



Desarrollo con Tecnologías Emergentes

Clase práctica

Herramientas para el trabajo en grupo:
Gestión de repositorios con GitLab

Contenido

1. Objetivo de la práctica	3
2. Instalación de git en el ordenador local	3
3. Preparacion de un repositorio compartido en gitlab.com	4
3.1 Clonar el repositorio	4
3.2 Añadir el archivo tg1.html	5
3.3 Añadir el archivo .gitlab-ci.yml	6
3.4 Subir los cambios al repositorio en gitlab.com	7
3.5 Actualizar los repositorios locales del resto del grupo	8
4. Comenzar a trabajar en un apartado del informe	10
5. Crear una rama y hacer cambios locales	11
6. Subir cambios al repositorio remoto	13
7. Solicitud de integración en rama master (merge request)	15
8. Aceptación de integración en rama master	16
9. Cerrar el issue en el tablero del proyecto	18

1. Objetivo de la práctica

Con esta práctica se utilizará las herramientas Git y GitLab para gestionar los repositorios compartidos de los trabajos en grupo.

2. Instalación de git en el ordenador local

Para poder trabajar con un repositorio compartido por un grupo en un proyecto de gitlab.com, hay que tener instalado el programa git en el ordenador local, para poder descargar y subir cambios del/al repositorio remoto mediante comando git.

Descargar e instalar desde <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalaci%C3%B3n-de-Git>, eligiendo el sistema que corresponda. Para Windows: <https://git-scm.com/download/win>

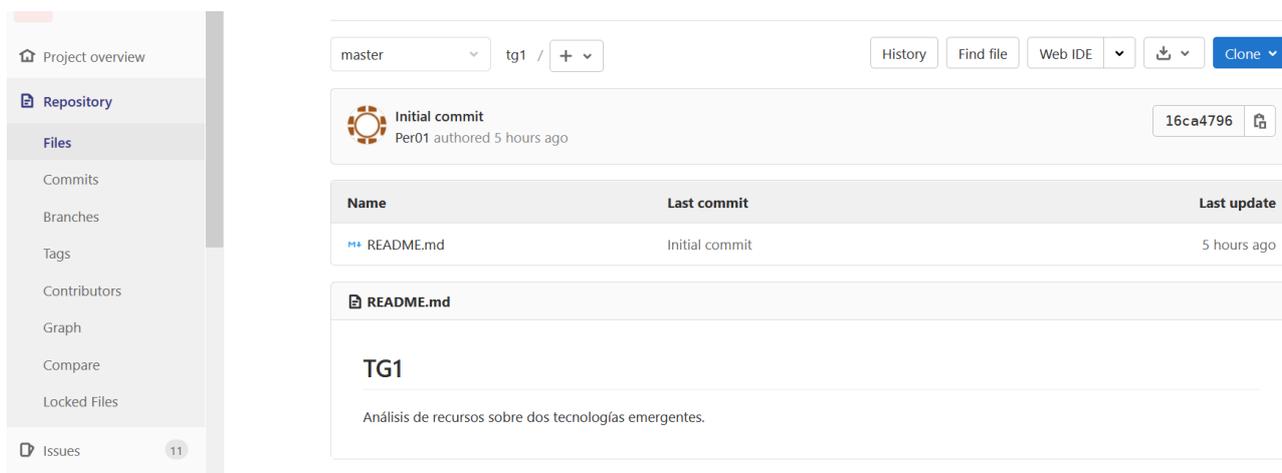
Comprobar que está instalado, desde el intérprete de comandos del ordenador con el comando:

```
D:\> git --version  
git version 2.5.2.windows.2
```

3. Preparacion de un repositorio compartido en gitlab.com

En GitLab, todos los proyectos tienen un repositorio git integrado, en el que se pueden crear carpetas y almacenar cualquier tipo de archivos.

En la práctica anterior se había creado un proyecto TG1 en gitlab.com, pero no se utilizó el repositorio del proyecto, que de momento sólo tiene el archivo README.md que se crea por defecto, como puede comprobarse desde la sección Repository del menú principal de GitLab.



The screenshot shows the GitLab interface for a repository named 'tg1'. The left sidebar contains navigation options: Project overview, Repository (selected), Files, Commits, Branches, Tags, Contributors, Graph, Compare, Locked Files, and Issues (11). The main content area shows the 'Initial commit' by 'Per01' from 5 hours ago, with commit ID '16ca4796'. Below this is a table of files:

Name	Last commit	Last update
README.md	Initial commit	5 hours ago

The README.md file content is displayed below the table:

```

TG1

Análisis de recursos sobre dos tecnologías emergentes.

```

El coordinador del grupo debe descargarse esta primera versión del repositorio a su ordenador local, añadir dos archivos nuevos que veremos a continuación y subir los cambios al repositorio en gitlab, para que todos los demás miembros puedan empezar a trabajar en sus apartados del informe que hay que realizar.

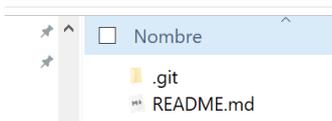
3.1 Clonar el repositorio

Para clonar el repositorio remoto en nuestro ordenador local hay que ejecutar el comando git clone con la URL del proyecto. En el siguiente comando cambiar “g20” por el grupo de cada uno.

```
D:\> git clone https://gitlab.com/dte2021g20/tg1
git version 2.5.2.windows.2
```

Al ejecutarlo se crea una carpeta de nombre tg1 en el lugar desde donde se ha ejecutado el comando, en este caso en la raíz del disco D: con una copia del repositorio remoto, que de momento sólo tiene el archivo README.md y una carpeta oculta de nombre .git, utilizada por el programa git para la información sobre el control de versiones del repositorio.

DATA (D:) > tg1

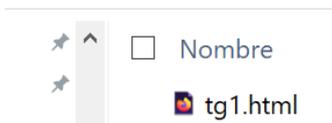


3.2 Añadir el archivo tg1.html

El objetivo del proyecto es crear un informe en formato html. Como parte del enunciado del trabajo TG1 se adjuntaba un archivo tg1.html preparado como plantilla para el informe.

Hay que crear una subcarpeta llamada “public” y copiar en ella el archivo tg1.html.

DATA (D:) > tg1 > public



La razón de crear esta subcarpeta se explica en el siguiente apartado.

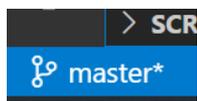
Además, el coordinador debe editar el archivo y escribir:

- en el título el número del grupo, el nombre de la categoría de las tecnologías asignadas y el nombre de las dos tecnologías
- en el índice y en los títulos de los apartados 2 y 4 los nombres de las dos tecnologías

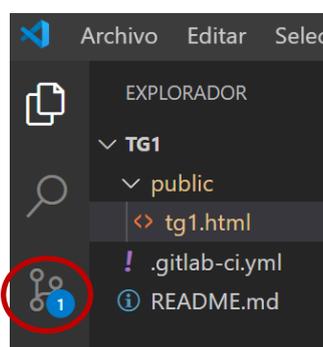
```
<body>
  <header>
    <h1>Desarrollo con Tecnologías Emergentes. Curso 2020-21<br>
    Trabajo TG1 del grupo G20<br>
    Título: Recursos sobre dos tecnologías emergentes para evaluar la
    accesibilidad de aplicaciones móviles Android<br>
    Tecnologías: Accessibility Engine (axe) y Accessibility Scanner</h1>
  </header>
  <nav>
    <ul>
      <li><a href="#1">1. Información general</a></li>
      <li><a href="#2">2. Tecnología Accessibility Engine (axe)</a></li>
      <ul>
        <li><a href="#2.1">2.1. Cursos</a></li>
        <li><a href="#2.2">2.2. Documentos</a></li>
        <li><a href="#2.3">2.3. Software</a></li>
      </ul>
      <li><a href="#3">3. Tecnología Accessibility Scanner</a></li>
      <ul>
        <li><a href="#3.1">3.1. Cursos</a></li>
        <li><a href="#3.2">3.2. Documentos</a></li>
      </ul>
    </ul>
  </nav>

```

Se recomienda usar el editor [Visual Studio Code](#), porque ofrece facilidades para su integración con git. Por ejemplo, en cada momento podemos saber en qué rama del repositorio estamos trabajando, pues lo indica en la parte inferior izquierda.



Y también nos avisa de si hay cambios pendientes de consolidar con un commit en el repositorio local.



3.3 Añadir el archivo .gitlab-ci.yml

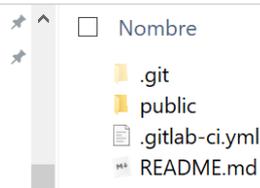
Vamos a aprovechar una utilidad que ofrece gitlab, que es la de desplegar automáticamente páginas web en un hosting público gratuito ubicado, no en el dominio gitlab.com donde está la plataforma de los proyectos, sino en otro dominio llamado gitlab.io, que es un hosting para páginas web.

Para ello es necesario crear un archivo llamado .gitlab-ci.yml en la raíz del repositorio, cuyo contenido debe ser el siguiente.

Archivo D:\tg1\gitlab-ci.yml
<pre>pages: stage: deploy script: - echo 'Despliegue' artifacts: paths: - public only: - master</pre>

De tal forma que la estructura final del repositorio local queda de la siguiente forma.

DATA (D:) > tg1



NOTA: Gracias el archivo `.gitlab-ci.yml`, cada vez que se suba un cambio al repositorio en `gitlab.com`, se activa una función conocida como de “integración y despliegue continuo” que ejecuta los trabajos definidos en este archivo. En este caso hay un solo trabajo llamado “pages” que lo que hace es desplegar al hosting `gitlab.io` el contenido de la carpeta indicada en la sección `paths`, que en este caso es “public”; es decir despliega el archivo `tg1.html` que está en esta carpeta a dicho hosting, de tal forma que quedará publicado como página web en la dirección: <https://dte2021g20.gitlab.io/tg1/tg1.html>

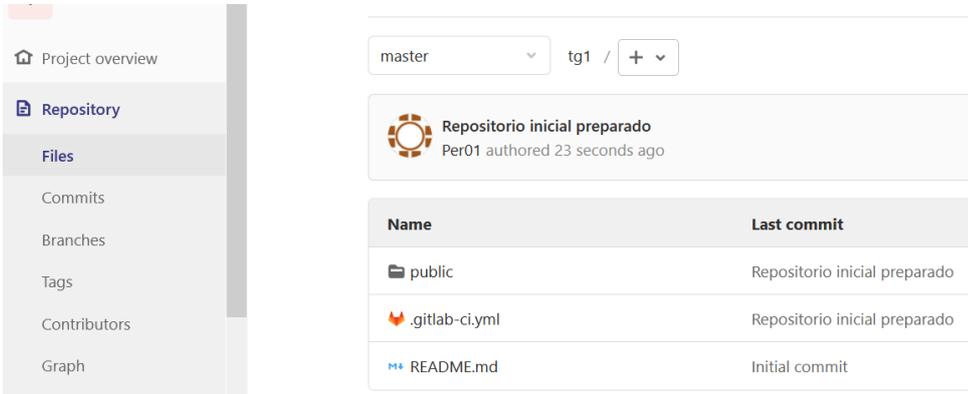
3.4 Subir los cambios al repositorio en `gitlab.com`

Ahora hay que actualizar el repositorio remoto con todos los cambios que hemos hecho, y para ello con `git` siempre se procede de la misma forma, con esta secuencia de comandos desde la carpeta del repositorio local: primero se prepara una nueva versión con `add` y `commit` y después se envía al repositorio remoto con `push`.

```
D:\tg1> git add .
D:\tg1> git status
D:\tg1> git commit -m "Repositorio inicial preparado"
D:\tg1> git push origin
Username for 'https://gitlab.com': ...
Password ...
```

NOTA: Si no queremos que `git` pida usuario y contraseña cada vez que ejecutemos `push` podemos ejecutar el comando: `git config --global credential.helper store`. Y si queremos que lo vuelva a pedir: `git config --global --unset credential.helper`.

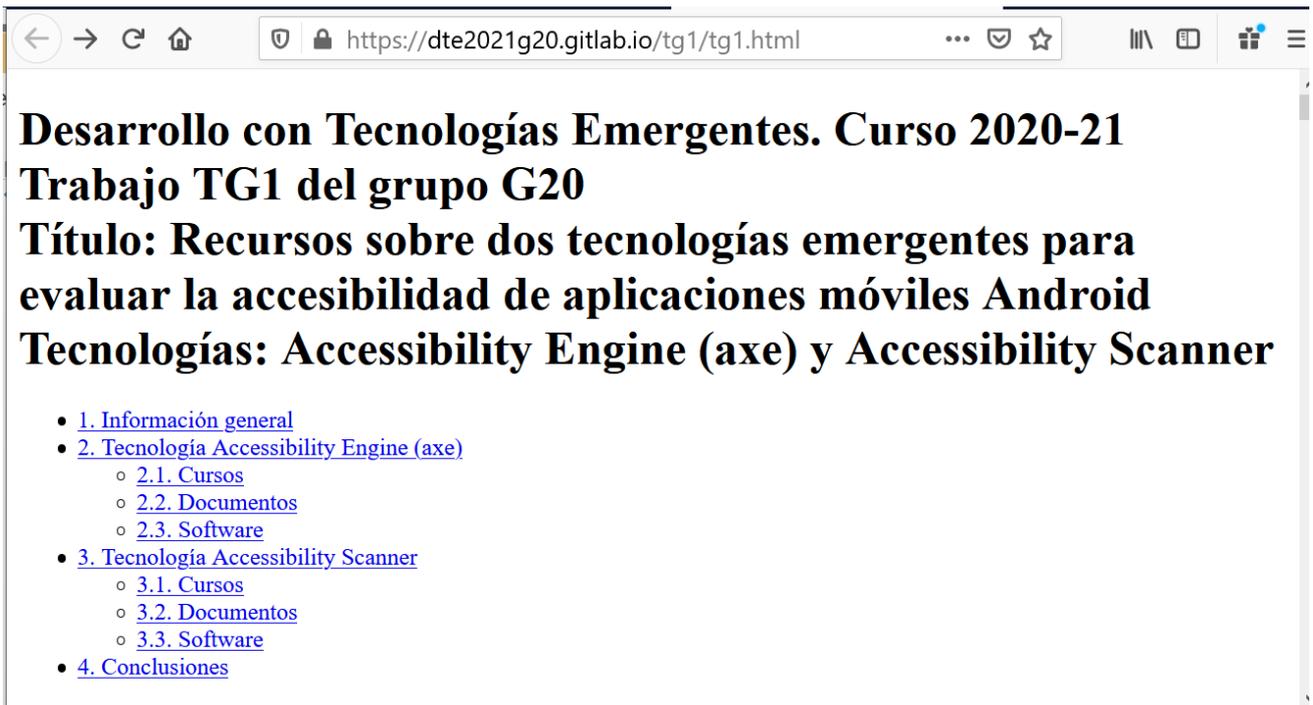
Si vamos al proyecto en `gitlab.com` podemos ver que el repositorio se ha actualizado a la nueva versión o `commit` llamado “Repositorio inicial preparado”.



The screenshot shows the GitLab repository interface. On the left is a sidebar with navigation options: Project overview, Repository, Files, Commits, Branches, Tags, Contributors, and Graph. The main content area shows the repository name 'master' and 'tg1 / +'. Below this, a message states 'Repositorio inicial preparado' by 'Per01' 23 seconds ago. A table lists the repository's files and their commit history:

Name	Last commit
public	Repositorio inicial preparado
.gitlab-ci.yml	Repositorio inicial preparado
README.md	Initial commit

Como hemos creado el archivo `.gitlab-ci.yml`, se habrá realizado automáticamente (tarda algunos minutos) el despliegue de la página web en el hosting <https://dte2021g20.gitlab.io/tg1/tg1.html>



The screenshot shows a web browser displaying the project page. The address bar shows the URL <https://dte2021g20.gitlab.io/tg1/tg1.html>. The page content is as follows:

Desarrollo con Tecnologías Emergentes. Curso 2020-21

Trabajo TG1 del grupo G20

Título: Recursos sobre dos tecnologías emergentes para evaluar la accesibilidad de aplicaciones móviles Android

Tecnologías: Accessibility Engine (axe) y Accessibility Scanner

- [1. Información general](#)
- [2. Tecnología Accessibility Engine \(axe\)](#)
 - [2.1. Cursos](#)
 - [2.2. Documentos](#)
 - [2.3. Software](#)
- [3. Tecnología Accessibility Scanner](#)
 - [3.1. Cursos](#)
 - [3.2. Documentos](#)
 - [3.3. Software](#)
- [4. Conclusiones](#)

3.5 Actualizar los repositorios locales del resto del grupo

Ahora que ya está el repositorio inicial preparado, el resto de miembros del grupo deben clonarlo en su ordenador local para empezar a trabajar en el informe.

Para ello deben ejecutar el comando `git clone` con la URL del proyecto. En el siguiente comando cambiar "g20" por el grupo de cada uno.



```
D:\> git clone https://gitlab.com/dte2021g20/tg1
```

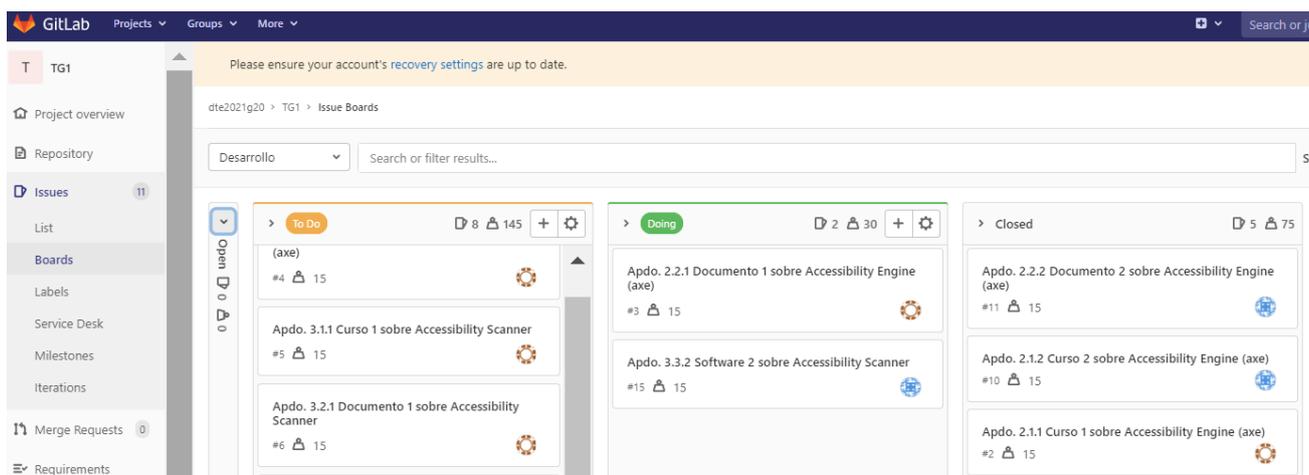
Si algún miembro del grupo ya había clonado por error el repositorio antes de que el coordinador hubiera subido los cambios, puede actualizar su repositorio local con el comando pull dentro de la carpeta del repositorio.

```
D:\tg1> git pull
```

4. Comenzar a trabajar en un apartado del informe

Cuando un miembro del grupo vaya a comenzar a escribir uno de los apartados del informe tg1.html, lo primero que debe hacer es arrastrar en el tablero el issue de ese apartado a la columna “Doing”, para que el resto del grupo sepa en qué está trabajando cada uno en cada momento.

Por ejemplo, si va a escribir el apartado 3.3.2 debería, debería quedar así.



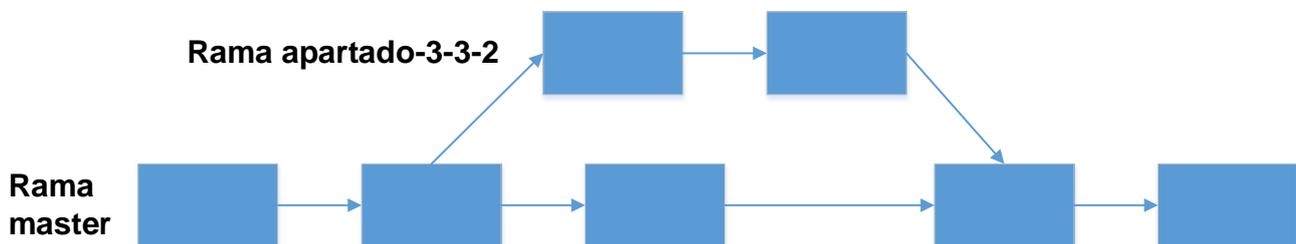
Y para estar seguros de que vamos a trabajar con la última versión del informe que haya en el repositorio compartido en gitlab, antes de empezar es conveniente ejecutar el comando pull dentro de la carpeta del repositorio.

```
D:\tg1> git pull
```

5. Crear una rama y hacer cambios locales

Para que el coordinador del grupo pueda después combinar los cambios que realice cada miembro del grupo en el informe tg1.html, cada miembro debe crear una rama en su repositorio local antes de editar el archivo.

El sistema de ramas de git permite manejar varias versiones en paralelo de uno o varios archivos. En la siguiente figura la rama larga sería la rama máster, la que maneja el coordinador del grupo. Y la rama que surge en un punto en paralelo podría ser por ejemplo la rama para modificar el apartado 3.3.2 del documento del trabajo tg1.html. Cada nodo del diagrama representaría una versión modificada del archivo tg1.html.



Se recomienda poner como nombre de una rama el del apartado del trabajo que se va a modificar.

Por ejemplo, si se va a modificar el apartado 3.3.2 se le podría dar el nombre “apartado-3-3-2”.

Una rama se crea utilizando el comando git branch, estando dentro de la carpeta del repositorio (tg1).

```
D:\tg1> git branch apartado-3-3-2
```

Ahora tenemos dos ramas. Con el comando git branch se muestran las ramas del repositorio, señalada con * la rama en la que nos encontramos trabajando:

```
D:\tg1> git branch
apartado-3-3-2
* master
```

NOTA: Si usamos Visual Studio Code como editor, podemos instalar la extensión Git Graph para ver las ramas: <https://marketplace.visualstudio.com/items?itemName=mhutchie.git-graph>

Debemos cambiarnos a la rama apartado-3-3-2 antes de hacer cambios en ese apartado del informe, para ello utilizamos el comando checkout.

```
D:\tg1> git checkout apartado-3-3-2
Switched to branch 'apartado-3-3-2'
```

Estamos en la rama apartado-3-3-2, ahora todo lo que hagamos en la carpeta tg1 o en cualquier subcarpeta, forma parte de esa nueva rama. Podemos trabajar desde Windows, al estar instalado Git en el ordenador, entiende que estamos trabajando en una versión diferente a la de máster.

Por ejemplo, editamos el archivo tg1.html e introducimos algo nuevo en el apartado 3.3.2.

Si utilizamos Visual Studio Code, éste está integrado con Git y nos avisa de la rama en la que estamos, en la parte inferior de la pantalla:



Suponemos que el miembro del grupo llamado Per02 es quien escribe ese apartado:

```
<h4>3.3.2 Software 2</h4>
<table>
  <tr>
    <th>ANALISTA</th>
    <td>Per02</td>
  </tr>
  <tr>
    <th>NOMBRE</th>
    <td>Test de Accesibilidad</td>
  </tr>
  <tr>
    <th>PARA QUÉ SIRVE</th>
    <td>Es una aplicación móvil que se instala en un dispositivo móvil
      Android, y aparece en la sección de servicios de accesibilidad
      del móvil, y si se activa, se muestra un icono siempre en
      pantalla que, si se selecciona, se realiza automáticamente una
      revisión de la accesibilidad de lo que se está mostrando en
      pantalla y genera un informe de errores de bajo contraste,
      tamaño reducido de botones, imágenes sin descripción, etc.
    </td>
  </tr>
  <tr>
    <th>PRECIO</th>
```

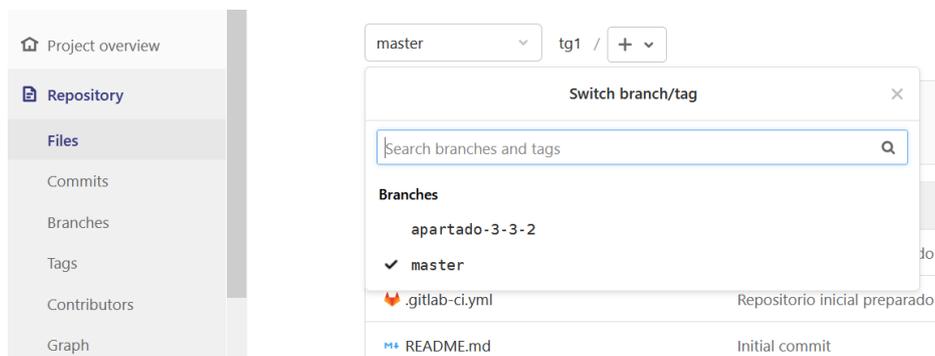
6. Subir cambios al repositorio remoto

Una vez terminado el apartado 3.3.2, el autor del mismo debe guardar los cambios en su ordenador local y enviarlos a la rama apartado-3-3-2 del repositorio remoto con los siguientes comandos:

```
D:\tg1> git add .  
D:\tg1> git status  
D:\tg1> git commit -m "Apartado 3.3.2 terminado"  
D:\tg1> git push origin apartado-3-3-2
```

Como no existía la rama en gitlab, se crea automáticamente, con el archivo tg1.html, que está en la carpeta public, modificado.

Para ver la rama apartado-3-3-2 en gitlab hay que seleccionarla de la lista de ramas:



Al seleccionarla, podemos ver que ha sido el usuario Per02 quien ha hecho un cambio en la carpeta public, que es donde está el archivo tg1.html que había modificado en su repositorio local.



The screenshot shows the GitLab interface for a repository. On the left is a sidebar with navigation options: Project overview, Repository, Files, Commits, Branches, Tags, Contributors, and Graph. The main content area shows the repository path 'apartado-3-3-2' and 'tg1'. A commit titled 'Apartado 3.3.2 terminado' by 'Per02' is highlighted. Below this is a table of commit history.

Name	Last commit
public	Apartado 3.3.2 terminado
.gitlab-ci.yml	Repositorio inicial preparado
README.md	Initial commit

Este cambio todavía no ha sido aceptado por el coordinador del grupo, y hasta que sea aceptado no se integrará en la rama master, que es donde va quedando la versión definitiva del informe tg1.html. Tampoco aparece publicado en el hosting en gitlab.io, porque en el archivo .gitlab-ci.yml se indicaba “only: master”, que significa que el despliegue al hosting sólo debe hacerse cuando haya cambios en la rama máster.

7. Solicitud de integración en rama master (merge request)

Para que el coordinador del grupo pueda aceptar los cambios realizados por otro miembro e integrarlos en la rama master del proyecto, el autor de los cambios debe solicitarlo, mediante lo que se denomina un merge request.

En el caso de los cambios en el apartado 3.3.2 del informe, su autor Per02 ya los ha subido al repositorio remoto. Una vez hecho esto, debe acceder a la sección Merge Request del menú principal de gitlab y pulsar el botón “New merge request”.

En la página que aparece debe elegir su rama como “Source branch” y la rama master como “Target branch”.

New Merge Request

Source branch	Target branch
<input type="text" value="te2021g20/tg1"/> <input type="text" value="apartado-3-3-2"/>	<input type="text" value="te2021g20/tg1"/> <input type="text" value="master"/>
 Apartado 3.3.2 terminado Per02 authored 16 minutes ago <input type="text" value="28e79e76"/> 	 Repositorio inicial preparado Per01 authored 1 hour ago <input checked="" type="checkbox"/> <input type="text" value="f04f3c0c"/> 
<input type="button" value="Compare branches and continue"/>	

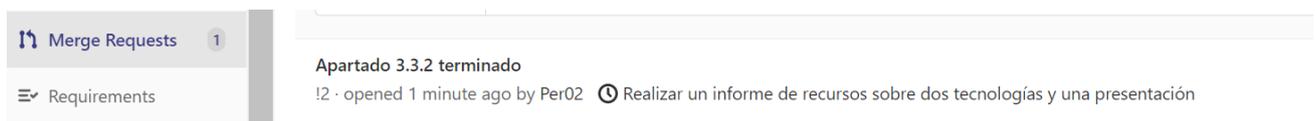
Después hay que seleccionar el botón “Compare branches and continue”.

En la página que se abre, hay que introducir estos datos:

- Assignees: Seleccionar al coordinador del grupo, para que se encargue de aceptar la integración.
- Milestones: Elegir el sprint que se creó al inicio del trabajo.
- Merge options: Dejar marcada la primera opción, para que se borre la rama cuando se haya aceptado la integración.

Y hay que pulsar “Submit merge request”.

En la sección Merge Request del menú principal se avisa de que hay una petición pendiente abierta por el miembro del grupo Per02.

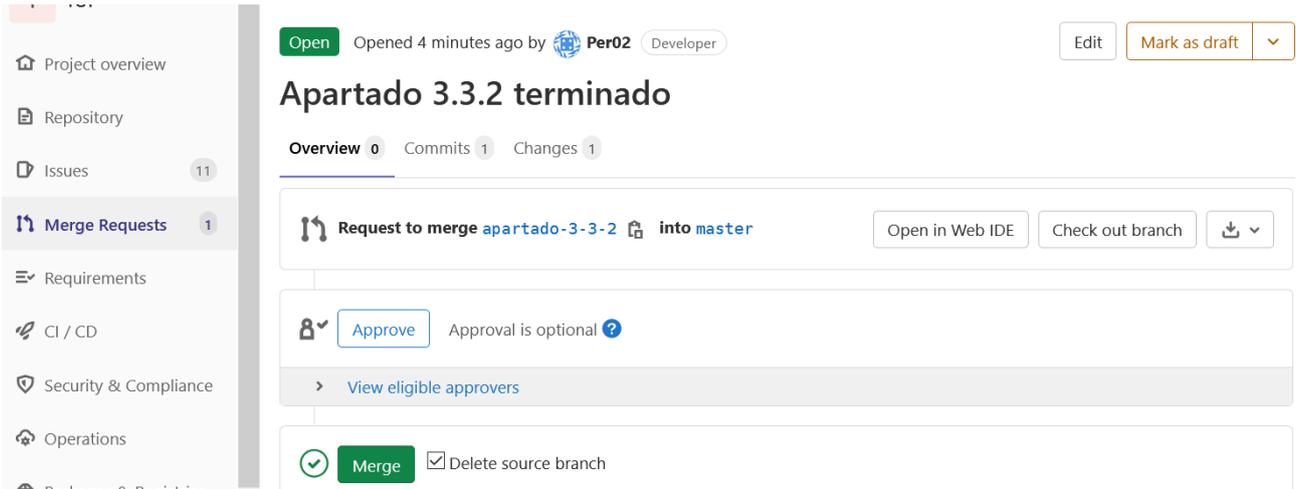


The screenshot shows the GitLab interface. On the left, there is a sidebar with a menu containing 'Merge Requests' (with a notification badge '1') and 'Requirements'. The main content area displays a notification for 'Apartado 3.3.2 terminado' by Per02, opened 1 minute ago. The notification text includes 'Realizar un informe de recursos sobre dos tecnologías y una presentación'.

8. Aceptación de integración en rama master

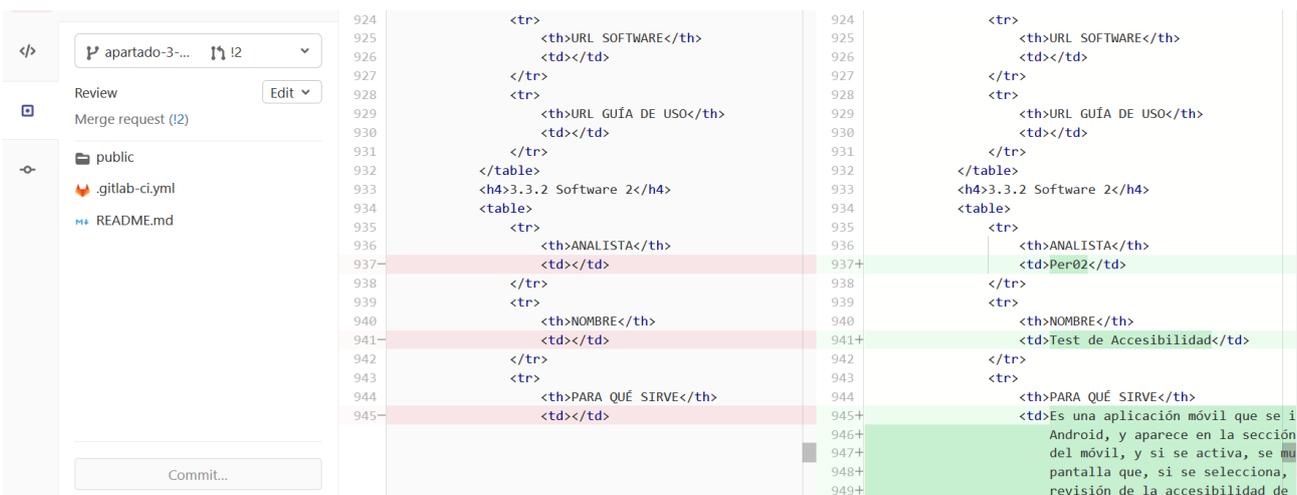
Cuando el coordinador del grupo vea que hay algún merge request abierto, debe estudiarlo y si decide que es correcto, aceptarlo para que se integren los cambios en la rama master del proyecto.

Para ello, debe abrirlo desde la sección Merge Requests del menú principal.

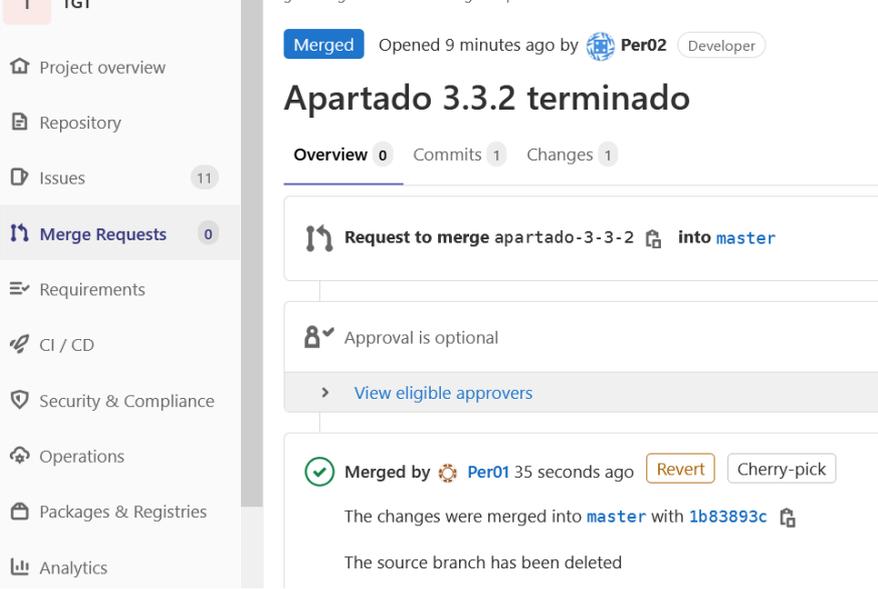


Antes de decidir si aceptarlo pulsando el botón “Merge” debe comprobar primero los cambios que ha hecho su compañero, para verificar que sólo ha tocado el apartado asignado y nada más del informe en tg1.html.

Esto puede hacerlo seleccionando el botón “Open in Web IDE”. Y se le muestran los cambios.



Si está de acuerdo, pulsaría el botón “Merge” y se integran los cambios en la rama master.



The screenshot shows a GitLab merge request interface. On the left is a sidebar with navigation options: Project overview, Repository, Issues (11), Merge Requests (0), Requirements, CI / CD, Security & Compliance, Operations, Packages & Registries, and Analytics. The main content area shows a merge request titled "Apartado 3.3.2 terminado" which has been merged. It was opened 9 minutes ago by user Per02 (Developer). The merge request includes a "Request to merge" section, an approval status of "Approval is optional" with a link to "View eligible approvers", and a "Merged by" section showing it was merged by Per01 35 seconds ago. Below this, it states "The changes were merged into master with 1b83893c" and "The source branch has been deleted".

Además, al cabo de unos minutos puede comprobarse que se ha hecho el despliegue automático de la nueva versión del archivo tg1.html al hosting gitlab.io, navegando a la URL <https://dte2021g20.gitlab.io/tg1/tg1.html>

3.3.2 Software 2

ANALISTA	Per02
NOMBRE	Test de Accesibilidad
PARA QUÉ SIRVE	Es una aplicación móvil que se instala en un dispositivo móvil Android, y aparece en la sección de servicios de accesibilidad del móvil, y si se activa, se muestra un icono siempre en pantalla que, si se selecciona, se realiza automáticamente una revisión de la accesibilidad de lo que se está mostrando en pantalla y genera un informe de errores de bajo contraste, tamaño reducido de botones, imágenes sin descripción, etc.
PRECIO	Gratis
LICENCIA DE USO	Desde la página de la app en Google Play se accede a una página general de Google en la que se indica "No podrás copiar, modificar, distribuir, vender ni alquilar ninguna parte de nuestros servicios o software. Tampoco podrás aplicar técnicas de ingeniería inversa ni intentar extraer el código fuente de ninguno de nuestros productos sin nuestro consentimiento por escrito o salvo si la legislación aplicable te lo permite."
CÓMO SE HA ENCONTRADO	Buscando en google con los términos: Accessibility Scanner
URL SOFTWARE	https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor&utm_source=www.apk4fun.com
URL GUÍA DE USO	https://support.google.com/accessibility/android/faq/6376582

3.3.3 Software 3

9. Cerrar el issue en el tablero del proyecto

Una vez que el coordinador del grupo ha integrado el apartado en el informe tg1.html de la rama máster y ya está publicado en el hosting, debe mover la ficha del issue de ese apartado a la columna "Closed" del tablero del proyecto.

> Closed		6	90
Apdo. 2.2.2 Documento 2 sobre Accessibility Engine (axe)	#11	15	
Apdo. 2.1.2 Curso 2 sobre Accessibility Engine (axe)	#10	15	
Apdo. 2.1.1 Curso 1 sobre Accessibility Engine (axe)	#2	15	
Apdo. 2.3.2 Software 2 sobre Accessibility Engine (axe)	#12	15	
Apdo. 3.1.2 Curso 2 sobre Accessibility Scanner	#13	15	
Apdo. 3.3.2 Software 2 sobre Accessibility Scanner	#15	15	