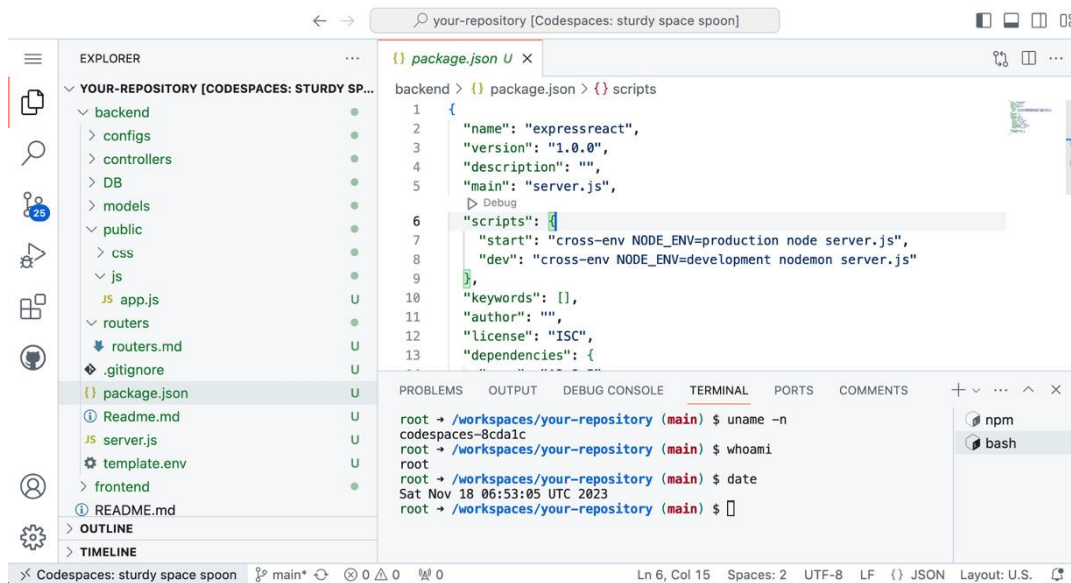
GitHub Codespaces: Transformando flujos de trabajo de desarrollo con entornos basados en la nube

GitHub Codespaces revoluciona la forma en que los desarrolladores trabajan al ofrecer entornos de desarrollo totalmente configurables y basados en la nube directamente dentro de GitHub. Marca un paso significativo en la evolución de los flujos de trabajo de desarrollo, facilitando la colaboración de los equipos y simplificando sus prácticas de DevOps.

GitHub Codespaces agiliza el desarrollo con características clave:

- Inicio rápido y espacios de trabajo uniformes: Permite la configuración instantánea de entornos basados en la nube, alineando los esfuerzos del equipo y reduciendo la complejidad de la configuración
- Colaboración e integración con pull requests: Facilita revisiones de código y discusiones más sencillas dentro del entorno de Codespaces, mejorando el trabajo en equipo en diferentes ubicaciones
- Entornos seguros y aislados: Ofrece espacios de trabajo seguros y contenerizados para cada usuario, protegiendo la integridad del proyecto y la seguridad de los datos

Como una implementación de Visual Studio Code disponible en el navegador, GitHub Codespaces ofrece las características y extensiones disponibles en VS Code. Aunque hay limitaciones en las extensiones disponibles en la versión de Codespaces, esta familiaridad es una gran ventaja. Permite a los desarrolladores aprovechar una herramienta de desarrollo potente y popular con los beneficios adicionales de accesibilidad en la nube e integración:



Comenzar un nuevo codespace es sencillo. Con solo unos pocos clics, los desarrolladores pueden iniciar un entorno de desarrollo que se configura automáticamente con las configuraciones y herramientas especificadas en la configuración .devcontainer del repositorio. Esto asegura la consistencia en todos los entornos de desarrollo, un aspecto fundamental de las prácticas de DevOps que enfatiza la reproducibilidad y reduce los problemas de "funciona en mi máquina".

Uno de los beneficios más significativos de GitHub Codespaces es la eliminación de la necesidad de clonar archivos pesados y configurar entornos de desarrollo complejos localmente. Esto no solo ahorra un tiempo significativo en la incorporación de nuevos ingenieros, sino que también simplifica el proceso de cambiar entre diferentes proyectos, cada uno con sus requisitos de entorno únicos. Al alojar el entorno de desarrollo en la nube, Codespaces reduce significativamente la sobrecarga de gestionar configuraciones de desarrollo locales.




El rendimiento y la personalización son aspectos clave de GitHub Codespaces. Los desarrolladores pueden elegir especificaciones de hardware para sus codespaces, asegurando que tengan los recursos necesarios para sus tareas. La personalización se extiende al propio entorno de desarrollo, con soporte para extensiones de Visual Studio Code y configuraciones personalizadas del editor, reforzando el enfoque DevOps en la experiencia del desarrollador.

Desde una perspectiva administrativa, GitHub Codespaces permite a los líderes de la organización establecer políticas específicas con respecto al entorno de desarrollo, incluidas las limitaciones sobre los tipos de máquinas que los usuarios pueden seleccionar para sus codespaces. Esta capacidad asegura que los entornos de desarrollo se alineen con los estándares organizacionales y las estrategias de gestión de recursos, ofreciendo un enfoque equilibrado entre costo y rendimiento dentro de las pautas de la empresa:

Policy name *

Machine-Policy

Constraints Add constraint ▾

 **Machine types**
2-core, 4-core  

Change policy target
This policy applies to all repositories within your organization

All repositories ▾

Save

Allowed values

- ☒ 2-core 8GB RAM • 32GB storage
- ☒ 4-core 16GB RAM • 32GB storage
- ☐ 8-core 32GB RAM • 64GB storage
- ☐ 16-core 64GB RAM • 128GB storage

Además, ofrece una combinación notable de colaboración y seguridad, logrando un equilibrio perfecto para los equipos de desarrollo modernos. La integración directa de los pull requests en el entorno de Codespaces mejora significativamente el proceso colaborativo, facilitando que los equipos propongan, discutan y fusionen cambios de manera eficiente, independientemente de sus ubicaciones físicas. Este flujo de trabajo simplificado para revisiones de código y colaboración se combina con una poderosa ventaja de seguridad.

Codespaces reduce los riesgos asociados con que los empleados descarguen todos los datos del código fuente localmente, abordando preocupaciones de seguridad comunes que surgen del almacenamiento y gestión de archivos locales. El espacio de trabajo de cada usuario está aislado de manera segura, lo que mitiga aún más los posibles problemas de seguridad relacionados con las prácticas de desarrollo locales. Esta integración cuidadosa de colaboración y seguridad dentro de Codespaces no solo aumenta la productividad, sino que también refuerza la protección de los datos del proyecto, ofreciendo una solución de desarrollo segura y colaborativa.

GitHub Codespaces es más que solo un entorno de desarrollo; es un catalizador para mejorar las prácticas de DevOps. Al ofrecer una plataforma flexible, colaborativa e integrada, empodera a los equipos para innovar más rápido, colaborar más efectivamente y entregar software de mayor calidad, todo dentro del ecosistema de GitHub. Esto convierte a GitHub Codespaces en una herramienta indispensable en el desarrollo de software moderno y en el panorama DevOps.

GitHub Discussions: Fomentando la comunidad y la colaboración

GitHub Discussions sirve como una plataforma vital para construir y fomentar la comunidad dentro de los repositorios de GitHub. Representa un avance significativo en cómo los desarrolladores, colaboradores y usuarios interactúan y colaboran. Esta característica va más allá de la comunicación tradicional de issues y pull requests, ofreciendo un espacio dedicado para preguntas, ideas y discusiones. GitHub Discussions crea un entorno donde la comunidad más amplia, incluidos aquellos que pueden no estar directamente involucrados en las contribuciones de código, puede participar, compartir ideas y proporcionar retroalimentación:

Welcome to your-repository Discussions! #7

The screenshot shows a GitHub Discussion thread. At the top, it says 'Welcome to your-repository Discussions! #7' and 'announced in Announcements'. The discussion is categorized under 'Announcements' and has '1 participant'. The content of the discussion is a welcome message from the repository maintainer, encouraging users to ask questions, share ideas, and engage with the community. Below the message, there is a '0 comments' section and a 'Add a comment' form. On the right side, there are options to 'Unsubscribe', 'Lock conversation', 'Transfer this discussion', 'Edit pinned discussion', 'Unpin discussion', 'Pin discussion to Announcements', 'Create issue from discussion', and 'Delete discussion'.

Un aspecto clave de GitHub Discussions es su capacidad para organizar la comunicación de manera efectiva. Las discusiones pueden categorizarse en diferentes tipos, como preguntas y respuestas, anuncios o discusiones generales. Esta categorización ayuda a mantener la claridad y el enfoque, facilitando que los usuarios encuentren conversaciones relevantes y contribuyan de manera significativa:

The screenshot shows the GitHub Discussions interface. At the top, there is a search bar with 'is:open' and a 'Sort by: Latest activity' dropdown. Below the search bar, there is a 'Categories' section with options like 'View all discussions', 'Announcements', 'General', 'Ideas', 'Polls', 'Q&A', and 'Show and tell'. The main section is titled 'Discussions' and lists several discussions. The first discussion is 'Introducing React Learning' started 2 minutes ago. The second discussion is 'Welcome to your-repository Discussions!' announced 5 minutes ago. The third discussion is 'Is subtask functionality required?' started 3 days ago. The fourth discussion is 'Announcement 2023/11/18' announced 4 minutes ago.

GitHub Discussions también juega un papel crucial en la compartición y preservación del conocimiento. A diferencia de los canales de comunicación transitorios, como salas de chat, las discusiones son persistentes y buscables, lo que las convierte en un recurso valioso para los miembros actuales y futuros de la comunidad. Esta calidad de archivo asegura que el conocimiento compartido en las discusiones continúe beneficiando a la comunidad mucho después de que las conversaciones hayan tenido lugar.

Excelencia en repositorios de GitHub

Hay muchas configuraciones en los repositorios de GitHub que pueden promover la colaboración. Estas configuraciones pueden reducir la necesidad de temer al fracaso y hacer que el proceso de

revisión sea más estandarizado. No cubriremos todas aquí, pero algunas de las principales son particularmente importantes.

Reglas del repositorio: Simplificación del flujo de trabajo y aseguramiento de la calidad del código

Las reglas de ramas son importantes para gestionar un repositorio de manera que logre un equilibrio entre la seguridad y la colaboración. Si bien GitHub permite establecer permisos para el repositorio o funciones específicas, no ofrece permisos basados en el código. Confiar únicamente en los permisos de escritura puede restringir la participación y obstaculizar la comunicación abierta. Las reglas de ramas permiten a los administradores del repositorio hacer cumplir políticas específicas en las ramas, particularmente las críticas, como las ramas main o master.

Estas políticas incluyen requisitos para revisiones de pull request, verificaciones de estado y restricciones sobre quién puede hacer push a la rama. Al configurar estas reglas, los equipos pueden asegurarse de que el código fusionado en ramas importantes cumpla con estándares de calidad predefinidos y que el proceso se alinee con el flujo de trabajo del equipo.

El conjunto de reglas permite una variedad de configuraciones. Se pueden aplicar solo a ramas específicas, o se puede autorizar a los administradores para omitirlas:

[Rulesets](#) / Team's Protection Ruleset ...

Ruleset Name

Team's Protection Ruleset

Enforcement status

Active

Bypass list

+ Add bypass

Repository admin Role

Always

Target branches

+ Add target

Default

releases/**/*

Applies to 1 target: `main`.

Branch protections

Restrict creations

☐ Require status checks to pass

Choose which status checks must pass before the ref is updated. When enabled, commits must first be pushed to another ref where the checks pass.

☒ Block force pushes

Prevent users with push access from force pushing to refs.

Save changes

Revert changes

Uno de los principales beneficios de las reglas de ramas es la aplicación de la revisión de código. Al requerir que los pull requests reciban un cierto número de aprobaciones antes de la fusión, los equipos pueden asegurarse de que cada cambio sea examinado y validado. Este proceso de revisión por pares no solo mejora la calidad del código, sino que también fomenta el intercambio de conocimientos y la colaboración entre los miembros del equipo:

☒ **Require a pull request before merging**
Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

Additional settings ^

Required approvals

2 ▾

The number of approving reviews that are required before a pull request can be merged.

☐ **Dismiss stale pull request approvals when new commits are pushed**
New, reviewable commits pushed will dismiss previous pull request review approvals.

☐ **Require review from Code Owners**
Require an approving review in pull requests that modify files that have a designated code owner.

☐ **Require approval of the most recent reviewable push**
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☐ **Require conversation resolution before merging**
All conversations on code must be resolved before a pull request can be merged.

Las **verificaciones de estado** son otra parte crítica de las reglas de ramas. Estas verificaciones pueden incluir pruebas automatizadas, resultados de linter de código o cualquier otro tipo de proceso automatizado que verifique la calidad y funcionalidad del código. Al requerir que estas verificaciones pasen antes de la fusión, los equipos pueden prevenir errores y problemas para que no lleguen al código base principal, manteniendo así altos estándares de calidad del código:

☒ **Require status checks to pass**
Choose which status checks must pass before the ref is updated. When enabled, commits must first be pushed to another ref where the checks pass.

Additional settings ^

☒ **Require branches to be up to date before merging**
Whether pull requests targeting a matching branch must be tested with the latest code. This setting will not take effect unless at least one status check is enabled.

lint-check X

Enter the name of a status check

+

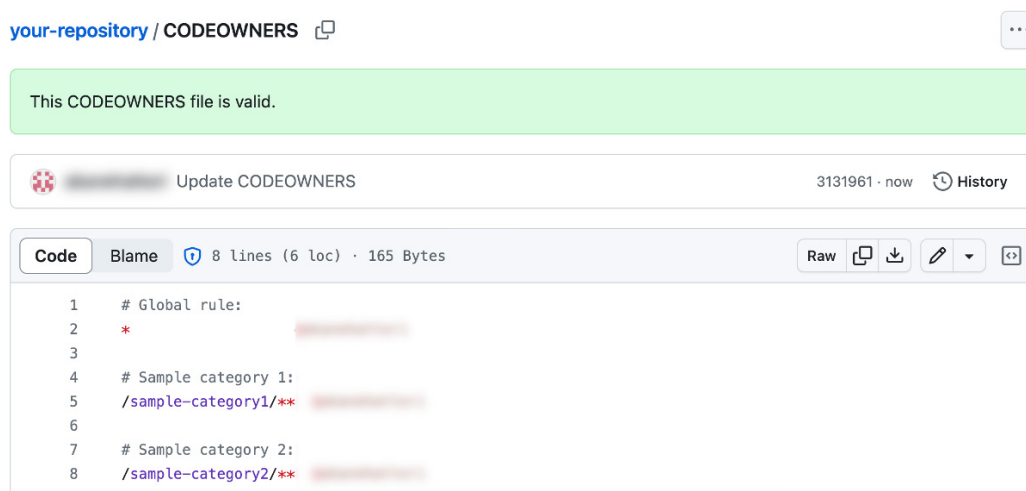
No status checks found
[Learn more about status checks](#)

Las reglas de ramas son una configuración fundamental en GitHub para salvaguardar la calidad del código y hacer cumplir la disciplina del flujo de trabajo. Permiten a los equipos automatizar y hacer cumplir aspectos críticos de su proceso de desarrollo, alineándose con los principios de DevOps. Al utilizar las reglas de ramas, los equipos pueden asegurarse de que su código base permanezca estable, seguro y consistente, apoyando así la entrega de software de alta calidad en un entorno

colaborativo. Esto convierte a las reglas de ramas en un elemento esencial en el desarrollo de software moderno y en un recurso invaluable en el conjunto de herramientas de DevOps.

CODEOWNERS: Revisión y propiedad simplificadas

El archivo CODEOWNERS en GitHub es una herramienta simple pero poderosa para asignar automáticamente revisores a pull requests y aclarar la propiedad de áreas específicas del código. Este archivo, ubicado en la raíz, docs/ o directorio .github/ del repositorio, enumera individuos o equipos junto con patrones de archivos. Cuando se realizan cambios en archivos que coinciden con estos patrones, los propietarios del código especificados son solicitados para revisión:




Los beneficios de usar un archivo CODEOWNERS incluyen los siguientes:

- Asignación automática de revisores: Agiliza el proceso de pull request al asignar automáticamente a los revisores correctos, asegurando que los cambios sean revisados por los expertos apropiados
- Propiedad clara: Aclara quién es responsable de partes específicas del código base, ayudando en la toma de decisiones más rápida y el mantenimiento más eficiente

Este aspecto de CODEOWNERS es particularmente importante, ya que se relaciona directamente con la seguridad del despliegue. En entornos donde CI/CD es la norma, asegurar que solo se despliegue código bien revisado y aprobado es crítico. Esto añade una capa de seguridad y eficiencia al proceso de despliegue, incorporando los principios preventivos y proactivos centrales para las prácticas DevOps efectivas.

Plantillas de issues y pull requests

Las plantillas en GitHub mejoran la colaboración y mantienen la consistencia en varios aspectos de la gestión de proyectos:



Bug report
Create a report to help us improve

Get started

Custom issue template
Describe this issue template's purpose here.

Get started


Feature request
Suggest an idea for this project

Get started

Las plantillas de issues guían a los colaboradores en la creación de informes de issues detallados y estructurados. Al proporcionar plantillas específicas para diferentes tipos de issues (informes de errores, solicitudes de características, etc.), aseguras que se incluya toda la información necesaria, facilitando la comprensión y resolución más rápida:

Issue: Bug report

Create a report to help us improve. If this doesn't look right, [choose a different type](#).



Add a title

Title

Add a description

WritePreview

HBI≡<>🔗☰☲☳🔗@🗨️↩️🗑️

****Describe the bug****
A clear and concise description of what the bug is.

****To Reproduce****
Steps to reproduce the behavior:
1. Go to '...'
2. Click on '...'
3. Scroll down to '...'
4. See error

****Expected behavior****

Las plantillas de pull requests aseguran que cada pull request cumpla con los estándares y requisitos del proyecto. Estas plantillas generalmente incluyen listas de verificación, secciones para describir los cambios, hacer referencia a issues relevantes y cualquier nota adicional. Esta estandarización simplifica el proceso de revisión y mejora la calidad de las contribuciones.

Cada uno de estos componentes juega un papel vital en la conformación de un repositorio de GitHub bien organizado, accesible y amigable para los colaboradores. Al implementar esta documentación y plantillas base estándar, estableces una base sólida para la colaboración y la gestión de proyectos, resonando con las mejores prácticas en la cultura DevOps y de código abierto.