

Práctica 1b (segunda parte):

Uso de GitHub/remoto

1. Objetivos

- Aprender a usar **git** de manera individual y de forma remota
- Conocer Team Explorer de Visual Studio
- Familiarizarse con github y crear nuestro primer repositorio en github

2. Requisitos técnicos

Sigue los pasos indicados, **respetar el uso de mayúsculas y minúsculas** así como el **nombre de las carpetas, archivos, clases y métodos** que se te indique que has de crear. Requisitos que tiene que cumplir este trabajo práctico para ser evaluado (si no se cumple alguno de los requisitos la calificación será **cero**) son:

- La solución entregada no contiene archivos compilados, por lo que, **antes de comprimir, debes limpiar la solución (Compilar > Limpiar solución)**.
- El archivo entregado se llama `hada-plb.zip` (**todo en minúsculas**).
- Al descomprimir el archivo `hada-plb.zip` se crea un directorio de nombre `hada-pl` (**todo en minúsculas**), el cual contiene a su vez otro directorio de nombre `hada-pl` (**todo en minúsculas**).
- El directorio de la solución `hada-pl` contiene la copia de trabajo y el directorio `.git`.
- Los archivos que hay dentro del directorio del proyecto `hada-pl` se llaman como se indica en el enunciado (**respetando** el uso de mayúsculas y minúsculas).
- Las etiquetas, nombres de ramas, clases C# y métodos implementados se llaman como se indica en el enunciado (**respetando** el uso de mayúsculas y minúsculas).

3. Guía de evaluación

Esta práctica contará un **1.25% de la nota final**.

La práctica se puntuará de acuerdo a los siguientes criterios:

- Los distintos *commits* pedidos a lo largo del enunciado supondrán hasta el 25% de la nota.
- La creación correcta de las etiquetas pedidas supondrá hasta el 10% de la nota.
- La creación y trabajo correctos con las ramas pedidas supondrá hasta el 30% de la nota.
- La correcta creación del repositorio remoto supondrá hasta el 10% de la nota.
- La correcta contestación a las preguntas marcadas en negrita (**P1-P9**) en un nuevo

fichero **README.md** a crear en la carpeta de la solución, supone un 25% de la nota.

4. Entrega

Se entregará el directorio de la solución `hada-pl1`, junto con todo su contenido, comprimido en un fichero llamado `hada-plb.zip`.

La entrega se realizará en <http://pracdlsi.dlsi.ua.es>, no se admite ningún otro método.

Fecha límite: 23/02/2020

5. Descripción

En la segunda práctica sobre git, vamos a hacer uso de **GitHub**, un servicio de repositorio remoto para git.

Parte 0: primeros pasos con github

Github es una plataforma de desarrollo colaborativo para alojar proyectos software haciendo uso del sistema de control de versiones Git.

En ella podemos crear repositorios remotos, que contendrán nuestros proyectos. Estos pueden ser “públicos” (todo el mundo puede clonarlos y su almacenamiento es gratuito) o “privados” (haciendo uso de una cuenta de pago).

Desde hace un tiempo Github permite hacer uso gratuito de ciertos recursos que normalmente forman parte de la cuenta de pago, como poder crear proyectos “privados”. Sólo necesitas una cuenta de email institucional de la UA (@alu.ua.es) asociada a tu cuenta de Github y darte de alta en Github-Education, siguiendo estos pasos:

1. Si no te has dado aún de alta en github, entra en <https://github.com/join>
2. Ahora entra en Github-Education (<https://education.github.com/pack>) y selecciona “Get your pack”
3. Revisa que cumples los requisitos y selecciona “Yes, I'm a student”
4. Escribe tu nombre
5. Verifica tu situación académica: usa el menú desplegable para seleccionar tu email institucional o pulsa el botón “add an email address” para añadirlo, si no fue el que utilizaste para registrarte.

Figura 1. Formulario para verificar situación académica

Verify academic status

Select your school-issued email address

☐ [username@gmail.com](#)

☐ [username@github.com](#)

or [upload proof of your current school affiliation](#)

6. Indica que perteneces a la Universidad de Alicante y el año en el que te graduarás.
7. Explica cómo utilizarás GitHub y pulsa el botón "Submit Request"
8. Espera a que se confirme por email que formas parte de GitHub Education

Parte 1: crear nuestro primer repositorio remoto

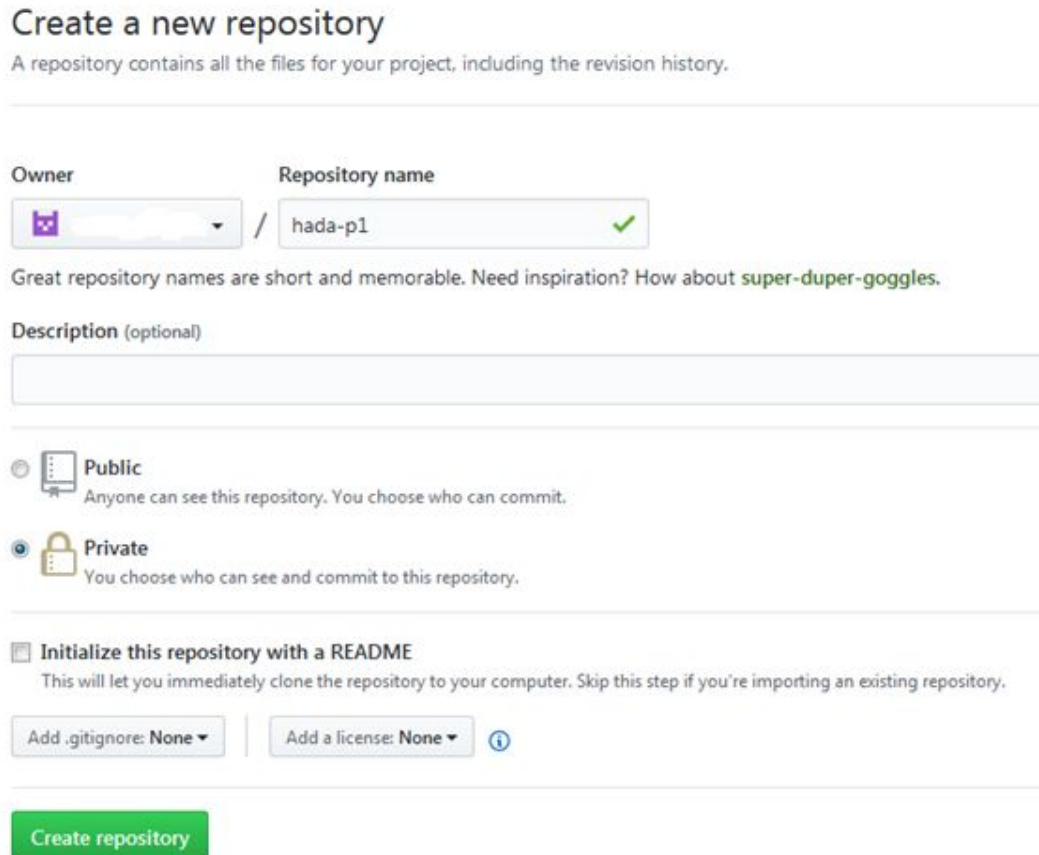
Crea un nuevo repositorio remoto en Github desde la dirección <https://github.com/new>

Esto nos permite acceder a un formulario como el de la Figura 2 e indicar: el nombre del repositorio (repository name), quién es el dueño (owner), una breve descripción y el tipo de repositorio (público o privado).




Una decisión importante es si inicializar el repositorio o no. Si el repositorio ya existe en nuestro ordenador, nos interesa NO inicializarlo. Por el contrario, si no hemos comenzado nuestra implementación, será más conveniente crear el repositorio con un readme para poder clonarlo en nuestro ordenador.

En esta práctica, vamos a elegir la primera opción, pues nuestro objetivo es subir nuestra práctica 1a a github. Para ello, creamos un repositorio siguiendo la configuración de la Figura 2.

Figura 2. Formulario para crear un nuevo repositorio





Create a new repository
A repository contains all the files for your project, including the revision history.

Owner:   / Repository name: 


Great repository names are short and memorable. Need inspiration? How about **super-duper-goggles**.

Description (optional):

☐  **Public**
Anyone can see this repository. You choose who can commit.

☒  **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None** 

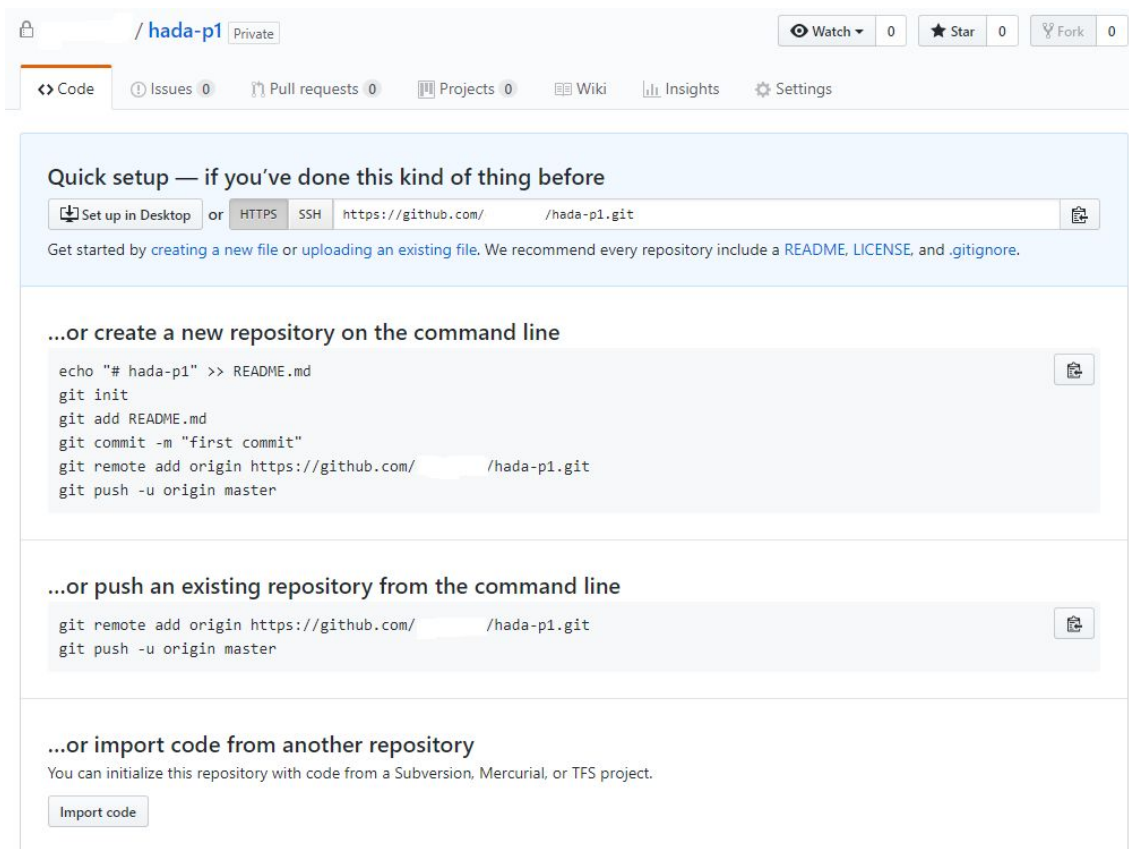
Create repository

Parte 2: añadir un repositorio remoto

Una vez creado el repositorio remoto, tal como muestra la Figura 3, github nos indica cómo podemos crear un nuevo repositorio o cómo subir un repositorio existente, en ambos casos en local desde línea de comandos. En ambos casos, para añadir un repositorio remoto asociado a nuestro proyecto, desde la carpeta del proyecto haríamos:

```
$git remote add origin ${url al repositorio remoto}
```

Figura 3. Formulario para configurar el repositorio creado



The screenshot shows the GitHub repository configuration page for a repository named 'hada-p1'. The page is titled 'Quick setup — if you've done this kind of thing before'. It provides three main options for setting up the repository:

- Set up in Desktop**: A button to download the GitHub Desktop application.
- HTTPS**: A button to use HTTPS for the remote repository.
- SSH**: A button to use SSH for the remote repository.

The repository URL is shown as `https://github.com/ /hada-p1.git`. Below the setup options, there are three sections for creating or pushing a repository from the command line:

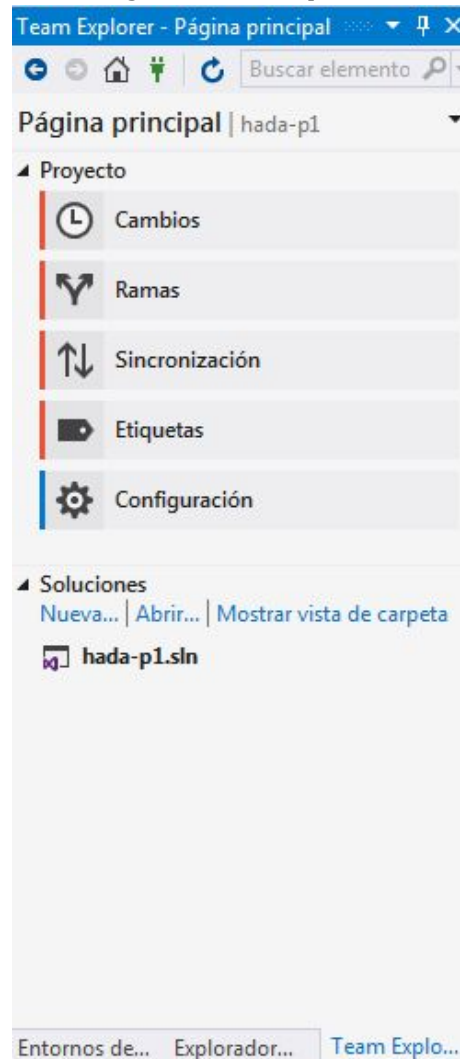
- ...or create a new repository on the command line**: A section with a code block containing the following commands:

```
echo "# hada-p1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/ /hada-p1.git
git push -u origin master
```
- ...or push an existing repository from the command line**: A section with a code block containing the following commands:

```
git remote add origin https://github.com/ /hada-p1.git
git push -u origin master
```
- ...or import code from another repository**: A section with a button labeled 'Import code' and a note: 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.'

Sin embargo, en esta práctica vamos a realizar esa misma operación desde Visual Studio. En la Figura 4 puedes ver Team Explorer, un complemento que se instala con Visual Studio y sirve para conectar los proyectos con un sistema de control de versiones. Lo encontrarás junto al Explorador de soluciones o en Ver > Team Explorer.

Figura 4. Team Explorer



Para añadir un repositorio remoto, desde Team Explorer debes seleccionar: Configuración
> Configuración de repositorios > Remotos > Agregar

Figura 5. Configuración de Team Explorer

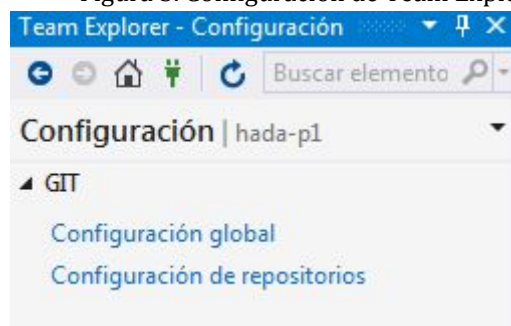
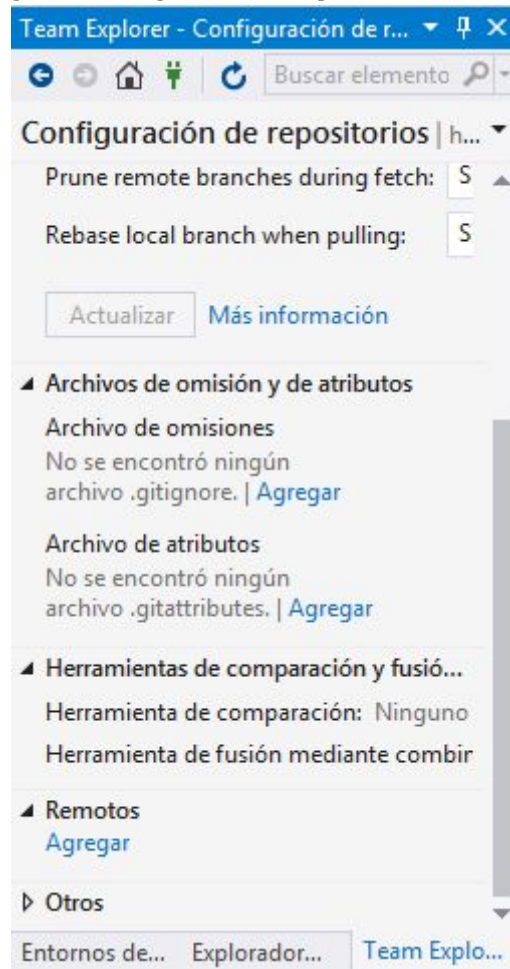
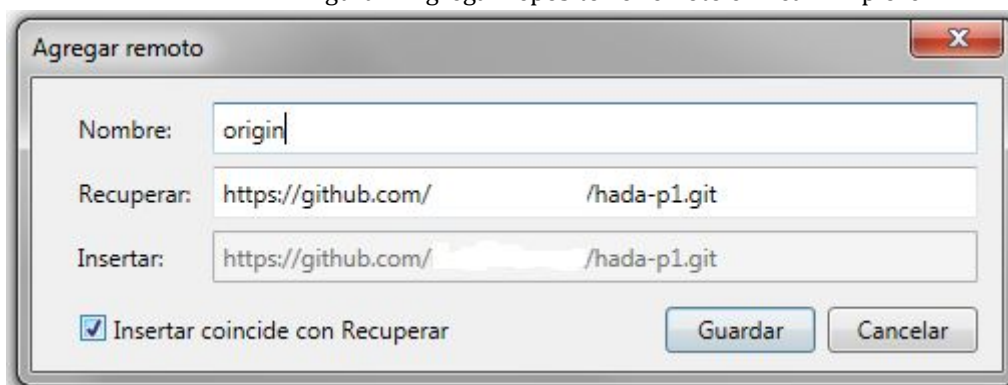


Figura 6. Configuración de repositorios en Team Explorer



En el cuadro de diálogo que se abre (Figura 7), añade el repositorio remoto cuyo nombre es *origin* y cuya url para recuperar e insertar datos sea la que te indica github al crear el repositorio.

Figura 7. Agregar repositorio remoto en Team Explorer




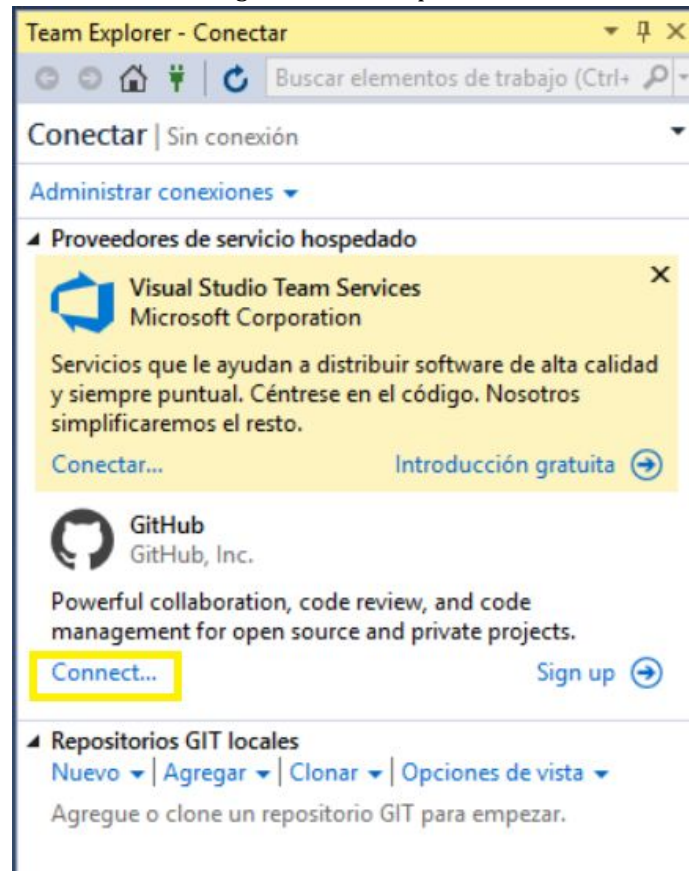
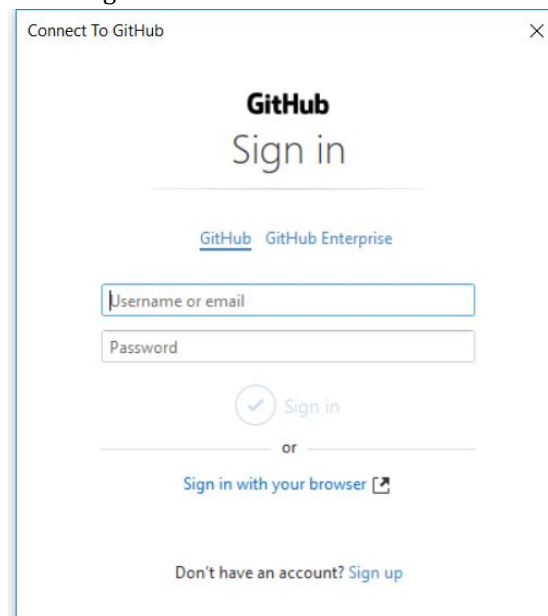
Por último, es importante que configures tus credenciales de acceso a github. Para ello, desde Team Explorer, pulsa en Administrar conexiones (el botón ) > Proveedores de Servicio hospedado > Github > Connect, tal como verás en la Figura 8.

Figura 8. Team Explorer Conectar



Esto abrirá un cuadro de diálogo como el de la Figura 9, donde debes indicar el correo con el que te has registrado en GitHub y la contraseña (Parte 0).

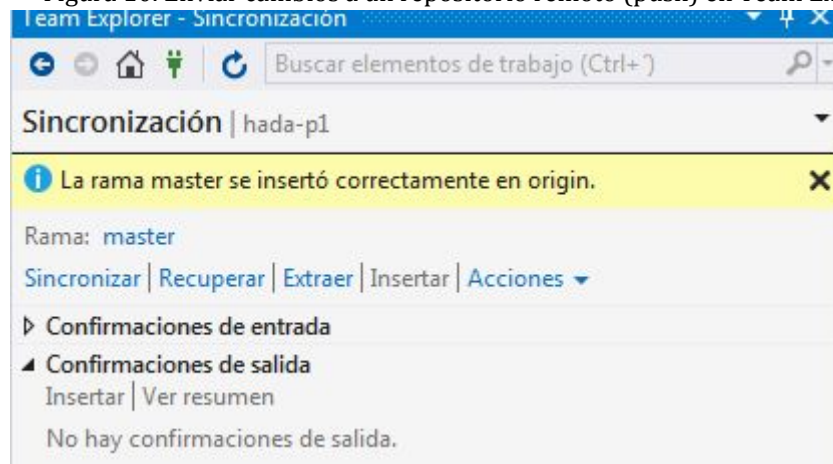
Figura 9. Introducir credenciales de GitHub



Parte 3: subir nuestros cambios al repositorio remoto

Una vez que hemos añadido el repositorio remoto, nuestras modificaciones locales se deben enviar al *origin* mediante un `push`. Por el contrario, si deseamos utilizar Team Explorer, desde nuestro proyecto local `hada-p1` nos vamos a: Proyecto > Sincronización > Confirmaciones de salida > Insertar. Haz el push mediante Team Explorer de la rama `master`, para que el resultado sea el mensaje de la Figura 10:

Figura 10. Enviar cambios a un repositorio remoto (push) en Team Explorer



Si ahora accedes en github a tu repositorio, podrás comprobar que tu práctica se ha subido a github. **P1 ¿Qué rama se ha subido exactamente?**

Importante: Ahora haz que toda la historia de tu repositorio local, es decir, todas las ramas y etiquetas, estén accesibles también en github.

Fíjate que aquí hemos podido hacer `push` directamente porque partimos de un proyecto en el que todos los archivos están preparados para ser subidos al repositorio remoto (hemos ejecutado los `git add` + `git commit` correspondientes). En la parte 5 de la práctica verás cómo realizar esas operaciones desde Team Explorer.

Si quisiéramos utilizar la línea de comandos, ejecutaríamos:

```
$git push -u origin master
```

donde `-u origin` representa el nombre que le hemos dado al repositorio remoto y `master` es el nombre de la rama que queremos subir.

Parte 4: modificar en remoto

El proyecto que hemos subido a github, tiene esta estructura de directorios:

```
hada-p1
├── hada-p1.sin
├── hada-p1
└── readme.md
```


[. .]

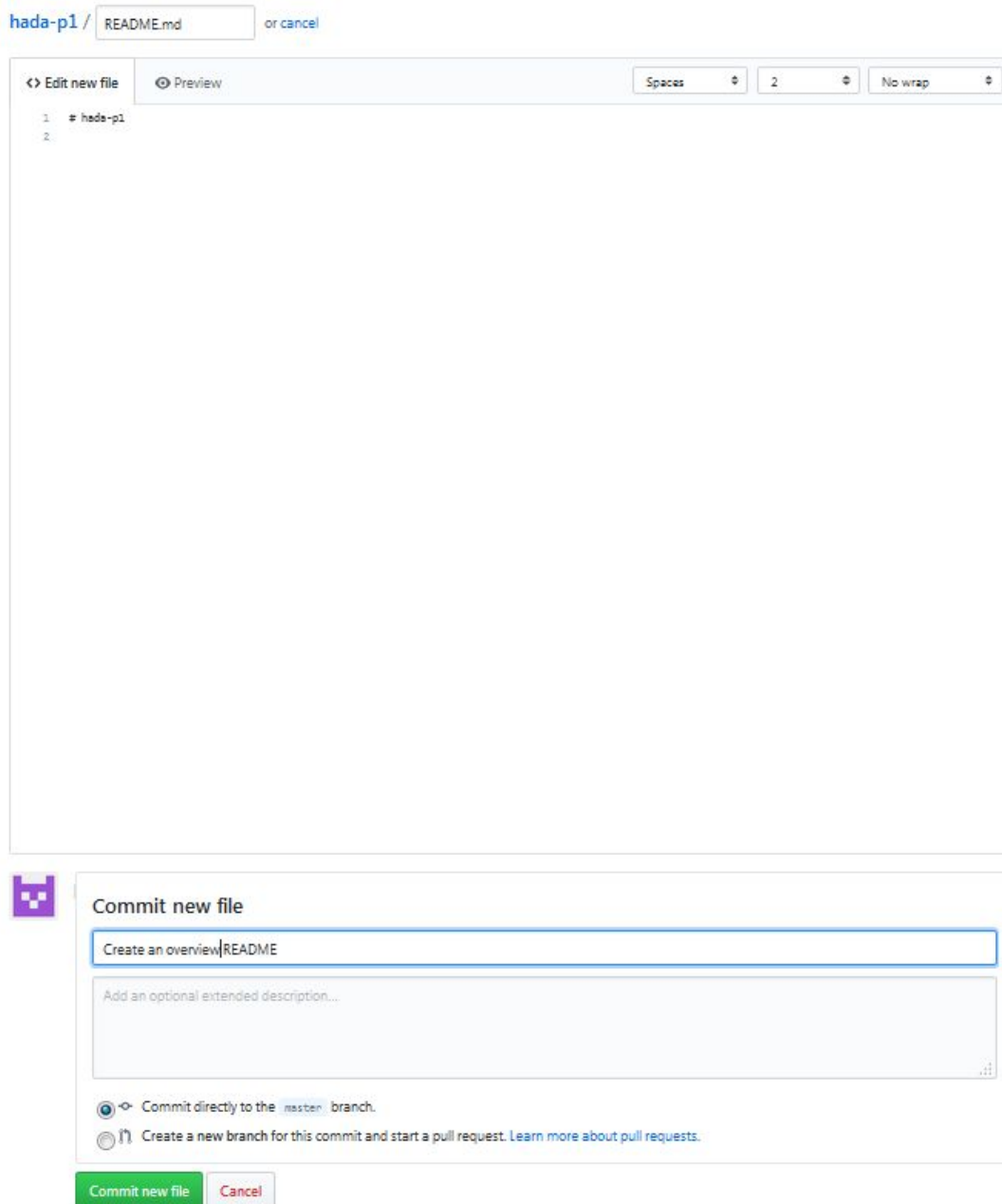
Lo recomendable es tener un readme en el raíz (en la carpeta del proyecto solución), por eso la web de github te sugiere añadir uno al acceder a tu repositorio. Vamos a hacerle caso y añadir un readme pulsando el botón “Add a README” en la web de github.

Figura 11. Github sugiere añadir un readme al proyecto



Esto te permitirá editar el archivo README directamente desde la web. En este caso, github ejecuta por tí el `git add README.md` correspondiente. Solo debes indicarle un mensaje para el `git commit`.

Figura 12. Commit desde Github



En este caso, puedes elegir entre dos opciones: trabajar directamente sobre la rama master o crear una nueva rama. Por ahora, trabaja con master.

Este cambio provocará un conflicto entre el repositorio remoto y el local, ya que en el remoto acabas de añadir un archivo que la copia local no tiene. Se explicará con más detalle cómo solucionar este conflicto en la parte 6.


Parte 5: modificar código en local y subir esos cambios al repositorio remoto

Comentar el código es una buena práctica de programación que te permitirá, por ejemplo,

recordar qué hace una función que implementaste hace tiempo o entender un código desarrollado por otra persona. En C# podemos documentar clases y funciones escribiendo en la línea anterior `///` y pulsando la tecla intro.

Por ejemplo:

```
///  
class HadaP1  
{  
...  
}
```

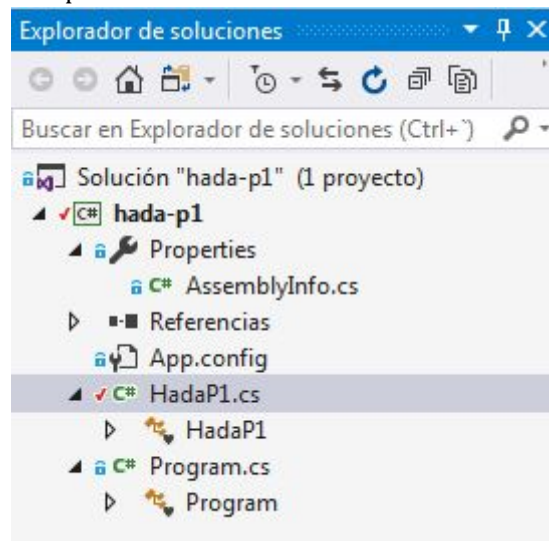


```
/// <summary>  
/// Esta clase contiene métodos para convertir de  
/// segundos a minutos y viceversa  
/// </summary>  
class HadaP1  
{  
...  
}
```

Documenta la clase HadaP1 y todos sus métodos. Consulta [1] para más información sobre los comentarios de código.

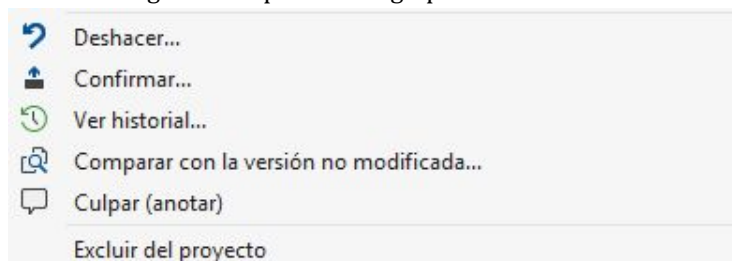
Al acabar fíjate en el Explorador de soluciones (Figura 13). Verás que el proyecto hada-p1 y la clase HadaP1.cs tienen una marca roja. Esto te indica que ese archivo tiene cambios sin incluir en el repositorio.

Figura 13. Explorador de solución con cambios no añadidos al repositorio



Si haces clic con el botón derecho sobre un archivo con esa marca, tienes varias opciones disponibles como ves en la Figura 14:

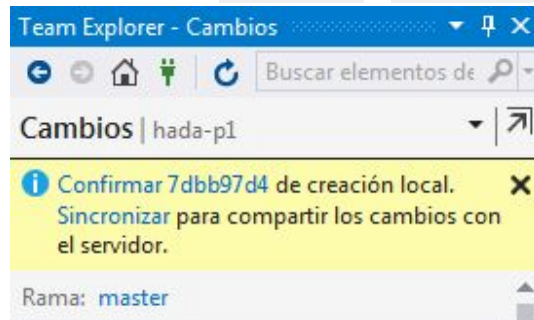
Figura 14. Opciones de git para archivos modificados



P2 ¿Qué pasa al utilizar la opción Comparar con la versión no modificada?

Como recordarás, para guardar los cambios en git del archivo modificado en el repositorio local de git, debíamos ejecutar los `git add` + `git commit` correspondientes desde la consola. Para hacer esta operación mediante Team Explorer, en la sección Cambios, busca el archivo HadaP1.cs y haz clic con el botón derecho >Agregar al stage. Añade un mensaje al commit y pulsa el botón “Hacer commit de cambios “staged””. Verás un mensaje como en la Figura 15.

Figura 15. Mensaje al realizar un `git add` + `git commit` satisfactoriamente



Ve a la web de github y observa si hay algún cambio en el repositorio remoto. Pulsa sincronizar. Ahora desde Sincronización (Figura 16), en la sección Confirmaciones de salida, pulsa Insertar.

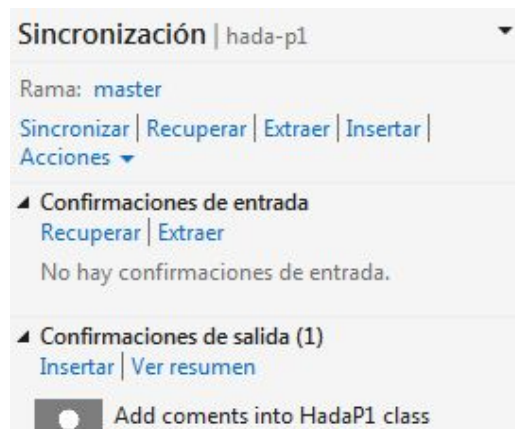
P3 ¿Y ahora ves algún cambio en el repositorio remoto? ¿Por qué?

P4 ¿Qué significa Confirmar > “Hacer commit de cambios staged” y cuál sería el comando git equivalente?

P5 ¿Qué significa Confirmar > “Hacer commit de cambios staged e insertar” y cuál sería el comando git equivalente?

P6 ¿Qué significa Confirmar > “Hacer commit de cambios staged y sincronizar” cuál sería el comando git equivalente?

Figura 16. Sección Sincronización



Este cambio provocará un conflicto entre el repositorio remoto y el local, ya que en el local acabas de modificar un archivo y habrá ciertas líneas de código que la copia remota no tiene todavía. Se explicará con más detalle cómo solucionar este conflicto en la parte 6.

Parte 6: conflictos

Los conflictos ocurren cuando trabajamos en paralelo una o varias personas.

Ahora que trabajas solo, los conflictos pueden aparecer al trabajar en varios ordenadores, por ejemplo cuando trabajas con el ordenador del laboratorio de prácticas y cuando trabajas con el de tu casa.

Cuando trabajas con varias personas, los conflictos pueden surgir al editar el mismo archivo.

Los sistemas de control de versiones nos ofrecen herramientas para resolver esos conflictos.

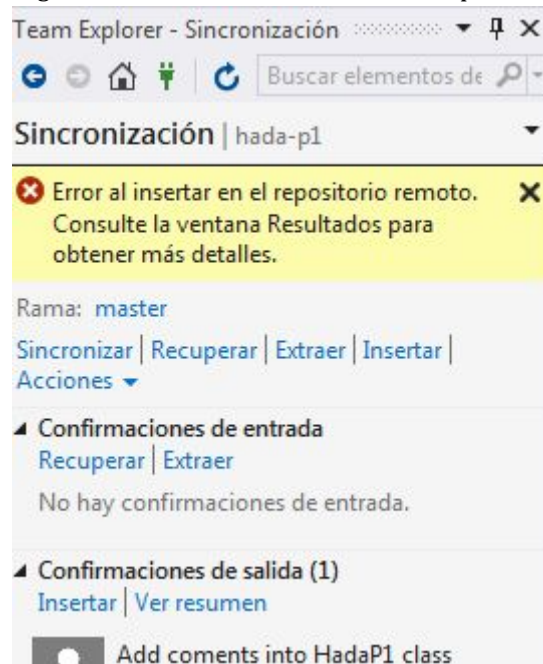
El resultado de la tercera parte de la práctica es un conflicto entre nuestra copia de trabajo local y la copia en remoto (*origin*) disponible en github. La razón es que la copia remota tiene un archivo que la copia local no tiene.

El resultado de la cuarta parte de la práctica es un conflicto entre nuestra copia de trabajo local y la copia en remoto (*origin*) disponible en github. La razón es que la copia local de las clases c# (*cs) tiene una versión diferente a la copia remota.

Ambos han evitado que tus cambios no se envíen a github. Visual Studio avisa con un error que te aparece tanto en la ventana Salida como en Team Explorer.

```
Enviando cambios de master
Error: failed to push some refs to 'https://github.com/usuario/hada-p1.git'
Error: hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
Se produjo un error al insertar en el repositorio remoto: rejected Updates were
rejected because the remote contains work that you do not have locally. This is
usually caused by another repository pushing to the same ref. You may want to first
integrate the remote changes before pushing again.
```

Figura 17. Error al enviar cambios al repositorio remoto

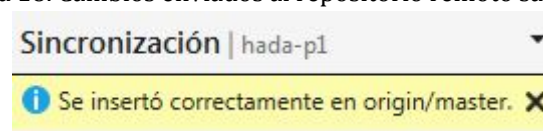


Para solucionarlo queremos conseguir los commits del repositorio remoto que no tenemos en local, es decir, hacer `fetch`. En Team Explorer, en la sección Confirmaciones de entrada, pulsarás Recuperar. Entonces verás los commits entrantes que no tenemos incluidos. Para incluirlos hacemos Sincronizar o Confirmaciones de entrada > Extraer.

P7 ¿Cuál es la diferencia entre pulsar Sincronizar o Extraer? ¿Cuáles serían los comandos git equivalentes?

Ahora tendrás 2 confirmaciones de salida que puedes insertar (push). Si los conflictos fueran más complicados y git no supiera resolverlo directamente te aparecerían otras opciones. Como no es el caso, podemos insertar los cambios en el repositorio remoto.

Figura 18. Cambios enviados al repositorio remoto satisfactoriamente



Comprueba si ves algún cambio en el repositorio remoto.

Parte 7: añadir un nuevo conversor

Ahora ya eres todo un experto en git y sabes trabajar en remoto y en local. Ahora debes crear un nuevo método que convertirá horas a minutos (en el archivo **HadaP1.cs**) y ofrece al usuario esta nueva opción de conversión (en el archivo **Program.cs**). Para ello debes:

1. crear una nueva rama partiendo de master llamada `devel2`
2. implementar el código correspondiente



3. documentar el código desarrollado
4. guardar los cambios en el repositorio local
5. guardar los cambios en el repositorio remoto
6. mezclar o fusionar la rama devel2 con la master, creando una etiqueta v0.4
7. repetir paso 5

Todo lo anterior debe quedar disponible en el repositorio remoto.

P8 ¿Qué significa Ramas > “Fusionar mediante combinación” y cuál sería el comando git equivalente?

P9 ¿Qué significa Confirmar > “Fusionar mediante cambio de base” cuál sería el comando git equivalente?

Referencias:

- [1] <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/xml/doc/recommended-tags-for-documentation-comments>
- [2] <https://docs.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-vs-2017?view=vsts>
- [3] <https://www.codemag.com/article/1411061/The-Simplest-Thing-Possible-Git-and-Visual-Studio>
- [4] <https://protecno.io/posts/como-usar-git-en-visual-studio/>
- [5] <https://github.com/github/VisualStudio/blob/master/docs/using/index.md>