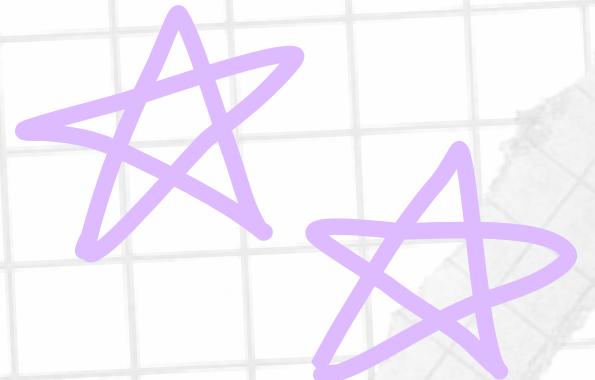


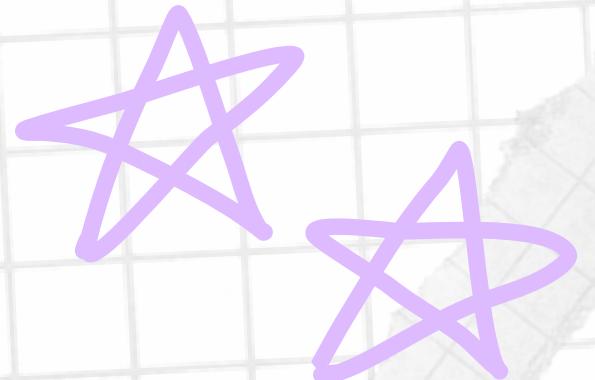
Propuesta
Línea base

Tecnologías

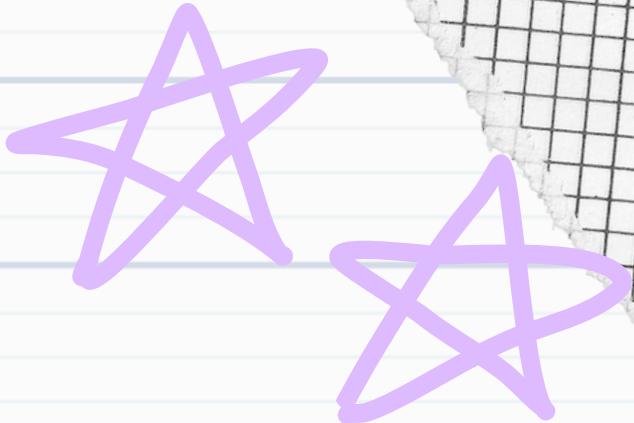


BASE DE DATOS

MySQL: amplia adopción, fácil acceso a soporte y documentación, open-source, compatible con múltiples plataformas y lenguajes, conocido por su rendimiento, seguridad robusta (autenticación por roles, encriptación, SSL).



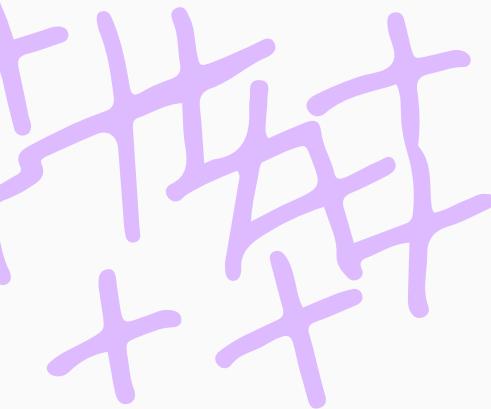
Front-End



ANGULAR: ARQUITECTURA BASADA EN COMPONENTES,
ESCALABILIDAD, USO DE TYPESCRIPT, MODULARIDAD QUE
FACILITA EL MANTENIMIENTO Y REUTILIZACIÓN DEL CÓDIGO.



ANGULAR

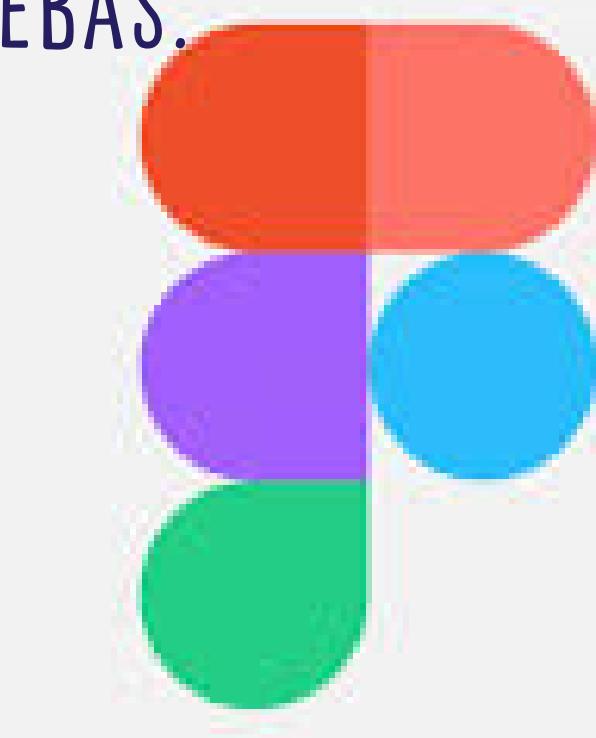


Back End

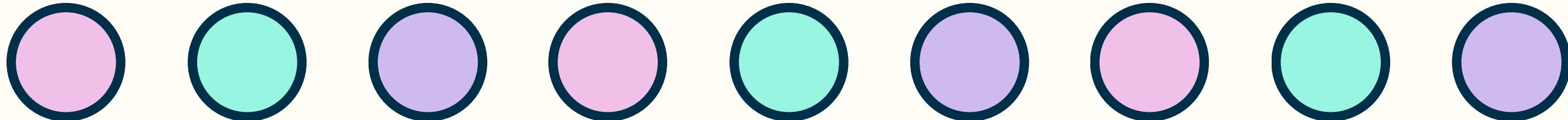
EXPRESS.JS PERMITE CONSTRUIR UN BACKEND ROBUSTO Y
ESCALABLE DE MANERA EFICIENTE, MIENTRAS QUE FIGMA Y
POSTMAN SON HERRAMIENTAS CLAVE PARA EL DISEÑO Y
PRUEBAS.



POSTMAN

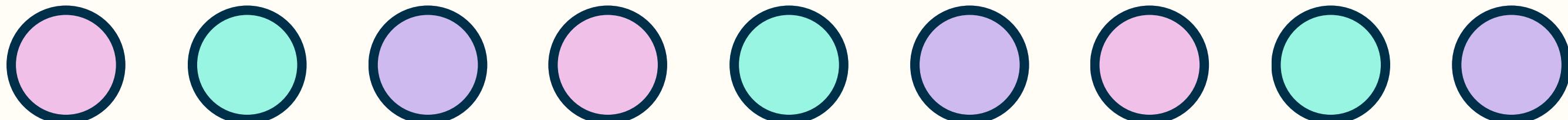


Figma



LENGUAJES DE PROGRAMACION

APIS (JSON), JAVASCRIPT (ANGULAR): JSON ES ESTÁNDAR PARA
INTERCAMBIO DE DATOS EN APIS. JAVASCRIPT PERMITE
INTEGRACIÓN EFICIENTE ENTRE FRONTEND Y BACKEND.



PARADIGMAS DE PROGRAMACION

- Orientado a Objetos: Facilita la representación de entidades del negocio (usuarios, dietas, etc.) y es aplicable en varios lenguajes como Python y Java.
- Programación Funcional: Útil para manejar funciones puras y reducir efectos secundarios, aplicable en JavaScript y Python.
- Programación Concurrente: Esencial en sistemas distribuidos o de alto rendimiento.

2

BUEAS PRACTICAS

- Front-end y Back-end: Seguir principios sólidos de arquitectura modular y reutilización de código.
- Nomenclatura de variables: Uso de Camel Case para mantener uniformidad.
- Manejo de Versionado: Uso de Git y GitHub para control de versiones y trabajo colaborativo.

3

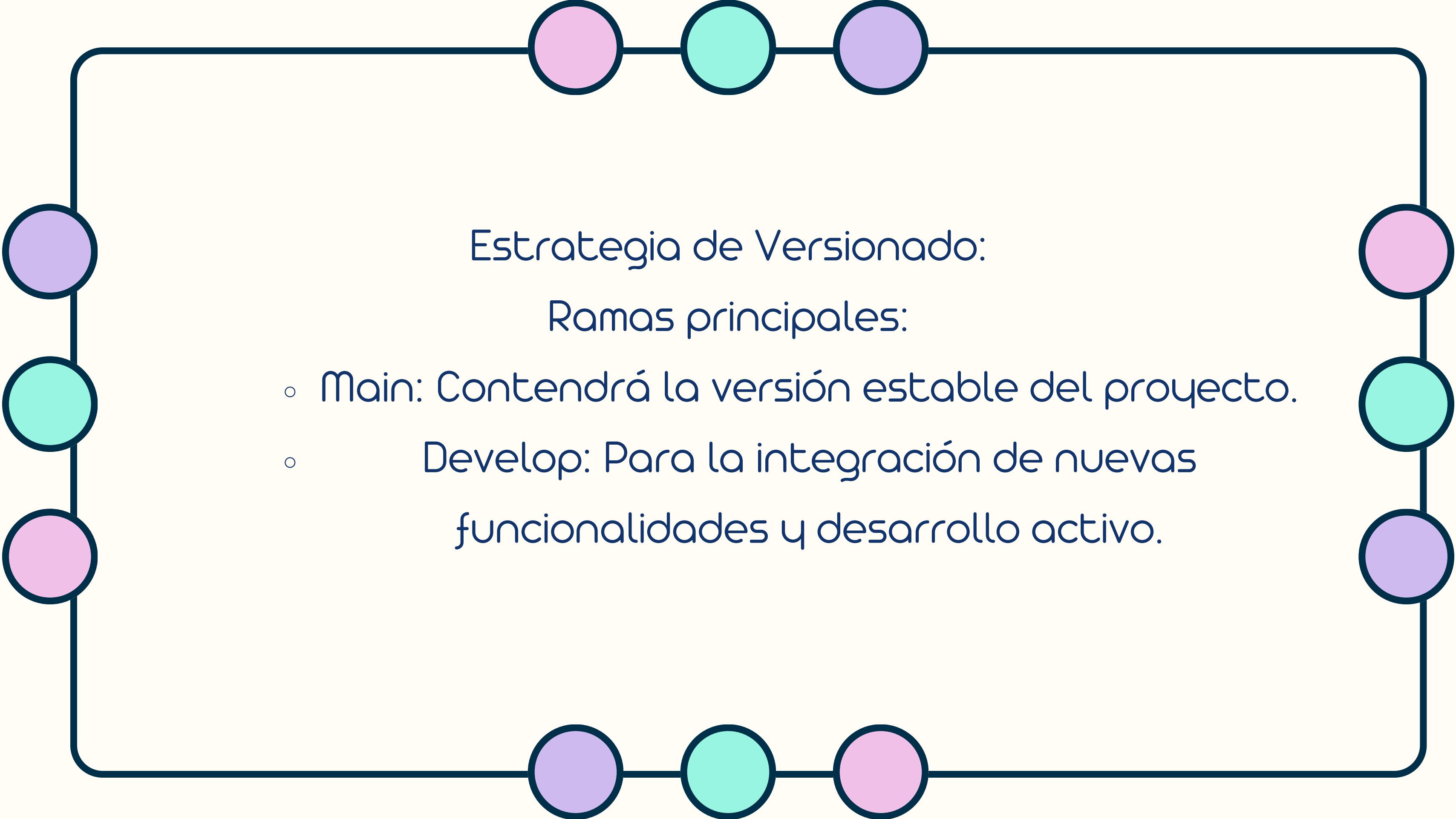
METODOLOGÍA

Scrum: Se propone Scrum por su enfoque en ciclos iterativos, mejora continua y adaptación a cambios rápidos. Es ideal para proyectos de desarrollo ágil con equipos multidisciplinarios.

4

MANEJO DE VERSIÓNADO

Utilizaremos Git como sistema de control de versiones y GitHub como plataforma de alojamiento de repositorios para garantizar un flujo de trabajo colaborativo y organizado.



Estrategia de Versionado:

Ramas principales:

- Main: Contendrá la versión estable del proyecto.
- Develop: Para la integración de nuevas funcionalidades y desarrollo activo.

Flujo de trabajo para el manejo de versionado (Git y GitHub)

Feature branches (Ramas de funcionalidades):

Cada nueva funcionalidad, mejora o cambio se desarrollará en su propia rama, creada a partir de la rama develop.

El nombre de estas ramas sigue un formato estándar: feature/nueva-funcionalidad.

Esto asegura que el trabajo en diferentes funcionalidades se mantenga separado, evitando conflictos y facilitando la colaboración en paralelo.

Ejemplo: Si un miembro del equipo está trabajando en el sistema de autenticación, creará la rama feature/autenticacion.





PULL REQUESTS:

UNA VEZ QUE LA FUNCIONALIDAD O MEJORA ESTÉ COMPLETA Y TESTEADA LOCALMENTE, SE ENVIARÁ UN PULL REQUEST (PR) PARA FUSIONAR LA RAMA FEATURE EN LA RAMA DEVELOP.

EL PULL REQUEST ES REVISADO POR OTRO MIEMBRO DEL EQUIPO, ASEGURANDO CALIDAD Y CONSISTENCIA EN EL CÓDIGO.

DURANTE LA REVISIÓN, SE VERIFICA QUE EL CÓDIGO SIGA LAS BUENAS PRÁCTICAS, NO ROMPA EL SISTEMA Y CUMPLA CON LOS REQUISITOS.

SOLO DESPUÉS DE LA APROBACIÓN DEL EQUIPO, SE PROCEDE A LA FUSIÓN.



- 1
- 2
- 3
- 4
- 5

RELEASE BRANCHES (RAMAS DE LANZAMIENTO):

CUANDO EL PROYECTO ESTÉ LISTO PARA UNA NUEVA VERSIÓN ESTABLE, SE CREARÁ UNA RAMA RELEASE DESDE DEVELOP. EN ESTA RAMA, SE REALIZARÁN PRUEBAS FINALES Y PEQUEÑOS AJUSTES NECESARIOS ANTES DEL LANZAMIENTO OFICIAL. UNA VEZ QUE LA RAMA RELEASE ESTÉ APROBADA, SE FUSIONA EN MAIN Y SE MARCA UNA NUEVA VERSIÓN DEL PROYECTO.

HOTFIXES (CORRECCIONES URGENTES):

SI SURGE UN PROBLEMA CRÍTICO EN PRODUCCIÓN, SE CREA UNA RAMA HOTFIX DESDE MAIN PARA RESOLVER EL PROBLEMA RÁPIDAMENTE.

UNA VEZ CORREGIDO EL ERROR, LA RAMA HOTFIX SE FUSIONA TANTO EN MAIN COMO EN DEVELOP PARA MANTENER AMBAS RAMAS ACTUALIZADAS CON LA CORRECCIÓN.

ESTO GARANTIZA QUE LA VERSIÓN EN PRODUCCIÓN SE MANTENGA ESTABLE, MIENTRAS QUE LA RAMA DE DESARROLLO TAMBIÉN RECIBE EL ARREGLO.

