

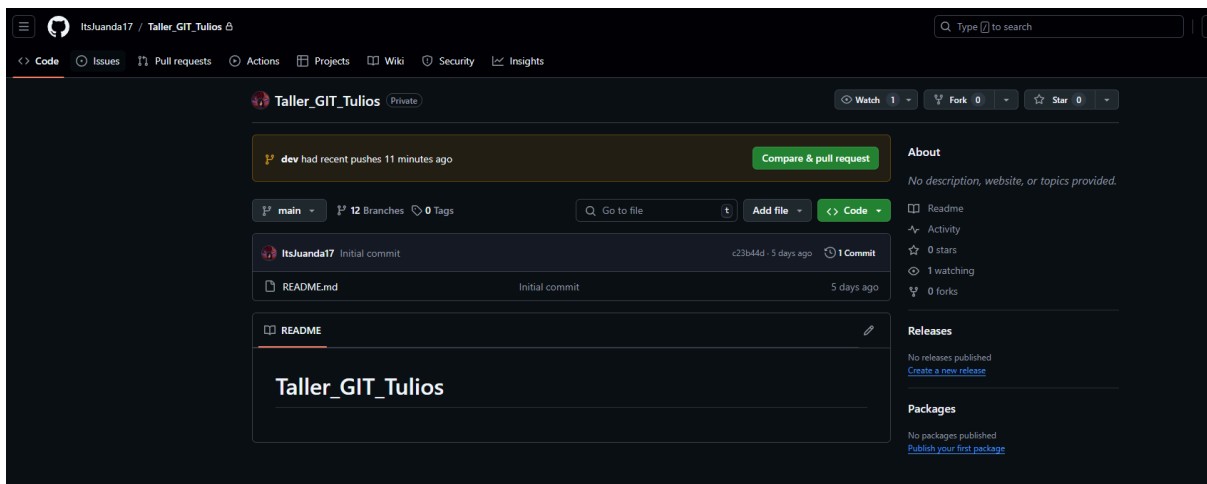
# TALLER DE GIT 1

Usuario A: Juan Pablo Ordoñez  
Usuario B: Juan Camilo Muñoz  
Usuario C: Juan David Acevedo  
Usuario D: Manuel Cardona  
Usuario E: Alejandro Quiñones

Enlace al repositorio: [https://github.com/ItsJuanda17/Taller\\_GIT\\_Tulios.git](https://github.com/ItsJuanda17/Taller_GIT_Tulios.git)

## Parte 1: Configuración Inicial

### 1. Repositorio de Github:



### 2. Git clone Juan Camilo Muñoz

```
MINGW64:/c:/Users/juanc/OneDrive/Escritorio/Universidad/Quinto Semestr...
juanc@LAPTOP-3Q2E57HJ MINGW64 ~/OneDrive/Escritorio/Universidad/Quinto Semestre/
Proyecto Integrador (master)
$ git clone https://github.com/ItsJuanda17/Taller_GIT_Tulios.git
Cloning into 'Taller_GIT_Tulios'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

### 3. Git clone Manuel Cardona

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. <https://aka.ms/PSWindows>

```
PS C:\Users\manue\OneDrive\Escritorio> git clone https://github.com/ItsJuanda17/Taller_GIT_Tulios.git
Cloning into 'Taller_GIT_Tulios'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\manue\OneDrive\Escritorio> |
```

#### 4. Git clone Juan Ordoñez:

```
PS C:\Users\juand\OneDrive\Escritorio\PROYECTO INTEGRADOR1\taller github> git clone https://github.com/ItsJuanda17/Taller_GIT_Tulios.git
Cloning into 'Taller_GIT_Tulios'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\juand\OneDrive\Escritorio\PROYECTO INTEGRADOR1\taller github> |
```

#### 5. Git clone Juan Acevedo:

```
Juan David Acevedo G@DESKTOP-BEH9TNU MINGW64 ~/OneDrive/Documentos/Taller Git
$ git clone https://github.com/ItsJuanda17/Taller_GIT_Tulios.git
Cloning into 'Taller_GIT_Tulios'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

#### 6. Git clone Alejandro Q.C

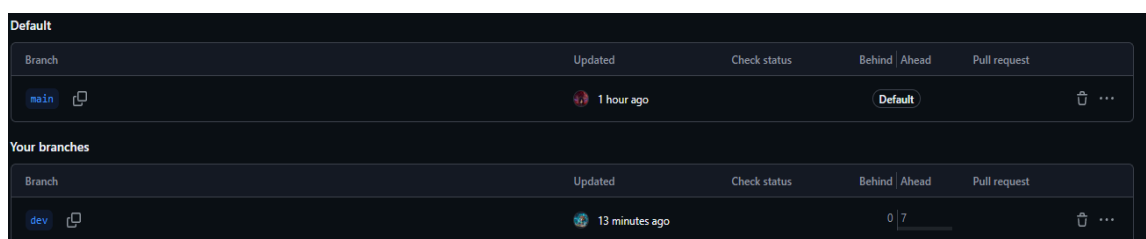
```
Cloning into 'Taller_GIT_Tulios' ...
remote: Enumerating objects: 67, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 67 (delta 8), reused 51 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (67/67), 11.85 KiB | 2.37 MiB/s, done.
Resolving deltas: 100% (8/8), done.
```

7. **Pregunta para reflexión:** ¿Cuál es la diferencia entre clonar un repositorio y hacer un fork?  
¿En qué situaciones utilizarías cada uno?

Clonar un repositorio crea una copia exacta en tu máquina local, ideal cuando tienes permisos para trabajar directamente en el proyecto original. Hacer un fork crea una copia del repositorio en tu cuenta, útil para trabajar de manera independiente en un proyecto, especialmente si no tienes permisos en el repositorio original. Usa clonar si eres parte del equipo y tienes acceso, y fork si quieres hacer contribuciones o experimentar sin afectar el proyecto original.

## Parte 2: Colaboración en Equipo usando Ramas

1. Captura de pantalla de las ramas para Parte 1: Creación de estructura inicial:



## 2. Captura de pantalla de las ramas para Parte 2: Nuevas Funcionalidades

Branch	Updated	Check status	Behind   Ahead	Pull request
<a href="#">main</a>	5 days ago		Default	...
<a href="#">dev</a>	9 minutes ago		0   60	...
<a href="#">feat/Imprimir-Nuevos-Atributos</a>	29 minutes ago		0   58	#10  ...
<a href="#">feature/agregar-parametro-filtro</a>	35 minutes ago		0   54	#6  ...
<a href="#">feature/potencia-vehiculo</a>	53 minutes ago		0   48	#8  ...
<a href="#">feat/main-vehicle-filter-year-range</a>	1 hour ago		0   37	#9  ...
<a href="#">feature/Atributo-color-en-vehiculo</a>	1 hour ago		0   44	#7  ...
<a href="#">feat/Imprimir-Vehiculos</a>	2 hours ago		0   32	#5  ...
<a href="#">feature/vehiculo-validaciones-combustible</a>	2 hours ago		0   24	#2  ...
<a href="#">feature/Lista-De-Vehiculos</a>	3 hours ago		0   13	#4  ...
<a href="#">feat/create-vehicle-class</a>	5 days ago		0   10	#3  ...
<a href="#">feat/HistorialMantenimiento</a>	5 days ago		0   5	#1  ...

## 3. Readme:

[https://github.com/ItsJuanda17/Taller\\_GIT\\_Tulios/blob/dev/README.md](https://github.com/ItsJuanda17/Taller_GIT_Tulios/blob/dev/README.md)

### Sistema de Gestión de Vehículos

---

#### Descripción del Proyecto

Este proyecto es un sistema de gestión de vehículos desarrollado como parte del curso de Ingeniería de Sistemas en la Facultad de Ingeniería, Diseño y Ciencias Aplicadas. El objetivo principal es gestionar información relevante de una flota de vehículos, incluyendo datos técnicos, historial de mantenimiento y otras características específicas.

El proyecto está diseñado para ser colaborativo, utilizando Git y GitHub para el control de versiones. Cada miembro del equipo es responsable de desarrollar y agregar funcionalidades específicas al sistema, lo que incluye la creación de clases y métodos para manejar la información de los vehículos y sus mantenimientos.

#### Integrantes del Equipo

- Usuario A: Juan pablo Ordoñez
- Usuario B: Juan Camilo Muñoz Barco
- Usuario C: Juan David Acevedo
- Usuario D: Jose Manuel Cardona
- Usuario E: Alejandro Quiñones

#### Funcionalidades Principales

- Usuario A
  - Añadir implementación de la clase Vehicle: se implementó la clase Vehicle incluyendo los atributos "brand", "model", "year", "mileage", "current\_status", "fuel\_type" junto con sus respectivos métodos getter y setter.
  - Filtro por rango de años: se añadió la función "search\_by\_year\_range" para que el sistema pueda buscar vehículos dentro de un rango de años. Ejemplo: `search_by_year_range: * main.search_by_year_range(2018, 2016)`
- Usuario B
  - Historial de Mantenimiento: Almacena la información sobre las reparaciones y mantenimientos realizados, como la fecha, descripción del servicio, kilometraje, costo, y nombre del mecánico.
  - Filtrar Vehículos por Año en Orden: Implementa un método para filtrar los vehículos por año, permitiendo obtener una lista de vehículos que cumplan con un año específico. Además, se ordena la lista de vehículos por año de forma ascendente o descendente. Ejemplo: `main.search_by_year(2015, "mayor")`
- Usuario C
  - Lista de vehículos: Implementará la clase "Main", que será el punto central de interacción del sistema, permitiendo gestionar una lista de vehículos. A través de esta clase, se podrán añadir vehículos a la lista y buscar vehículos por año
  - Cambios Adicionales en la Clase Vehículo: Modifica la clase "Vehículo", para agregar un nuevo atributo "color". Agrega los getter y setter pertinentes
- Usuario D
  - Validaciones Adicionales para el Tipo de Combustible en la Clase Vehículo: Implementa validaciones adicionales en la clase Vehículo, asegurando que el tipo de combustible solo pueda ser de una lista predefinida (por ejemplo, "Gasolina", "Diesel", "Eléctrico").
  - Agregar un nuevo atributo "potencia": Implementa un nuevo atributo llamado "potencia" en la clase Vehicle.py con sus getters y setters pertinentes.

- Usuario E
  - Imprimir Vehículos con sus características: Implementa método para la impresión de los vehículos registrados en el sistema, mostrando información relevante de cada uno de ellos, incluyendo la marca, el modelo, el año, el tipo de combustible, potencia y color.

### Instrucciones de Configuración

1. Clonar el repositorio:

```
git clone <URL-del-repositorio>
```
2. Navegar al directorio del proyecto:

```
cd nombre-del-proyecto
```

### Estructura del Proyecto

- code/: Contiene el código fuente del proyecto, incluyendo las clases y métodos desarrollados.
- README.md: Este archivo, que incluye la descripción del proyecto y las instrucciones necesarias.
- CODESTYLE.md: Documento que especifica las reglas de estilo de codificación acordadas por el equipo y el estandar de commits.
- Informe.pdf: Documento que detalla el proceso de desarrollo, incluyendo evidencia de commits, merges y resolución de conflictos.

### Requisitos del Sistema

- Lenguaje de Programación: Python

4. **Pregunta para reflexión:** ¿Por qué es importante seguir una convención para nombrar las ramas? ¿Qué beneficios tiene en un equipo grande?

Seguir una convención para nombrar las ramas es esencial en proyectos colaborativos, especialmente en equipos grandes, porque facilita la comunicación, organiza el código de manera clara, y permite una mejor gestión del proyecto. Los beneficios incluyen una comunicación más efectiva entre los miembros, la evitación de conflictos y errores, la posibilidad de automatizar tareas basadas en nombres de ramas, y la creación de un historial de cambios más legible y fácil de auditar. Además, ayuda a mantener la estructura del proyecto a medida que este crece en complejidad y escala.

### Parte 3: Gestión de Commits y Estándares de Codificación

1. Captura de pantalla del contenido del CODE STYLE:
  - a. Estándar de los Commit

### Commit Standard

#### Formato del Mensaje de Commit

Un buen mensaje de commit debe seguir el siguiente formato:

```
<type>[optional scope]: <description>

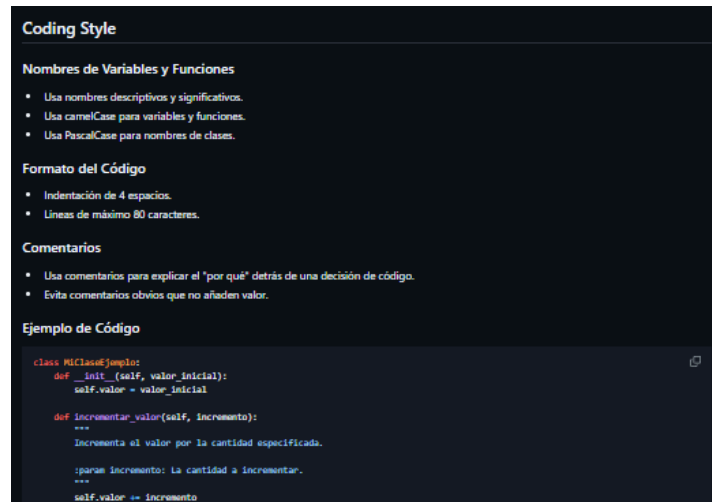
[optional body]

[optional footer(s)]
```

#### Tipos de Commit

- feat: Una nueva característica.
- fix: Una corrección de errores.
- docs: Cambios en la documentación.
- style: Cambios que no afectan el significado del código (espacios en blanco, formato, etc.).
- refactor: Cambios en el código que no corrigen errores ni añaden características.
- test: Añadir o corregir tests.
- chore: Cambios en el proceso de construcción o herramientas auxiliares y librerías.

b. Estilo de Código:



2. Captura de pantalla de los commits:

doc : Especificación de funcionalidad en el README JManuel2004 committed 1 hour ago	db7fac5		<>	...
feat: agregar atributo 'potencia' JManuel2004 committed 1 hour ago	44386a9		<>	...
feat: se implemento un metodo que permite buscar vehiculos dentro de un rango de años jpo08 committed 1 hour ago	7a3a64b		<>	...
feat: se añade nuevo atributo color en la clase vehiculo ItsJuanda17 committed 1 hour ago	4f14329		<>	...
feat: se añade nuevo atributo color en la clase vehiculo ItsJuanda17 committed 2 hours ago	f9d972b		<>	...
docs: explicacion de la nueva funcionalidad en el README JuanCamiloMunozB committed 2 hours ago	84c48bf		<>	...
feat: adición de parametro a funcion de filtro para ordenar de mayor o menor JuanCamiloMunozB committed 2 hours ago	6fea1c0		<>	...

3. **Pregunta Reflexiva:** ¿Qué diferencia hay entre un commit estándar y uno amend? ¿Cuándo usarías cada uno?

Un commit estándar registra un nuevo cambio en el historial de Git, mientras que un commit con --amend modifica el último commit, permitiendo corregir el mensaje o agregar cambios olvidados. Usas un commit estándar para registrar cambios nuevos y --amend para ajustar el commit más reciente sin crear uno nuevo.

## Parte 4: Merge y Resolución de Conflictos







### Explicación general de la manera en que se resolvieron los conflictos:

1. Primero, se realizan cambios en las ramas de manera local y se realizan commits para guardar estos cambios.
2. Al intentar fusionar las ramas (mediante **merge**), Git puede detectar conflictos.

3.Una vez detectados los conflictos, se deben abrir los archivos en el editor de texto que se esté utilizando. En estos archivos, Git marcará las secciones en conflicto utilizando delimitadores. A continuación, se debe editar el archivo para resolver los conflictos, eligiendo el código que se desea conservar o combinando ambos cambios según sea necesario.

4.Después de resolver los conflictos, se deben guardar los archivos y marcar los archivos como resueltos.

## 1. Pull Request para Parte 1: Creación de Estructura inicial

<input type="checkbox"/>	 0 Open	<input checked="" type="checkbox"/> 5 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Reviews ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>		<b>Feat/imprimir vehiculos</b>	#5 by Legnatbird was merged 1 minute ago • Approved						2
<input type="checkbox"/>		<b>Feature/lista de vehiculos</b>	#4 by ItsJuanda17 was merged 47 minutes ago • Approved						2
<input type="checkbox"/>		<b>Feat/create vehicle class</b>	#3 by jpo08 was merged 23 minutes ago • Approved						2
<input type="checkbox"/>		<b>Feature/vehiculo validaciones combustible</b>	#2 by JManuel2004 was merged 14 minutes ago • Approved						2
<input type="checkbox"/>		<b>feat: adicion de Historial de Mantenimiento</b>	#1 by JuanCamilMunozB was merged 12 minutes ago • Approved						1

## 2. Pull Requests de las primeras 5 funcionalidades a la rama dev

Feature/lista de vehiculos #4

Open

ItsJuanda17 wants to merge 6 commits into dev from feature/lista-de-vehiculos

Conversation 2

Commits 6

Checks 0

Files changed 3

ItsJuanda17 commented 5 days ago

Este Pull Request introduce la implementación de la clase Main, que actúa como el punto central para gestionar una lista de vehículos. La clase proporciona las siguientes funcionalidades:

Gestión de Vehículos:

Añadir Vehículos: Permite agregar nuevos vehículos a la lista. Cada vehículo está representado por la clase Vehicle, que incluye atributos para la marca (make), el modelo (model) y el año (year).

Búsqueda de Vehículos:

Buscar por Año: Permite buscar vehículos en la lista que coincidan con un año específico. Esta funcionalidad devuelve una lista de vehículos que tienen el año proporcionado.

ItsJuanda17 added 4 commits 5 days ago

Creacion rama dev 5027452

Merge branch 'dev' of https://github.com/ItsJuanda17/taller\_GIT\_Tuulos a84311c

feat: edición de lista de vehiculos 9d373bd

funcionalidad: lista de vehiculos 9c6840f

JuanCamiloMunoz8 approved these changes 32 minutes ago

View reviewed changes

JuanCamiloMunoz8 left a comment

aprobado

Jpo08 approved these changes 16 minutes ago

View reviewed changes

Feat/create vehicle class #3

Open

jpo08 wants to merge 3 commits into dev from feat/create-vehicle-class

Conversation 2

Commits 3

Checks 0

Files changed 1

jpo08 commented 5 days ago

Añadir implementación de la clase Vehicle:

en este Pull Request se implemento la clase Vehicle incluyendo los atributos "brand", "model", "year", "mileage", "current\_status", "fuel\_type" junto con sus respectivos métodos getter y setter.

jpo08 added 3 commits 5 days ago

Feat: Implementation of the vehicle class with getters and setters 8ae170a

Feat: Implementation of the vehicle class with getters and setters 54a50fd

Merge remote-tracking branch 'origin/dev' into feat/create-vehicle-class 8f5d585

JuanCamiloMunoz8 approved these changes 28 minutes ago

View reviewed changes

JuanCamiloMunoz8 left a comment

se ve bien

ItsJuanda17 approved these changes 25 minutes ago

View reviewed changes

ItsJuanda17 left a comment

Completo

Feature/vehiculo validaciones combustible #2

Open

JManuel2004 wants to merge 3 commits into dev from feature/vehiculo-validaciones-combustible

Conversation 2

Commits 3

Checks 0

Files changed 3

JManuel2004 commented 5 days ago

Se implementaron validaciones en la clase Vehicle para asegurar que el atributo fuel\_type solo acepte valores de una lista predefinida: "Gasoline", "Diesel" o "Electric". La validación se realiza en el método validate\_fuel, y en el setter set\_fuel\_type, lanzando un ValueError si el tipo de combustible proporcionado no está en la lista permitida.

JManuel2004 added 3 commits 5 days ago

Añadida validación de tipo de combustible en la clase Vehicle b6b988a

eliminar carpeta src f583f5d

Cambio de carpeta CODESTYLE ffc28fa

JuanCamiloMunoz8 approved these changes 5 days ago

View reviewed changes

JuanCamiloMunoz8 left a comment

melo, cumple con los requisitos

ItsJuanda17 approved these changes 5 days ago

View reviewed changes

ItsJuanda17 left a comment

melo, me gusto

feat: adición de Historial de Mantenimiento #1

Open

JuanCamiloMunoz8 wants to merge 1 commit into dev from feat/HistorialMantenimiento

Conversation 1

Commits 1

Checks 0

Files changed 2

JuanCamiloMunoz8 commented 5 days ago

Este pull request incorpora la funcionalidad de gestión del Historial de Mantenimiento en la rama feature/historial-mantenimiento a la rama develop. Se ha implementado la clase HistorialMantenimiento, que permite almacenar y gestionar información sobre las reparaciones y mantenimientos realizados a los vehículos.

Cambios principales:

- Creación de la clase HistorialMantenimiento con los siguientes atributos: fecha, descripción\_servicio, kilometraje\_en\_servicio, costo, y nombre\_mecanico.
- Implementación de métodos getter y setter para cada uno de los atributos.
- Actualización del README.md con detalles sobre el uso y la estructura de la nueva funcionalidad.

feat: edición de Historial de Mantenimiento a8c9fd5

ItsJuanda17 approved these changes 31 minutes ago

View reviewed changes

ItsJuanda17 left a comment

Completo

Jpo08 approved these changes 17 minutes ago

View reviewed changes

Changes approved

2 approving reviews [Learn more about pull request reviews.](#)

Show all reviewers

2 approvals

Feat/imprimir vehiculos #5

Open

Legnatbird wants to merge 21 commits into main from feat/Imprimir-Vehiculos

Conversation 1

Commits 21

Checks 0

Files changed 5

Legnatbird commented 2 minutes ago

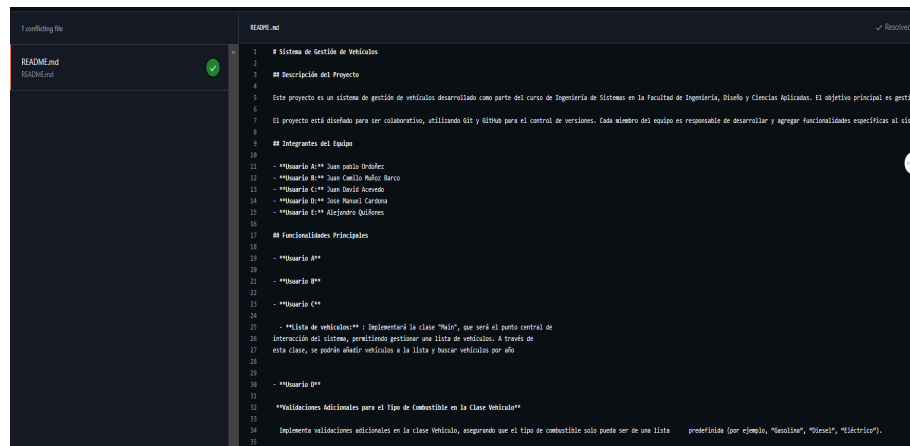
Este Pull Request introduce la el método de impresión de todos los vehículos en la clase Main.

Imprimir Vehículos

- Itera la lista vehicles de la clase e imprime la información relevante

ItsJuanda17 and others added 21 commits 5 days ago

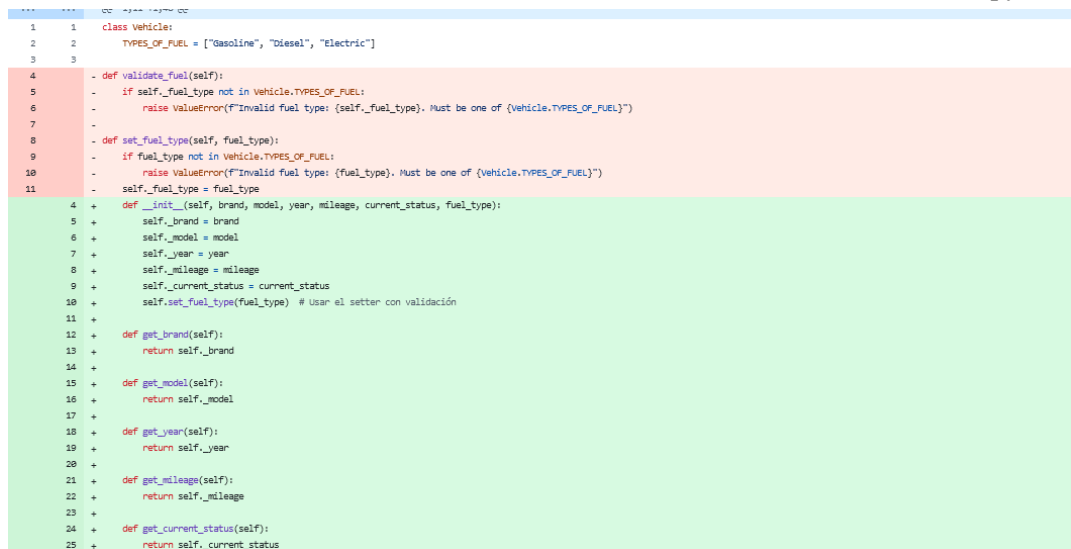
### 3. Feature/lista de vehículos: Resolución de conflictos en el README



The screenshot shows a code editor with a dark theme. On the left, a sidebar indicates a conflict in the README.md file. The main editor area shows the content of README.md with line numbers 1 through 29. The text describes a vehicle management system project. A conflict resolution overlay is visible on the right side of the editor, showing a green checkmark and the word 'Resolved'.

```
1 # Sistema de Gestión de Vehículos
2
3 ## Descripción del Proyecto
4
5 Este proyecto es un sistema de gestión de vehículos desarrollado como parte del curso de Ingeniería de Sistemas en la Facultad de Ingeniería, Biología y Ciencias Aplicadas. El objetivo principal es gestión
6
7 El proyecto está diseñado para ser colaborativo, utilizando Git y GitHub para el control de versiones. Cada miembro del equipo es responsable de desarrollar y agregar funcionalidades específicas al sistema
8
9 ## Integrantes del Equipo
10
11 - **Nombre A:** Juan Pablo Ordoñez
12 - **Nombre B:** Juan Camilo Muñoz Barco
13 - **Nombre C:** Juan David Acosta
14 - **Nombre D:** Juan Manuel Carmona
15 - **Nombre E:** Alejandro Quiroga
16
17 ## Funcionalidades Principales
18
19 - **Nombre A:**
20
21 - **Nombre B:**
22
23 - **Nombre C:**
24
25 - **Lista de vehículos:** Implementaré la clase 'Vehículo', que será el punto central de
26 interacción del sistema, permitiendo gestionar una lista de vehículos. A través de
27 esta clase, se podrán añadir vehículos a la lista y buscar vehículos por año
28
29 - **Nombre D:**
30
31 ## Validaciones Adicionales para el Tipo de Combustible en la Clase Vehículo
32
33 Implementa validaciones adicionales en la clase Vehículo, asegurando que el tipo de combustible solo pueda ser de una lista predefinida (por ejemplo, "Gasoline", "Diesel", "Electric").
34
35
```

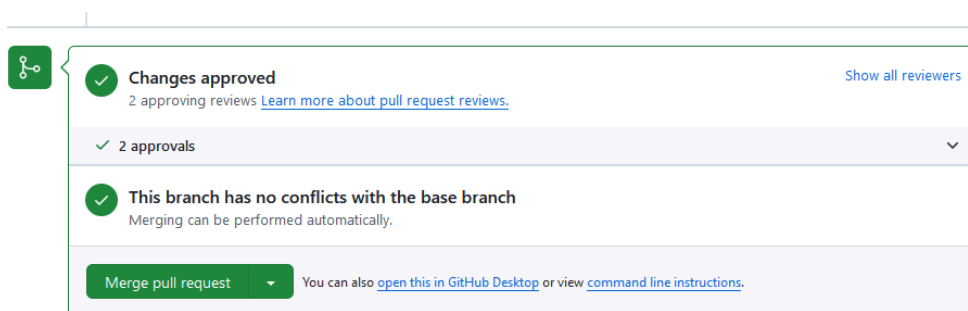
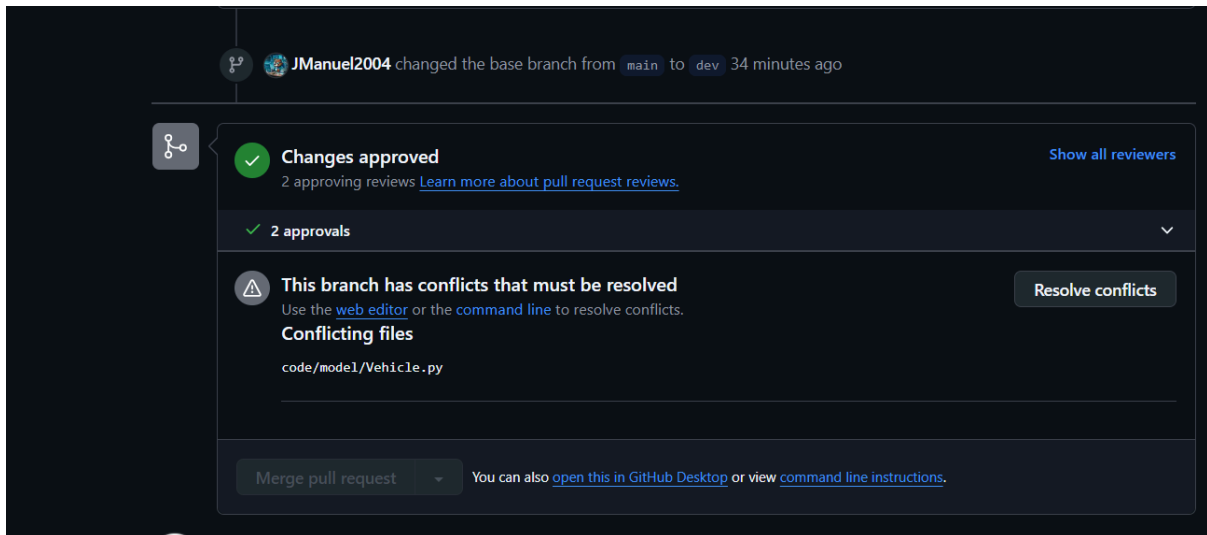
### 4. Feature/vehiculo validaciones combustible : Resolucion de conflictos en Vehicle.py



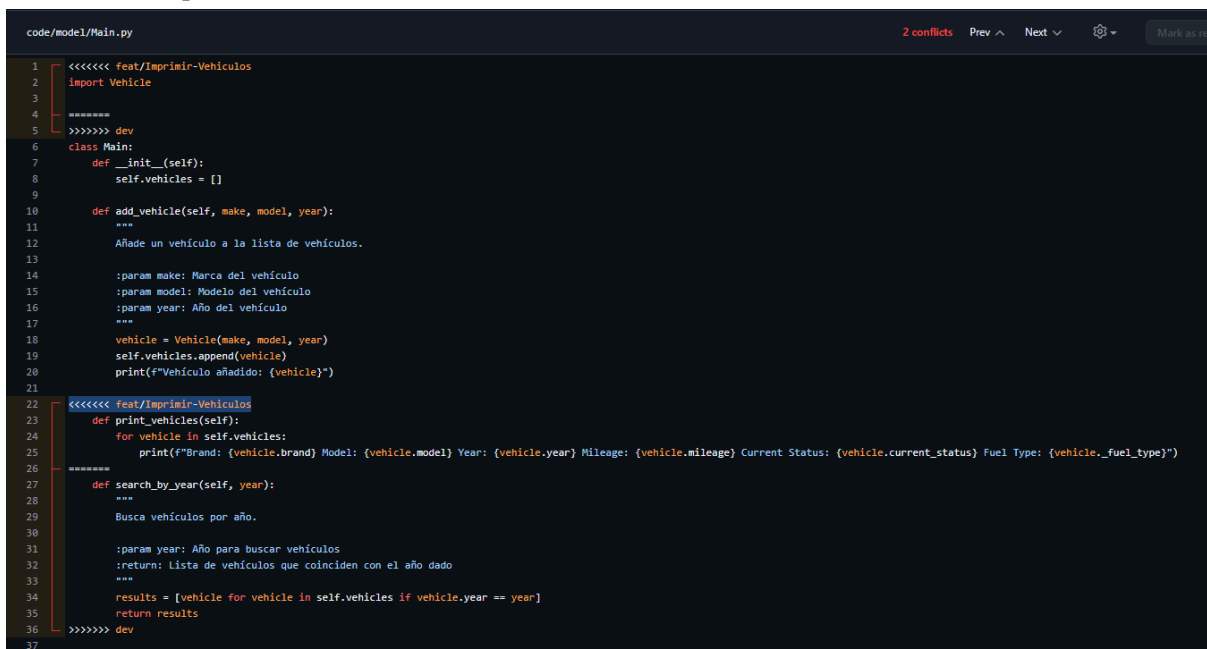
The screenshot shows a code editor with a dark theme. The main editor area shows the content of Vehicle.py with line numbers 1 through 25. The code defines a Vehicle class with attributes like brand, model, year, mileage, current\_status, and fuel\_type. It includes methods for validation, setting, and getting these attributes. A conflict resolution overlay is visible on the right side of the editor, showing a green checkmark and the word 'Resolved'.

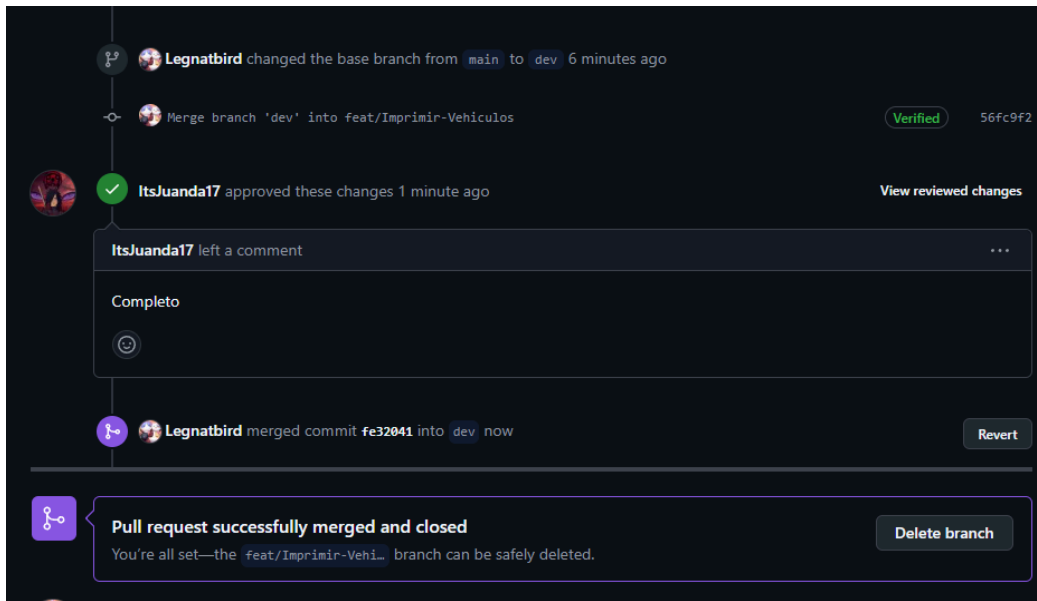
```
1 class Vehicle:
2     TYPES_OF_FUEL = ["Gasoline", "Diesel", "Electric"]
3
4     def validate_fuel(self):
5         if self.fuel_type not in Vehicle.TYPES_OF_FUEL:
6             raise ValueError(f"Invalid fuel type: {self.fuel_type}. Must be one of {Vehicle.TYPES_OF_FUEL}")
7
8     def set_fuel_type(self, fuel_type):
9         if fuel_type not in Vehicle.TYPES_OF_FUEL:
10            raise ValueError(f"Invalid fuel type: {fuel_type}. Must be one of {Vehicle.TYPES_OF_FUEL}")
11            self.fuel_type = fuel_type
12
13     def __init__(self, brand, model, year, mileage, current_status, fuel_type):
14         self.brand = brand
15         self.model = model
16         self.year = year
17         self.mileage = mileage
18         self.current_status = current_status
19         self.set_fuel_type(fuel_type) # Usar el setter con validación
20
21     def get_brand(self):
22         return self.brand
23
24     def get_model(self):
25         return self.model
26
27     def get_year(self):
28         return self.year
29
30     def get_mileage(self):
31         return self.mileage
32
33     def get_current_status(self):
34         return self.current_status
35
```



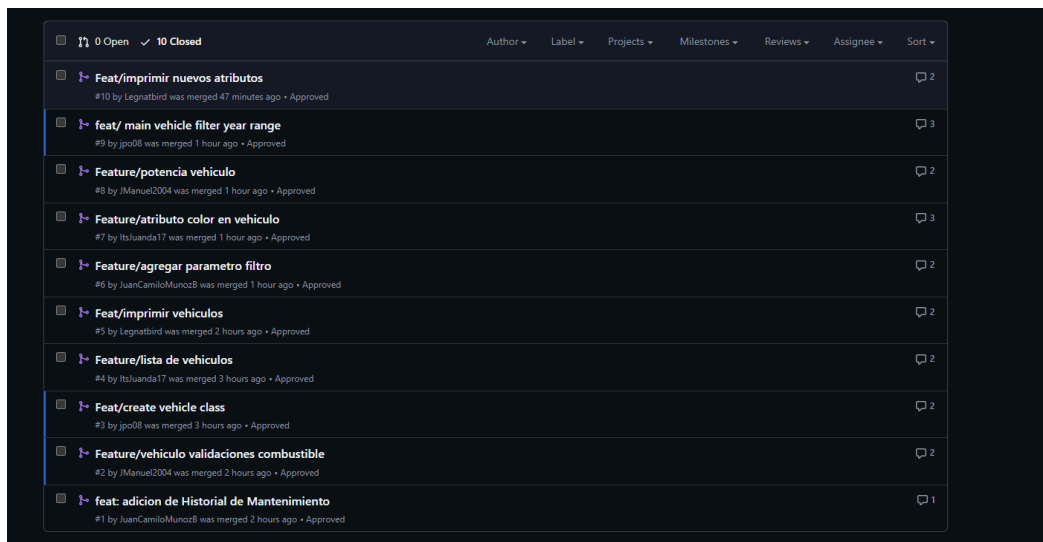


## 5. feat/Imprimir-Vehículos: conflictos

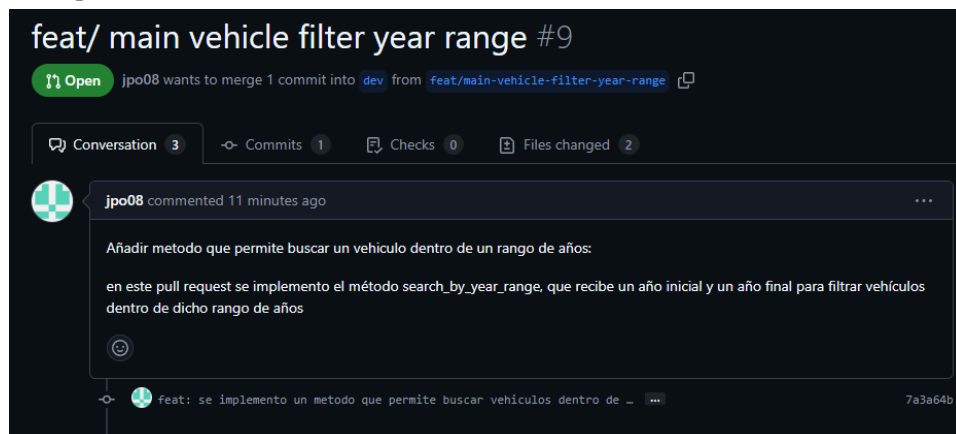




## 6. Pull Request para Parte 2: Nuevas Funcionalidades






## 7. Pull Requests de las 5 funcionalidades restantes a la rama dev



## Feature/atributo color en vehiculo #7

 Merged ItsJuanda17 merged 8 commits into `dev` from `feature/Atributo-color-en-vehiculo` 1 minute ago

 Conversation 3  Commits 8  Checks 0  Files changed 3




ItsJuanda17 commented 29 minutes ago

En esta actualización, se ha modificado la clase Vehide para incluir un nuevo atributo color.



## Feature/potencia vehiculo #8

 Open JManuel2004 wants to merge 2 commits into `dev` from `feature/potencia-vehiculo`

 Conversation 2  Commits 2  Checks 0  Files changed 3



JManuel2004 commented 19 minutes ago

introduce un nuevo atributo power a la clase Vehicle, que representa la potencia del vehículo en caballos de fuerza. Se han añadido los métodos `get_power()` y `set_power(power)` para gestionar este atributo, con una validación incluida en `set_power` para asegurar que el valor no sea negativo. Además, el constructor de la clase ahora utiliza los setters para inicializar `power` y `fuel_type`, garantizando que se apliquen las validaciones correspondientes desde el momento de la creación del objeto. También se ha actualizado la documentación para reflejar el nuevo atributo.



## Feature/agregar parametro filtro #6

 Open JuanCamiloMuno... wants to merge 2 commits into `dev` from `feature/agregar-parametro-filtro`

 Conversation 2  Commits 2  Checks 0  Files changed 3



JuanCamiloMunoB commented 35 minutes ago

En esta rama `feature`, se realizaron las siguientes modificaciones:

### 1. Clase `Main`:


- Se actualizó el filtro por año para que ahora reciba un parámetro adicional que especifique si el filtro debe listar vehículos con años **mayores** o **menores** al año dado.
- Implementación de la lógica que permite filtrar adecuadamente según este parámetro.

### 2. `README.md`:

- Se actualizó el documento para incluir la descripción de la nueva funcionalidad del filtro por año.



## Feat/imprimir nuevos atributos #10

 Open Legnatbird wants to merge 3 commits into `dev` from `feat/Imprimir-Nuevos-Atributos`

 Conversation 2  Commits 3  Checks 0  Files changed 2



Legnatbird commented 21 minutes ago

Añadir los nuevos métodos para la impresión



Add new atr

7997c7b



Legnatbird changed the base branch from `main` to `dev` 16 minutes ago

## 8. Feature/potencia vehículo: conflictos

```
1  class Vehículo:
2      TIPOES_FUELE = ["Gasoline", "Diesel", "Electric"]
3
4  <<<<<< feature/potencia-vehículo
5      def __init__(self, brand, model, year, mileage, current_status, fuel_type, power):
6          =====
7      def __init__(self, brand, model, year, mileage, current_status, fuel_type, color):
8          >>>>>> dev
9          self.brand = brand
10         self.model = model
11         self.year = year
12         self.mileage = mileage
13         self.current_status = current_status
14
15     <<<<<< feature/potencia-vehículo
16         self.set_fuel_type(fuel_type) # Usar el setter para validación
17         self.set_power(power) # Usar el setter para validación
18
19     def get_power(self):
20         return self._power
21
22     =====
23     self.set_fuel_type(fuel_type) # Usar el setter con validación
24     self._color = color
25
26 >>>>>> dev
27
28     def get_brand(self):
29         return self._brand
```

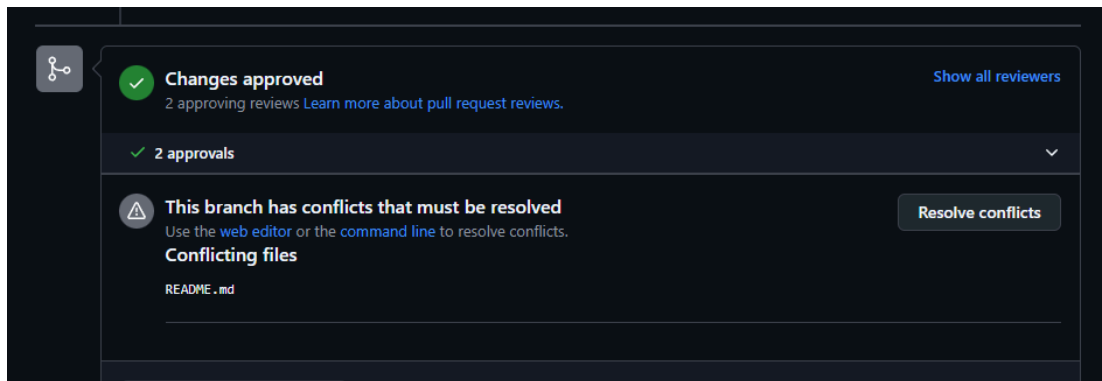
```
@@ -1,15 +1,15 @@
1  class Vehículo:
2      TIPOES_FUELE = ["Gasoline", "Diesel", "Electric"]
3
4  - def __init__(self, brand, model, year, mileage, current_status, fuel_type, color):
4  + def __init__(self, brand, model, year, mileage, current_status, fuel_type, power=None, color=None):
5      self.brand = brand
6      self.model = model
7      self.year = year
8      self.mileage = mileage
9      self.current_status = current_status
10
10 - self.set_fuel_type(fuel_type) # Usar el setter con validación
10 + self.set_fuel_type(fuel_type)
11 + self.set_power(power)
11 + self._color = color
12
13
14 + def get_power(self):
15 +     return self._power
16
17
17     def get_brand(self):
18         return self._brand
19
20
21 @@ -47,6 +50,11 @@ def set_mileage(self, mileage):
47  50     def set_current_status(self, current_status):
48  51         self.current_status = current_status
49  52
53 +     def set_power(self, power):
54 +         if power is not None and power < 0:
55 +             raise ValueError("Power cannot be a negative number.")
56 +         self._power = power
57 +
58
58     def set_fuel_type(self, fuel_type):
59         if fuel_type not in Vehículo.TIPOES_FUELE:
```

## 9. Feature/agregar parámetro filtro: conflictos

```
ulios (feature/agregar-parametro-filtro)
$ git merge origin/dev
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
CONFLICT (modify/delete): code/CODESTYLE.md deleted in HEAD and modified in origin/dev. Version origin/dev of code/CODESTYLE.md left in tree.
Auto-merging code/model/Main.py
Automatic merge failed; fix conflicts and then commit the result.
```

```
1 # Sistema de Gestión de Vehículos
2 ## Funcionalidades Principales
3 lista de vehículos que cumplan con un año específico. Además, se ordena la lista de vehículos por año de forma ascendente o
4 descendente.
5
6
7 - **Usuario C**
8
9 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
10 <<<<<< HEAD (Current Change)
11 - **Lista de vehículos:** : Implementará la clase "Main", que será el punto central de interacción del sistema, permitiendo
12 gestionar una lista de vehículos. A través de esta clase, se podrán añadir vehículos a la lista y buscar vehículos por año
13
14 =====
15 - **Lista de vehículos:** : Implementará la clase "Main", que será el punto central de
16 interacción del sistema, permitiendo gestionar una lista de vehículos. A través de
17 esta clase, se podrán añadir vehículos a la lista y buscar vehículos por año
18
19 - **Cambios Adicionales en la Clase Vehículo:** : Modifica la clase "Vehículo", para agregar un nuevo atributo "color". Agrega
20 los
21 getter y setter pertinentes
22
23 >>>>>> origin/dev (Incoming Change)
```

## 10. Feature/Imprimir-Nuevos-Atributos



## 11. Pull Request para subir los cambios de la rama dev a main

1 Open 10 Closed

Author Label Projects Milestones Reviews Assignee Sort

Merge dev into main #11 opened now by ItsJuanda17

### Merge dev into main #11

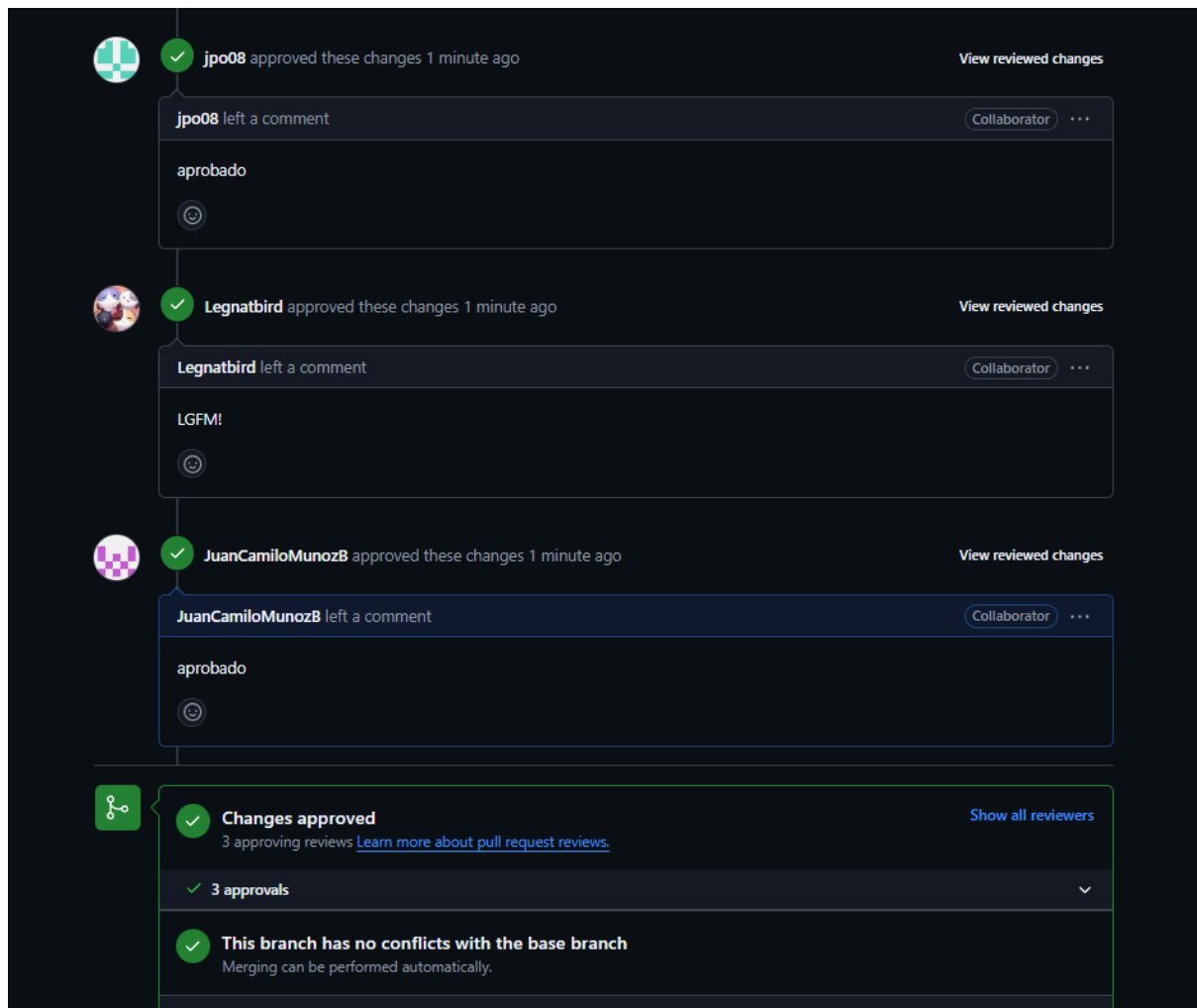
Open ItsJuanda17 wants to merge 63 commits into main from dev

Conversation 3 Commits 63 Checks 0 Files changed 5

ItsJuanda17 commented 1 minute ago

Owner

Este pull request propone la fusión de la rama dev en la rama main. La rama dev contiene una serie de cambios y mejoras que han sido desarrollados y probados. La integración de estos cambios en main es necesaria para que las nuevas características, correcciones y mejoras estén disponibles en la versión principal del proyecto.



12. **Pregunta para reflexión:** ¿Qué estrategias de resolución de conflictos podrías aplicar en un proyecto con múltiples colaboradores?

Algunas estrategias clave son:

1. **Comunicación clara:** Usar issues y pull requests para discutir cambios.
2. **Revisiones de código:** Implementar revisiones antes de aceptar cambios.
3. **Convenciones de codificación:** Establecer estándares para evitar discrepancias.
4. **Branching:** Trabajar en ramas separadas y hacer merge solo cuando esté listo.
5. **Uso de herramientas como CI/CD:** Automatizar pruebas para evitar errores al integrar cambios.

## Parte 5: Trabajo Final y Documentación del Proyecto

Para asegurarte de que la documentación sea útil:

1. **Claridad:** Escribir de forma concisa y fácil de entender.
2. **Actualización:** Mantener la documentación al día con los cambios del proyecto.
3. **Ejemplos:** Incluir ejemplos prácticos y casos de uso.
4. **Estructura:** Organizar el contenido de manera lógica y accesible.
5. **Contribución:** Instrucciones claras sobre cómo colaborar y seguir las normas del proyecto.