Introducción a GIT

El mejor amigo de un desarrollador

Por: Enrique Walter Philippeaux

Universidad Tecnológica Nacional
Facultad Regional Córdoba
IEEE Student Branch

Algo muy común...

Algo muy común...

¿Alguna vez les paso?

Algo muy común...

¿Alguna vez les paso?

1. Entrega

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2
4. Entrega finalok

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2
4. Entrega finalok
5. Entrega finalisima

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2
4. Entrega finalok
5. Entrega finalisima
6. Entrega finalahorasi

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2
4. Entrega finalok
5. Entrega finalisima
6. Entrega finalahorasi
7. Entrega FINAL

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2
4. Entrega finalok
5. Entrega finalisima
6. Entrega finalahorasi
7. Entrega FINAL
8. Entrega finalasd

Algo muy común...

¿Alguna vez les paso?

1. Entrega
2. Entrega Final
3. Entrega final2
4. Entrega finalok
5. Entrega finalisima
6. Entrega finalahorasi
7. Entrega FINAL
8. Entrega finalasd
9. EntregAAAAAAAAAA

¿Qué es GIT?

¿Qué es GIT?

- Git es un **Sistema de Control de Versiones**.

¿Qué es GIT?

- Git es un **Sistema de Control de Versiones**.
  - Facilita mantener múltiples versiones de un proyecto.

¿Qué es GIT?

- Git es un **Sistema de Control de Versiones**.
  - Facilita mantener múltiples versiones de un proyecto.
  - Facilita la interacción entre múltiples desarrolladores.

¿Qué es GIT?

- Git es un **Sistema de Control de Versiones**.
  - Facilita mantener múltiples versiones de un proyecto.
  - Facilita la interacción entre múltiples desarrolladores.
  - Previene la pérdida de información entre versiones.

¿Qué es GIT?

- Git es un **Sistema de Control de Versiones**.
  - Facilita mantener múltiples versiones de un proyecto.
  - Facilita la interacción entre múltiples desarrolladores.
  - Previene la pérdida de información entre versiones.
- Nos permite ver cambios que hicimos en nuestro proyecto, y revertirlos.

¿Qué es GIT?

- Git es un **Sistema de Control de Versiones**.
  - Facilita mantener múltiples versiones de un proyecto.
  - Facilita la interacción entre múltiples desarrolladores.
  - Previene la pérdida de información entre versiones.
- Nos permite ver cambios que hicimos en nuestro proyecto, y revertirlos.
- No es **GITHUB**.

¿Qué es GitHub?

¿Qué es GitHub?

- **Github.com** es un sitio web que **almacena** repositorios de git, en un **servidor remoto**.

GitHub

¿Qué es GitHub?

- **Github.com** es un sitio web que **almacena** repositorios de git, en un **servidor remoto**.
- Facilita compartir proyectos entre equipos.

GitHub

¿Qué es GitHub?

- **Github.com** es un sitio web que **almacena** repositorios de git, en un **servidor remoto**.
- Facilita compartir proyectos entre equipos.
- Proporciona una interfaz gráfica amigable

GitHub

¿Qué es GitHub?

- **Github.com** es un sitio web que **almacena** repositorios de git, en un **servidor remoto**.
- Facilita compartir proyectos entre equipos.
- Proporciona una interfaz gráfica amigable
- Es gratis.

GitHub

¿Donde utilizamos GIT?

¿Donde utilizamos GIT?

¿Donde utilizamos GIT?

- Códigos fuente

¿Donde utilizamos GIT?

- Códigos fuente
- Documentos

¿Donde utilizamos GIT?

- Códigos fuente
- Documentos
- Desarrollos electrónicos

¿Donde utilizamos GIT?

- Códigos fuente
- Documentos
- Desarrollos electrónicos
- Proyecto Final/Tesis

¿Cómo se configura?

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

- Nombre del autor

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

- Nombre del autor

```
$ git config –global user.name "Juan Perez"
```

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

- Nombre del autor
  ```
  $ git config –global user.name "Juan Perez"
  ```
- Email del autor

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

- Nombre del autor
  ```
  $ git config –global user.name "Juan Perez"
  ```
- Email del autor
  ```
  $ git config –global user.email "jp@git.com"
  ```

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

- Nombre del autor
  ```
  $ git config –global user.name "Juan Perez"
  ```
- Email del autor
  ```
  $ git config –global user.email "jp@git.com"
  ```
- Fecha y hora

¿Cómo se configura?

Nuestros **commits** llevan consigo los siguientes datos:

- Nombre del autor
  ```
  $ git config –global user.name "Juan Perez"
  ```
- Email del autor
  ```
  $ git config –global user.email "jp@git.com"
  ```
- Fecha y hora
  Este es automático..

Utilizando git

Utilizando git

- Creamos/Nos situamos en la carpeta de nuestro proyecto

Utilizando git

- Creamos/Nos situamos en la carpeta de nuestro proyecto

```
git init
```

Inicializamos el **repositorio** de git

Utilizando git

- Creamos/Nos situamos en la carpeta de nuestro proyecto

`git init`

Inicializamos el **repositorio** de git

`git clone url`

Si queremos trabajar sobre un **repositorio existente**, lo clonamos.

Utilizando git

- Agregamos y trabajamos sobre nuestros archivos

Utilizando git

- Agregamos y trabajamos sobre nuestros archivos

```
git add .
```

Añadimos los nuevos cambios

Utilizando git

- Agregamos y trabajamos sobre nuestros archivos

```
git add .
```

Añadimos los nuevos cambios

```
git commit -m "mensaje :)"
```

Creamos un **commit**

Utilizando git

Teniendo nuestro repositorio podemos vincularlo a **Github**.

- Creamos nuestro **repositorio** en **Github**

Utilizando git

Teniendo nuestro repositorio podemos vincularlo a **Github**.

- Creamos nuestro **repositorio** en **Github**
- Seguimos las instrucciones en **Github** para conectar nuestro **repositorio** a su **servidor remoto**.

Teniendo nuestro repositorio podemos vincularlo a **Github**.

- Creamos nuestro **repositorio** en **Github**
- Seguimos las instrucciones en **Github** para conectar nuestro **repositorio** a su **servidor remoto**.

```
git push origin
```

Subimos nuestros cambios al **servidor remoto**

Ramas/Branches

¿Qué es una rama?

Según *Atlassian*:

¿Qué es una rama?

Según *Atlassian*:

- Son parte del proceso de desarrollo diario.

¿Qué es una rama?

Según *Atlassian*:

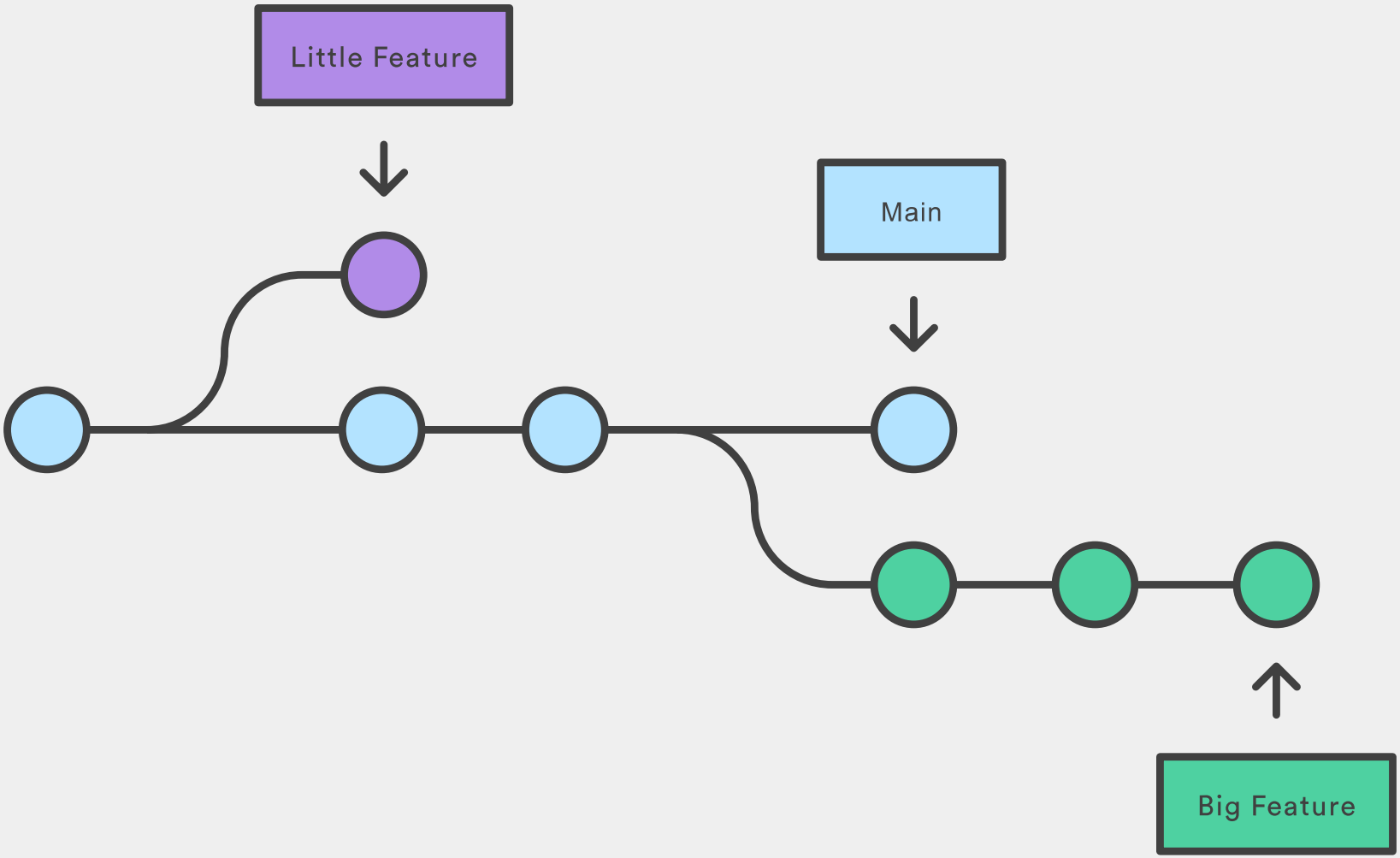
- Son parte del proceso de desarrollo diario.
- Son un puntero a las instantáneas de nuestros cambios.

¿Qué es una rama?

Según *Atlassian*:

- Son parte del proceso de desarrollo diario.
- Son un puntero a las instantáneas de nuestros cambios.
- Las utilizamos al agregar nuevas funciones, o solucionar errores.

Diagrama ejemplar



Comandos útiles

`git branch`

Enumera todas las ramas de tu repositorio

Comandos útiles

`git branch`

Enumera todas las ramas de tu repositorio

`git checkout <nombre>`

Nos **situa** en la rama llamada `<nombre>`

Comandos útiles

`git branch`

Enumera todas las ramas de tu repositorio

`git checkout <nombre>`

Nos **situa** en la rama llamada `<nombre>`

`git checkout -b <nueva>`

Crea una nueva rama llamada `<nueva>`

Comandos útiles

```
git branch
```

Enumera todas las ramas de tu repositorio

```
git checkout <nombre>
```

Nos **situa** en la rama llamada `<nombre>`

```
git checkout -b <nueva>
```

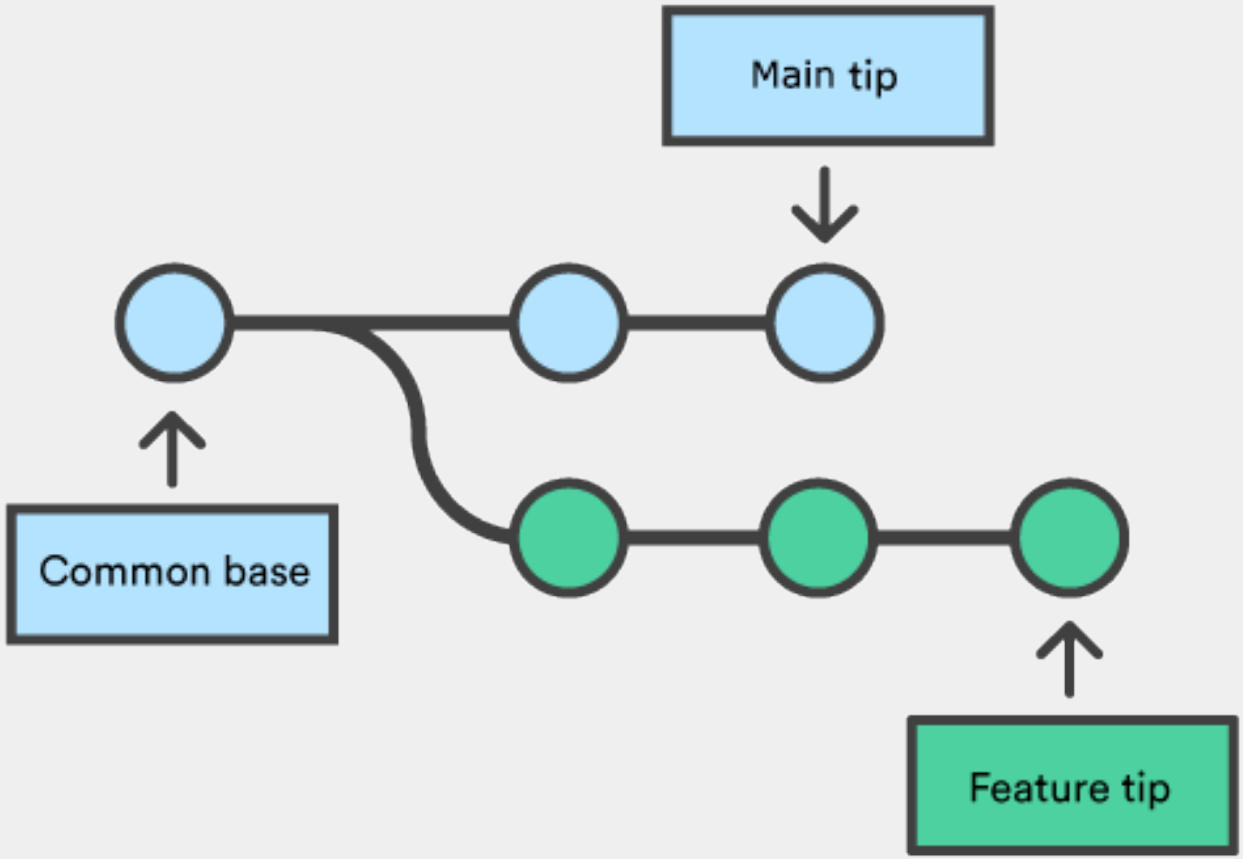
Crea una nueva rama llamada `<nueva>`

```
git branch -D <test>
```

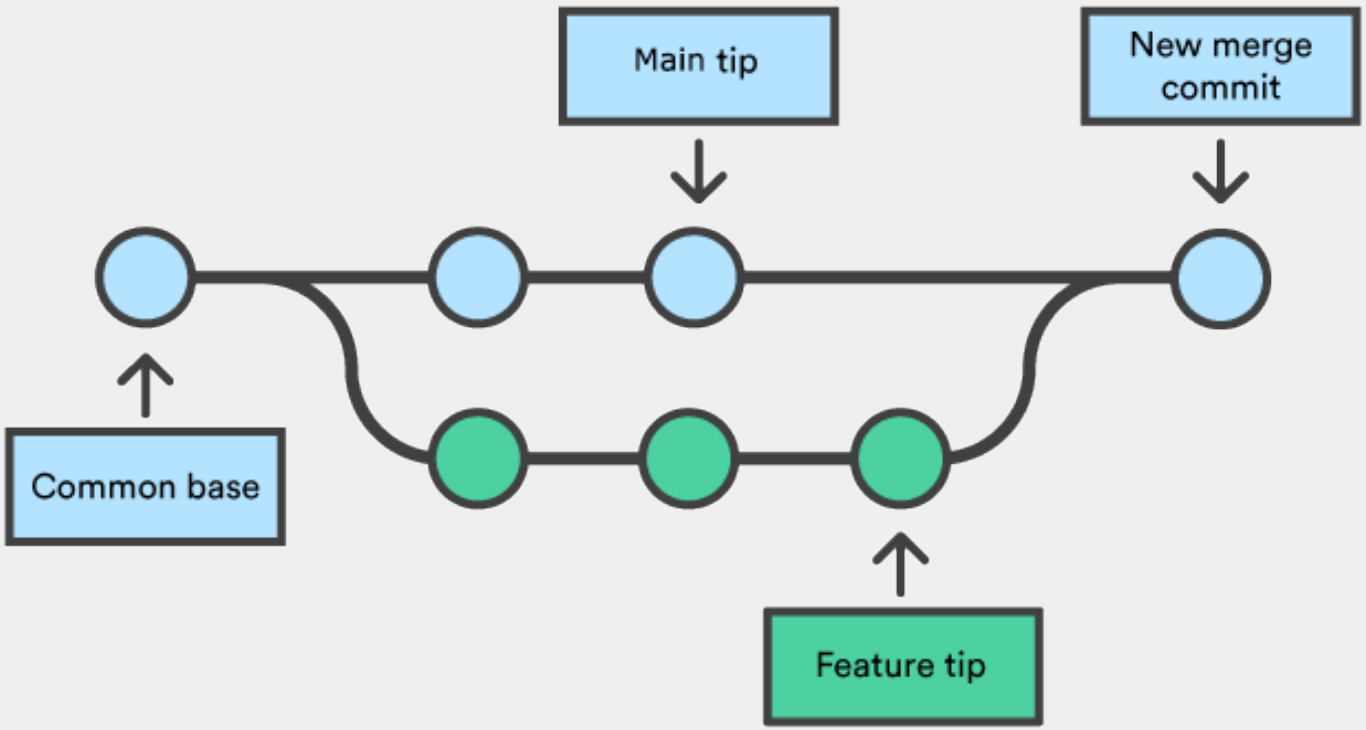
Borra la rama llamada `<test>`

Fusión/Merge

¿Qué es un Merge?



¿Qué es un Merge?



`git checkout feature`

Nos situamos en la **rama feature**

`git checkout feature`

Nos situamos en la **rama feature**

Ahora realizo mis cambios en la rama...

Flujo de trabajo

```
git checkout feature
```

Nos situamos en la **rama feature**

Ahora realizo mis cambios en la rama...

```
git add .
```

Agregamos nuestros cambios

Flujo de trabajo

```
git checkout feature
```

Nos situamos en la **rama feature**

Ahora realizo mis cambios en la rama...

```
git add .
```

Agregamos nuestros cambios

```
git commit -m "mensaje"
```

Creamos un **commit**.

Flujo de trabajo

```
git checkout feature
```

Nos situamos en la **rama feature**

Ahora realizo mis cambios en la rama...

```
git add .
```

Agregamos nuestros cambios

```
git commit -m "mensaje"
```

Creamos un **commit**.

```
git merge feature main
```

Fusionamos los cambios que hicimos en la rama **feature** a la rama principal: **main**.

Resumen

1. La fusión en git combina secuencias de cambios en un solo historial unificado

Resumen

1. La fusión en git combina secuencias de cambios en un solo historial unificado
2. Git permite fusionar las confirmaciones automáticamente salvo que haya **conflictos**

Conflictos

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollo

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**e intentan **fusionar** sus cambios.

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**e intentan **fusionar** sus cambios.
- La manera mas eficiente de **evitarlos** es trabajar en **ramas separadas**

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**e intentan **fusionar** sus cambios.
- La manera mas eficiente de **evitarlos** es trabajar en **ramas separadas**y esperar que el otro desarrollador suba sus cambios a la rama **main**

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**e intentan **fusionar** sus cambios.
- La manera mas eficiente de **evitarlos** es trabajar en **ramas separadas**y esperar que el otro desarrollador suba sus cambios a la rama **main**para **Integrarlos** en mi rama de trabajo, y subir los mios.

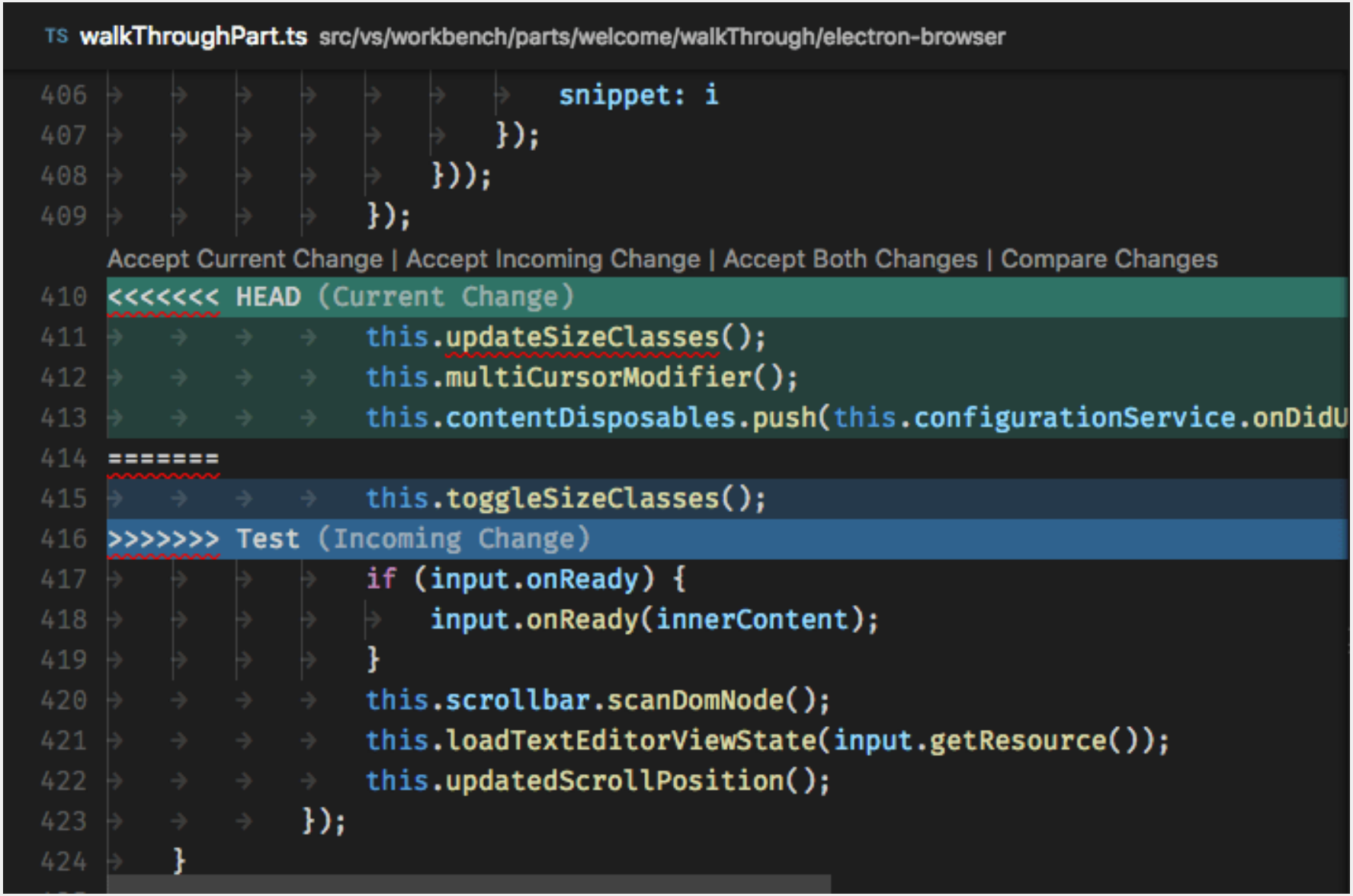
Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**e intentan **fusionar** sus cambios.
- La manera mas eficiente de **evitarlos** es trabajar en **ramas separadas**y esperar que el otro desarrollador suba sus cambios a la rama **main**para **Integrarlos** en mi rama de trabajo, y subir los mios.
- Si surgen conflictos, ¡No se preocupen!

Conflictos

- Los conflictos emergen cuando dos miembros del mismo equipo de desarrollotrabajan en el **mismo archivo**e intentan **fusionar** sus cambios.
- La manera mas eficiente de **evitarlos** es trabajar en **ramas separadas**y esperar que el otro desarrollador suba sus cambios a la rama **main**para **Integrarlos** en mi rama de trabajo, y subir los mios.
- Si surgen conflictos, ¡No se preocupen!
  Siempre y cuando trabajemos con archivos de texto.

Resolviendo un conflicto en VSCode



Etiquetas/tags

¿Qué es una etiqueta?

¿Qué es una etiqueta?

- Una referencia a un punto específico de la historia del repositorio.

¿Qué es una etiqueta?

- Una referencia a un punto específico de la historia del repositorio.
- Se utiliza para capturar un punto, y marcar una "versión".

¿Qué es una etiqueta?

- Una referencia a un punto específico de la historia del repositorio.
- Se utiliza para capturar un punto, y marcar una "versión".
- Es una rama que **no cambia**.

Comandos útiles

`git tag`

Nos muestra la lista de etiquetas del repositorio.

Comandos útiles

`git tag`

Nos muestra la lista de etiquetas del repositorio.

`git tag <nombre>`

Crea una nueva etiqueta desde **mi punto actual**, llamada `<nombre>`

Comandos útiles

`git tag`

Nos muestra la lista de etiquetas del repositorio.

`git tag <nombre>`

Crea una nueva etiqueta desde **mi punto actual**, llamada `<nombre>`

`git tag -a <nombre> -m "descripcion"`

Crea una nueva etiqueta desde **mi punto actual**, llamada `<nombre>`, incluyendo una descripcion a preferencia.

Comandos útiles

`git tag`

Nos muestra la lista de etiquetas del repositorio.

`git tag <nombre>`

Crea una nueva etiqueta desde **mi punto actual**, llamada `<nombre>`

`git tag -a <nombre> -m "descripcion"`

Crea una nueva etiqueta desde **mi punto actual**, llamada `<nombre>`, incluyendo una descripcion a preferencia.

`git checkout v1.4`

Nos situa en la **etiqueta** `"v1.4"`, tal como si fuera una **rama**.

Terminando...

Recursos Útiles

- Tutorial de Atlassian acerca de **git**:
  `https://www.atlassian.com/es/git/tutorials`
- Integración nativa en **VSCode**:
  `https://code.visualstudio.com/`
- Cliente de **GitHub**
  `https://desktop.github.com/`

¡Gracias por participar!

Universidad Tecnológica Nacional
Facultad Regional Córdoba
IEEE Student Branch

IEEE