Nombre: Luis Carlos Alegría Treviño y Alberto Alejandro Rodríguez Pérez

Matrícula: 440138274 y 870124922

Materia: Modelos y métodos para el

desarrollo de software

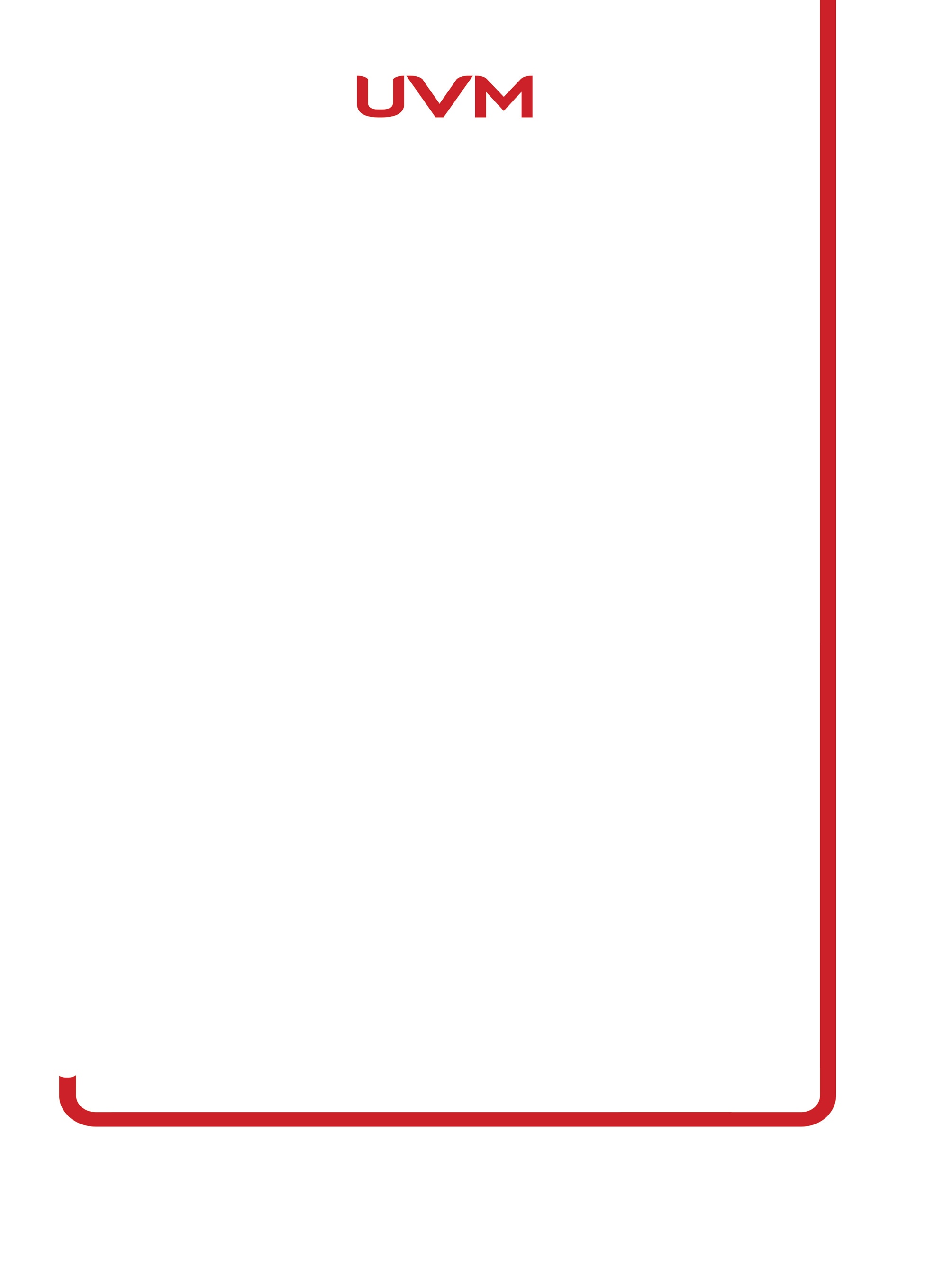
Profesor: Julio Cesar Villa López

Campus: Cumbres

Fecha de entrega: 06/06/2024

Uso y configuración de GIT.

[Subtítulo del documento]



Investiga y contesta:

1. **¿Cómo se establece el nombre de usuario en git?**

Para el nombre de usuario es muy fácil gracias el comando dentro de GitHub el cual es git config --global user.name "Tu Nombre".

1. **¿Cómo se establece el correo del usuario en git?**

Al igual que el nombre de usuario gracias al comando dentro de GitHub es muy fácil establecer el correo del usuario git config --global user.email "correo".

1. **¿Cómo se activa el coloreado de salida en git?**

El coloreado de salida dentro de GitHub la podemos configurar con un comando pequeño y esto es muy importante según la comodidad git config --global color.ui auto.

1. **¿Cómo se muestra el estado original en los conflictos de git?**

Hay dos comandos para esta pregunta ya que con git estatus se puede ver el estado de este archivo que está en conflicto y con git diff se puede observar el contenido del archivo con conflictos.

1. **¿Cómo se muestra la configuración?**

Para poder observar la lista de configuración es un simple comando git config –list.

1. **¿Cómo se crea un repositorio en GIT?**

Un repositorio se puede crear de diferentes formas ya que dentro de la aplicación de GitHub o con un comando dentro de GIT git init.

1. **¿Cómo se muestra el estado de un repositorio en git?**

Dentro de Git puede haber muchas formas de ver el estado de un repositorio, pero la más común dentro de Git es git status.

1. **¿Cómo se muestra las diferencias entre versiones en git?**

Al igual que con los conflictos se puede ver las diferencias entre versiones con el mismo comando de git diff.

1. **¿Cómo se puede volver a un directorio de trabajo o versión anterior?**

Para volver a un commit en específico se utiliza git checkout <commit-hash>

y para revertir ciertos cambios en el area de trabajo de este último commit git reset –hard.

1. **¿A qué se refiere con un pull en git?**

Un "pull" en Git es el proceso de actualizar tu repositorio local con los cambios que se han hecho en el repositorio remoto.

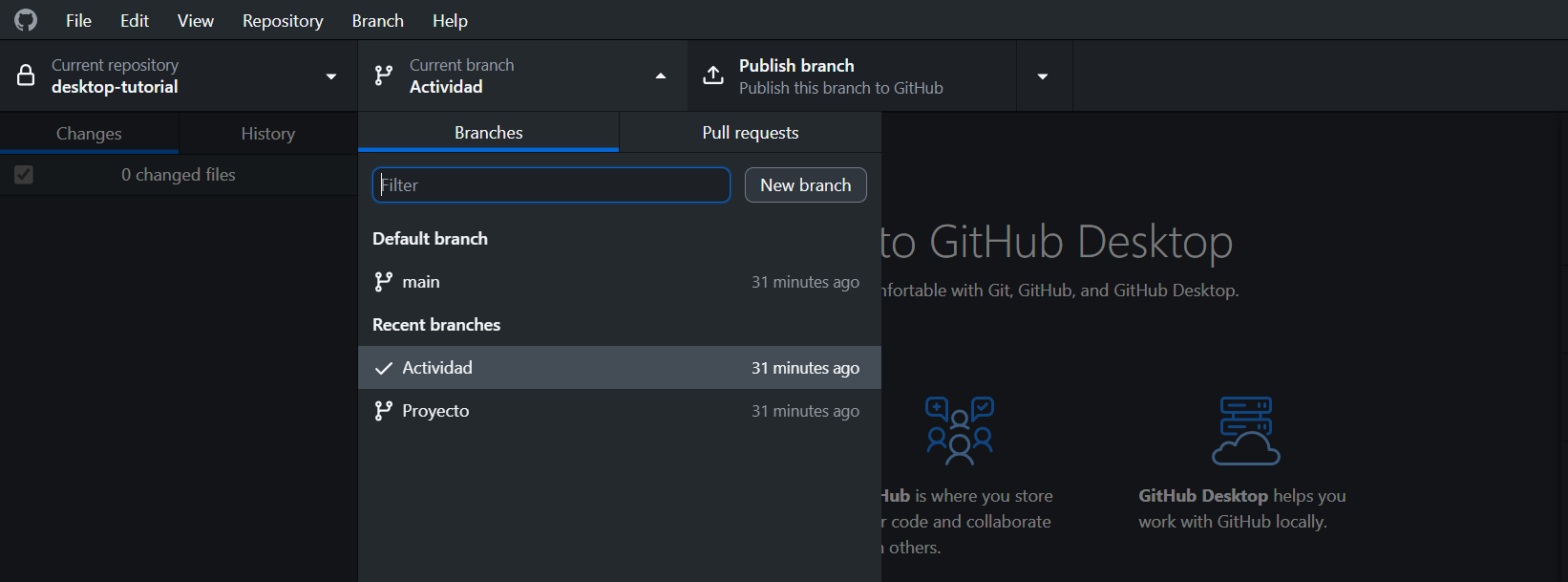
1. **¿A qué se refiere con un push en git?**

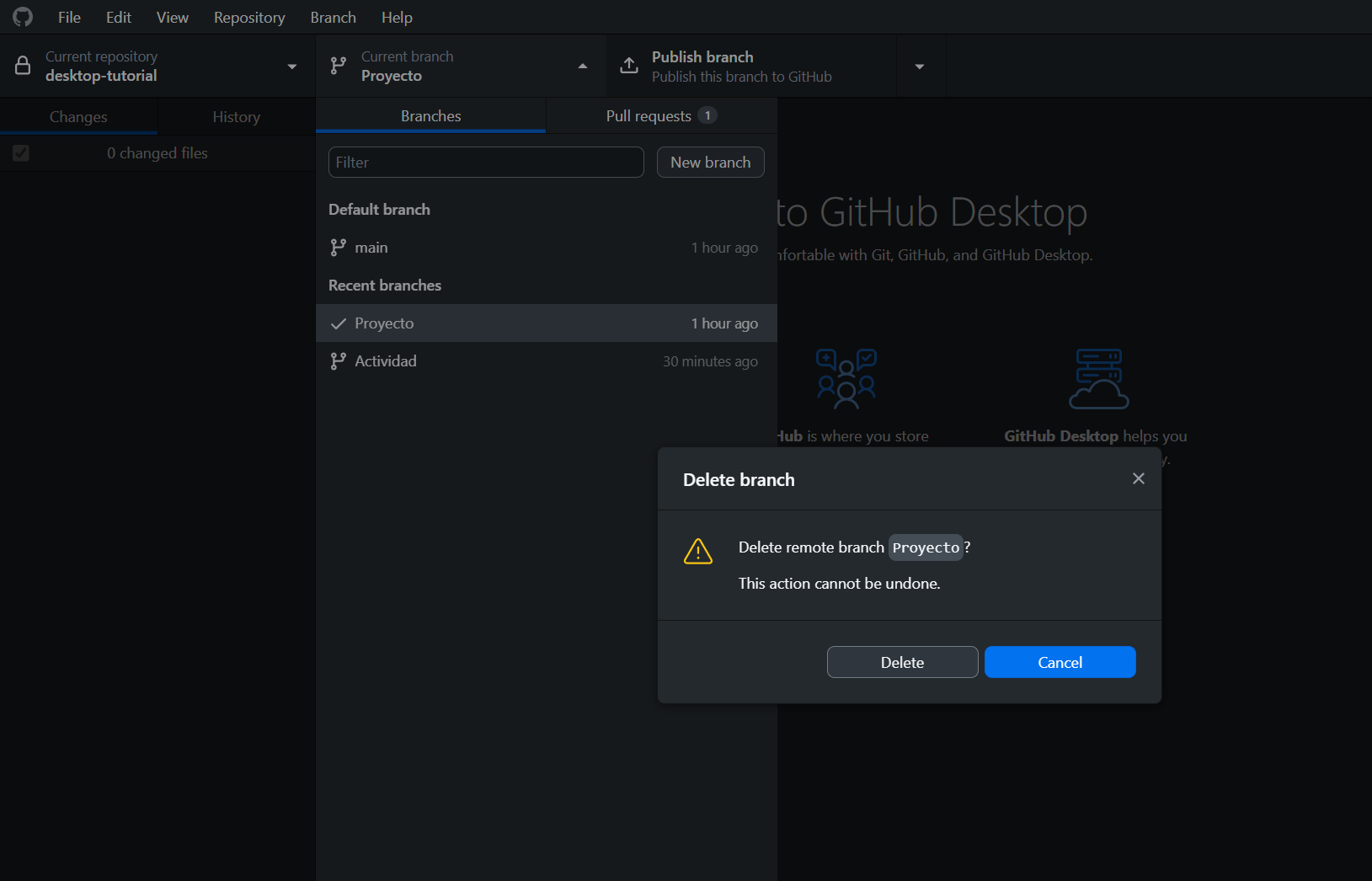
Un "push" en Git es el proceso de enviar tus cambios locales al repositorio remoto.

Practica:

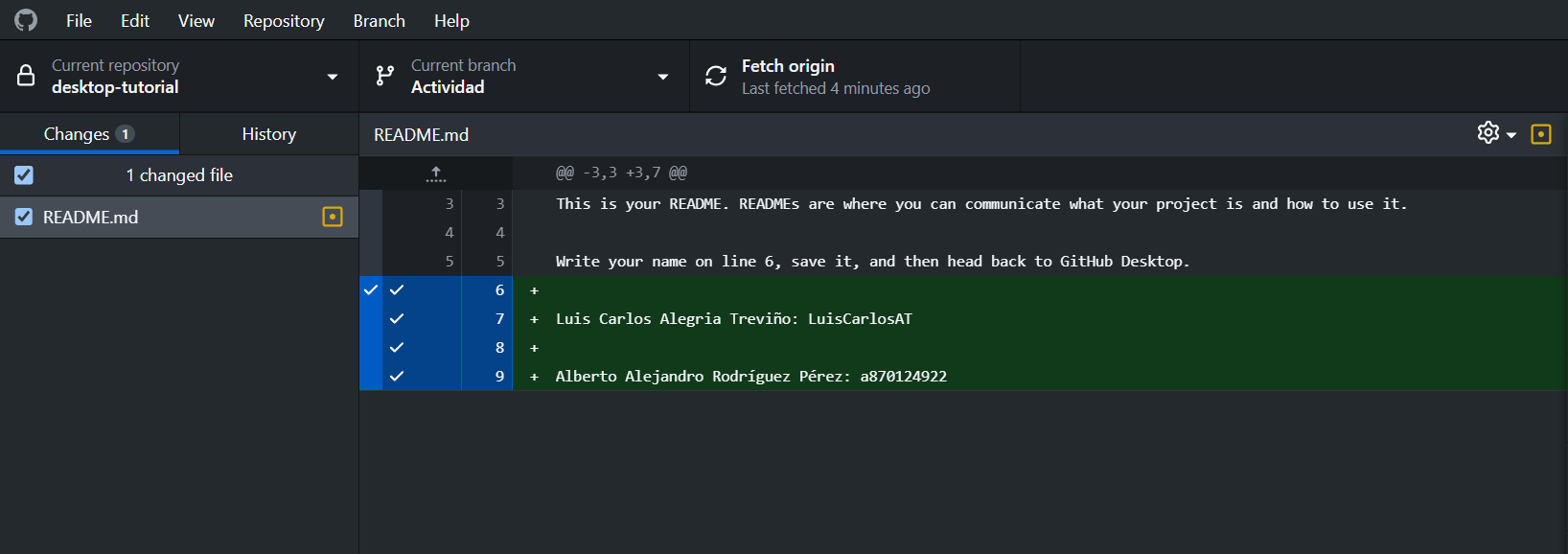
* Crea un repositorio local en Git
* Comparte ese repositorio con cada uno de tus compañeros de equipo del directorio remoto.
* Que cada compañero tenga su escritorio remoto y añada los cambios necesarios.
* Añadan los cambios al stage.
* Registren los cambios.
* Suban los cambios al remoto.
* Realiza las siguientes operaciones sobre la rama:
  + Crea su rama.
  + Ver en que rama se encuentra.
  + Cambiar de rama.
  + Ver los cambios entre ramas.
  + Fusionar ramas

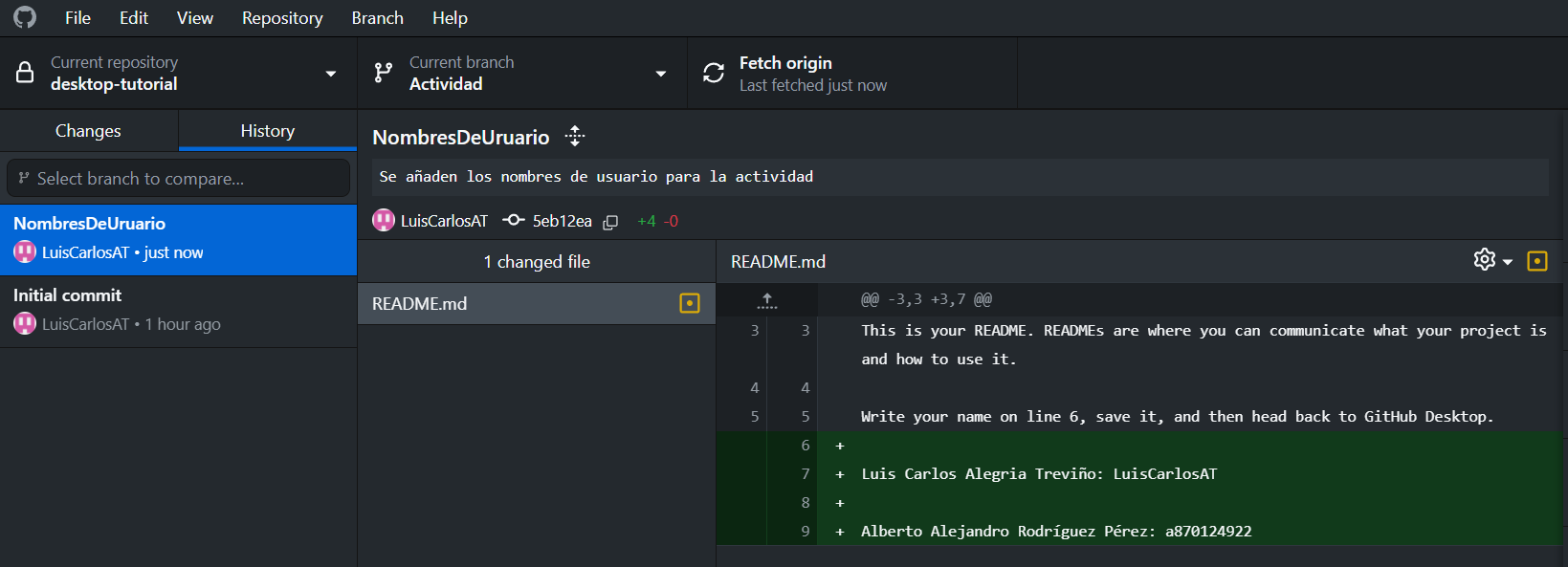
Eliminar rama.

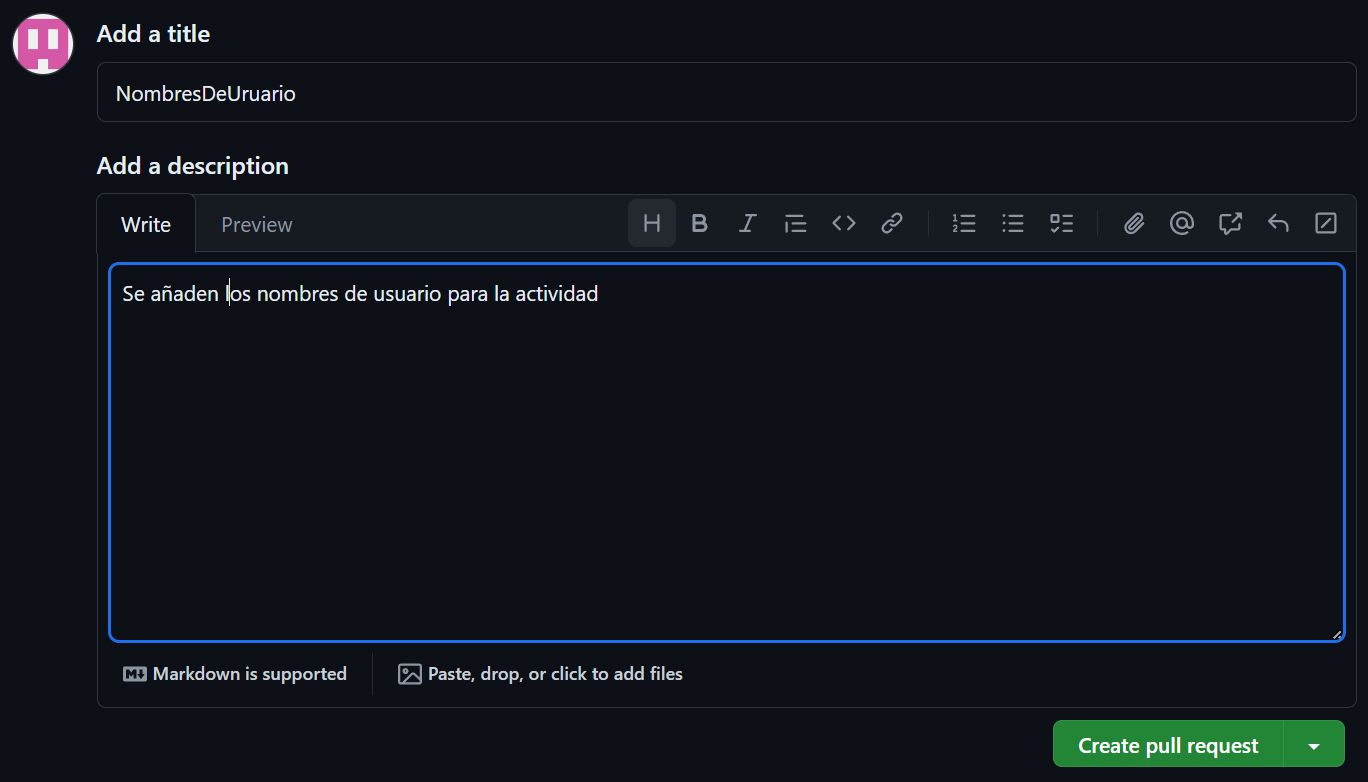




* Cada compañero deberá de mostrar su rama y su usuario además de que comparte el mismo repositorio.
* Para un pull request:
  + Crear la rama.
  + Añada su nombre de usuario en README.md en el apartado Colaboradores.
  + Sube los cambios en local.
  + Subir los cambios.
  + Ir a github y solicitar una pull request.
  + Agregar un comentario.







Código.

Ser requiere que se genere lo siguiente:

Codificación de botón de eliminar y botón limpiar.

Botón eliminar:

Debe de buscar en base al código y si no existe mencionar que así es.

Botón Limpiar:

Debe de limpiar todos los campos de texto, esto lo realiza llamando una función que se debe de generar en el código, es decir no escribimos el código en el botón si no que generamos una función y se manda llamar.

BD:

Deberás mostrar el antes y el después.

* Abre tu plantilla y genera las actividades que consideres necesarias para los pasos anteriores.
* Genera las historias de usuario.
* Pasamos a codificar.

A entregar:

* Planificación e historias de usuario en repositorio que compartirás con el docente.
* Captura de pantalla de cada uno de los pasos solicitados.
* BD
* Interfaz funcionando.
* Liga o enlace a tu repositorio.

**Historias de usuario**

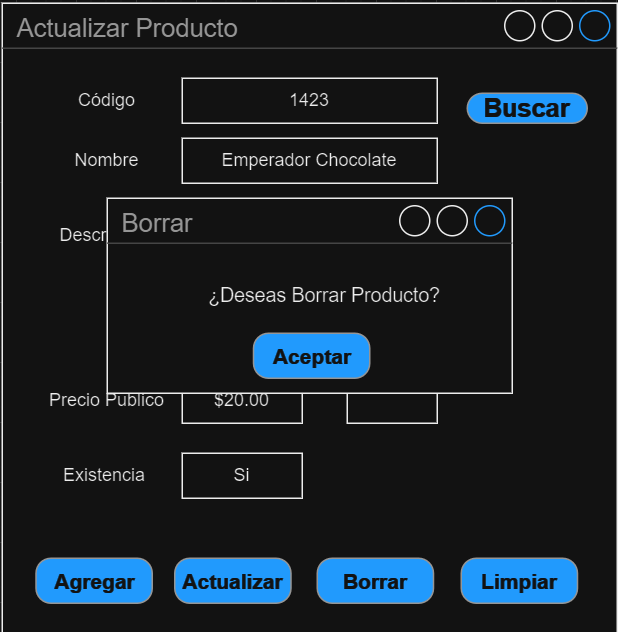
|  |  |  |
| --- | --- | --- |
| **Historia de usuario** | | |
| **Número:** 6 | **Usuario:** Desarrollador | |
| **Nombre historia:** Implementación de funcionalidad del botón “Borrar” | | |
| **Prioridad en negocio:** Alta | | **Riesgo en desarrollo:** Alta |
| **Puntos estimados:** 12 | | **Iteración asignada:** 1 |
| **Programador responsable:** Alberto Alejandro Rodríguez Pérez | | |
| **Descripción:** Implementar las sentencias en forma de código en el botón “Borrar” para que la interfaz pueda conectarse a la base “metosoftdos” y que la información de los datos pueda ser borrada de la tabla “productos”.  En caso de no existir el elemento a borrar, se debe mostrar un mensaje que lo demuestre.  Validar que los datos borrados en el software sean los mismos a los borrados en la tabla “productos”.  Validar el funcionamiento correcto de la interfaz. | | |
| **Observaciones:** La eliminación de los datos se debe dar por medio del código, es decir, el id del producto en la base de datos “metosoftdos”. | | |

|  |  |  |
| --- | --- | --- |
| **Historia de usuario** | | |
| **Número:** 7 | **Usuario:** Desarrollador | |
| **Nombre historia:** Implementación de funcionalidad del botón “Limpiar” | | |
| **Prioridad en negocio:** Alta | | **Riesgo en desarrollo:** Media |
| **Puntos estimados:** 9 | | **Iteración asignada:** 1 |
| **Programador responsable:** Luis Carlos Alegría Treviño | | |
| **Descripción:** Implementar las sentencias dentro de una función en forma de código para limpiar todos los campos de texto, el botón “Limpiar” debe llamar a dicha función, más no debe de escribirse código en el botón.  Validar que los datos a limpiar en el software sean borrados de la pantalla, más no de la base de datos.  Validar el funcionamiento correcto de la interfaz. | | |
| **Observaciones:** Al pulsar el botón “Limpiar” deben de borrarse todos los campos de texto y aparecer vacíos para poder registrar un nuevo producto. | | |

|  |  |  |
| --- | --- | --- |
| **Historia de usuario** | | |
| **Número:** 8 | **Usuario:** Desarrollador | |
| **Nombre historia:** Creación y llenado de repositorio | | |
| **Prioridad en negocio:** Media | | **Riesgo en desarrollo:** Baja |
| **Puntos estimados:** 8 | | **Iteración asignada:** 1 |
| **Programador responsable:** Alberto Alejandro Rodríguez Pérez | | |
| **Descripción:** Se debe ingresar a la plataforma GitHub en la cual se realizará el repositorio. Se debe de invitar a todos los miembros del equipo a colaborar en el repositorio. Es necesario realizar algunas operaciones en las ramas para entender el funcionamiento del sistema, y por último es necesario subir la planeación, las historias de usuario, las interfaces y demás elementos que fueron trabajados en el desarrollo de este entregable. | | |
| **Observaciones:** Entre las operaciones que se tienen que realizar con las ramas, se encuentran su creación, cambio, fusión y eliminación, además deben de agregarse comentarios en los distintos apartados del repositorio. | | |

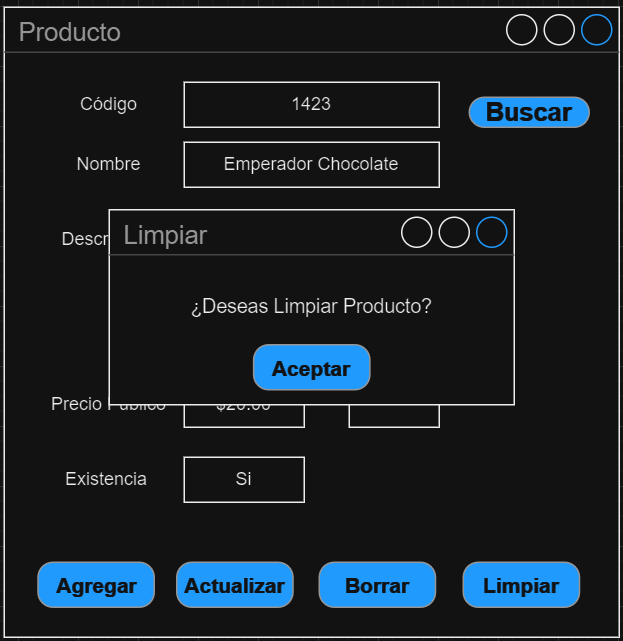
**Interfaces con explicación**





En este caso, al utilizar el botón “Borrar”, se eliminará el producto por completo, incluyendo su ID, ya que se estará eliminando completamente el producto del catálogo.





Al utilizar el botón “Limpiar”, se limpiará la pantalla por completo y se asignará automáticamente un nuevo ID para ingresar un nuevo producto.