IES BARAJAS

DESARROLLO WEB EN ENTORNO CLIENTE

Autor:
Juan José García Lazo

1

Programación orientada a Objetos

Constructor

Método (función) especial que introduce datos en las distintas características de la clase

Clase (Abstracto, "sin vida")

Estructura (plantilla, molde) que define un objeto. Es un conjunto de atributos (características, datos…) y métodos (funciones, acciones, comportamientos)

Instancia (Real, con datos específicos, "con vida")

Objeto con unas características determinadas proporcionadas por el *"constructor"*

2

Programación Orientada a Objetos

Propiedades

ENCAPSULACIÓN

Todo lo referente a un objeto está aislado del resto.

- Protege contra modificaciones
- Simplifica la programación
- Permite reutilizar código

ABSTRACCIÓN

Oculta la complejidad de los distintos métodos incluidos

- Mejora el mantenimiento

HERENCIA

Como en genética. Se obtienen clases derivadas añadiendo y/o modificando atributos y/o métodos de la clase base.

- Evita repetir código
- Acelera el desarrollo

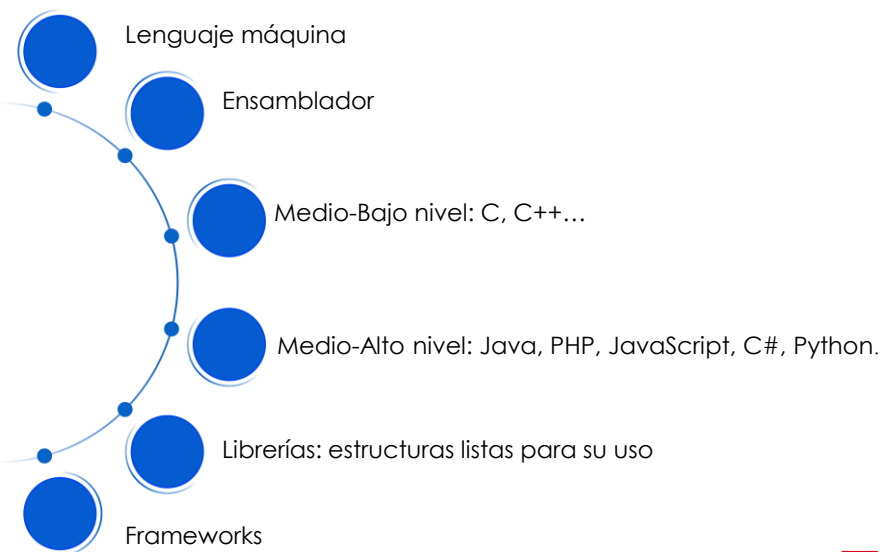
POLIMORFISMO

Si dos clases derivadas tienen atributos y/o métodos añadidos diferentes, se podrán intercambiar a lo largo del proceso de desarrollo del programa y funcionará sin tener que tocar el código..

- Evita estructuras switch, if…
- Simplifica la programación

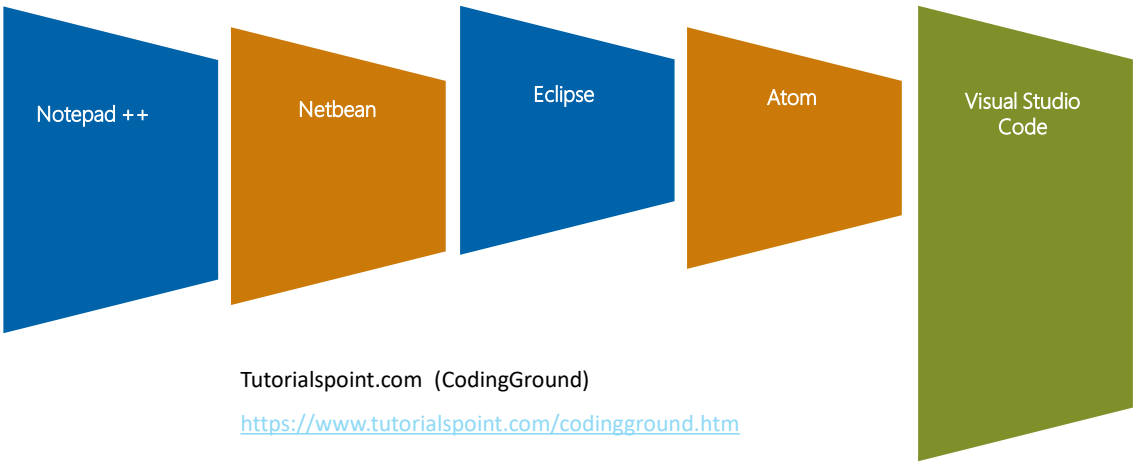
3

Niveles de lenguajes



4

IDE´s
Entornos de Desarrollo Integrado

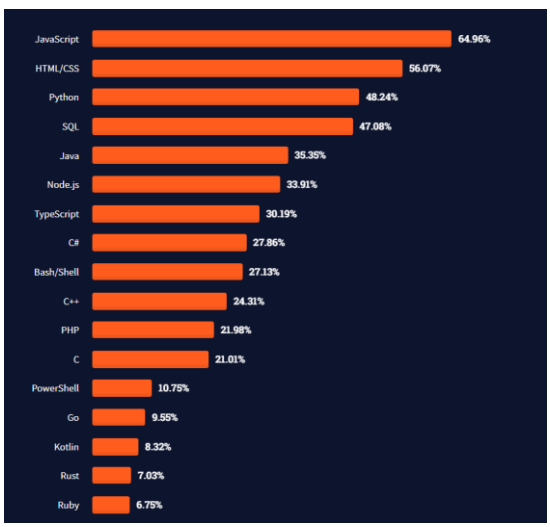


Tutorialspoint.com (CodingGround)

https://www.tutorialspoint.com/codingground.htm

5

Lenguajes más demandados



Ranking 2021 Stack Overflow Developer Survey 2021

Lenguajes de programación: C++, C#, Swift, Java…

Lenguajes de scripting: JS, Shell, Perl, PHP, Python…

6

Control de versiones

Git	Aplicación
GitHub	Plataforma de repositorios
GitLab	Plataforma de repositorios privados fundamentalmente

Descarga desde
https://git-scm.com

Libro Pro Git (S. Chacon, B. Straub)
https://git-scm.com/book/en/v2

7

git

La cantidad de líneas de código que un desarrollador va acumulando plantea el problema del control de cada cambio, cada corrección o fallo detectado, que debe ser anotado para facilitar su manejo: confirmar el cambio, volver atrás si el camino no es el apropiado, controlar la aparición de nuevos errores provocados por los cambios que arreglan otros…y por esto resulta casi imprescindible utilizar los distintos modelos de software de control de versiones que existen hoy día que registran ordenadamente el histórico de un desarrollo.

Si la forma de trabajo es en equipo, conocer qué cambios ha hecho cada cual es fundamental. Las herramientas de control de versiones ofrecen una comparativa de dos revisiones resaltando los datos o apartados modificados, el código añadido, etc. así como su autor, fecha, hora…

Además permiten crear ramificaciones del proyecto (branchs) para trabajar en paralelo hasta averiguar la eficacia del software y devolverla o no (descartarlo) al desarrollo original.

Git es un sistema de control de versiones de tercera generación de tipo DVCS (Distributed Version Control System) que permite separar y combinar operaciones de confirmación para generar las diferentes revisiones, o el que diferentes ramas de un proyecto contengan a su vez diferentes partes.

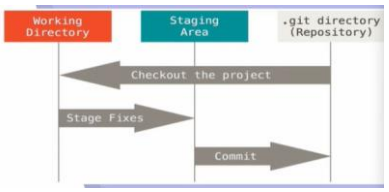
8

Git

Sistema distribuido de control de versiones
Proporciona historial de revisiones
Ramas (branches) y fusiones (merges)
Todo el histórico se descarga desde el servidor a la máquina

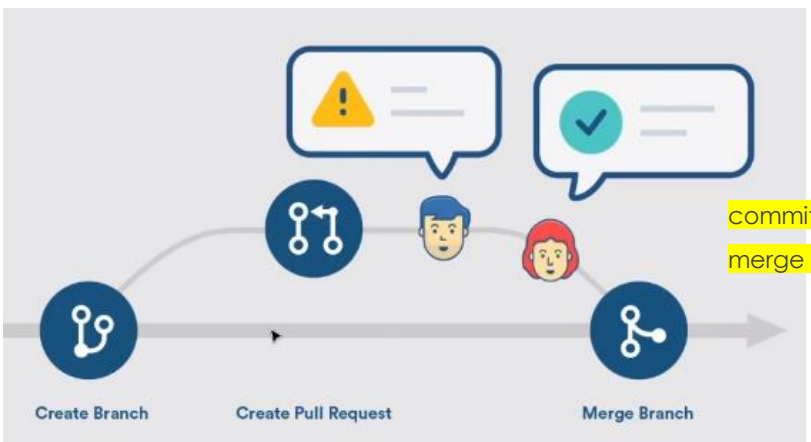
Estados

- Confirmado (commited). Almacenado en el servidor
- Modificado (untracked). Con cambios pero sin confirmar
- Preparado (staged) Intermedia. Repositorio de preparados para confirmar



9

Code Review

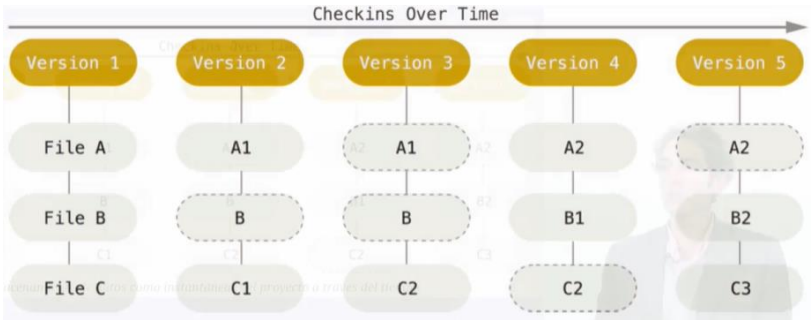


commits = versiones

merge = rama

10

Git



---- No copiado

11

Git (OpenWebinars)

Preparamos el entorno para trabajar con GIT

- Descargamos **GIT for Windows** de aquí:
  `https://github.com/git-for-windows/git/releases`
  - Se integra en el explorador de archivos de windows.
- **Tkdiff** : Herramienta visual comparadora de archivos; Se descarga de aqul:
  `http://www.posoft.de/html/tkdiffMain.html`

Advertencia sobre vimdiff

- vimdiff está incluido en a distro de GIT para las diferencias de ficheros, pero si el binario de tkdiff lo metemos en:
- `c:\archivosdeprograma\git\usr\bin\tkdiff`

Se copia en el directorio /usr/bin que ha creado la instalación de git para que git lo vea

DIFERENCIAS: Advertencia sobre *vimdiff versus tkdiff*

- vimdiff está incluido en a distro de GIT para mostrar las diferencias entre ficheros. Es muy poco visual. Podemos disponer de otra herramienta más intuitiva incluyendo el binario de tkdiff en:

`c:\archivosdeprograma\git\usr\bin\tkdiff`

- Elegimos tkdiff:
  `git config --global diff.tool tkdiff`
  - O bien con **git diftool**
- Con **vimdiff** podemos conmutar de ventana con CTRL+w+TeclaIZq o bien CRTL+w-TeclaDcha

EDICIÓN: Advertencia sobre *vim versus nano*

- vim está incluido en a distro de GIT para editar ficheros. Es muy poco visual. También disponemos de nano.
- `git config --global core.editor nano`
- Con **nano** seleccionar texto es: AltIzq+A y movemos cursores. Cuando haya seleccionado:
  - Si pulso AltIzq+6 lo que hago es COPIAR texto
  - Si pulso CRTL+K lo que hago es CORTAR texto.
  - Si pulso CRTL+U lo que hago es PEGAR texto.

12

Git

Configuración inicial windows

El fichero se encuentra en:

```
$ git config --global user.name "NOMBRE"
$ git config --global user.email DIRECCION@DOMINIO.COM
$ git config --global diff.tool tkdiff
$ git config --global credential.helper manager
$ git config --global core.editor nano
```


13

Git

- git init,
- git status,
- git add,
- git commit,
- git ignore,

- git status
- git diff
- git add
- git log
- git show

Fuente: Cuso de Git Openwebinars (Juan Carlos Rubio)

14

Git

CREAR Y CLONAR

Clonar repositorio existente
git clone ssh://usuario@url.com/repo.git

Crear repositorio local
git init

Clonar repositorio local
git clone /direccion/de/repositorio

RAMAS

Crear nueva rama o branch
git checkout -b <rama>

Cambiar a la rama master
git checkout master

Eliminar una rama local
git branch -d <rama>

Eliminar una rama remota
git push origin --delete <rama>

Hacer push de la rama al repo remoto
git push origin <rama>

AÑADIR Y ELIMINAR

Añadir cambios de un archivo
git add nombrearchivo.extension

Añadir todos los archivos con cambios
git add .

Eliminar archivo
git rm nombrearchivo.extension

FUSIÓNAR CÓDIGOS

Fusionar cambios de una rama
git merge <rama>

Ver diferencias entre dos ramas
git diff <ramaOrigen <RamaDestino>

COMMIT Y SINCRONIZACIÓN

Añadir commit de los cambios
git -m "mensaje de commit"

Hacer push de los cambios al repo remoto
git push origin master

Conectar el repo local con el remoto
git remote push origin <server>

Actualizar cambios en local desde remoto
git pull

Ver cambios en el directorio
git status

Cambios en los archivos de seguimiento
git diff

Crear tag de un commit
git tag version <id del commit>

Ver historico de cambios
git log

15

Git Repositorio local

Creación estructura inicial del repositorio

MINGW64:/c/Users/jjgl/Desktop/escenario1

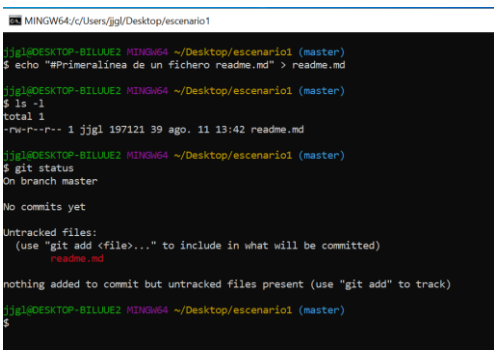
```
jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop
$ mkdir escenario1

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop
$ cd escenario1/

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1
$ git init
Initialized empty Git repository in C:/Users/jjgl/Desktop/escenario1/.git/

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ ls -la
total 16
drwxr-xr-x 1 jjgl 197121 0 ago. 11 13:38 ./
drwxr-xr-x 1 jjgl 197121 0 ago. 11 13:37 ../
drwxr-xr-x 1 jjgl 197121 0 ago. 11 13:38 .git/

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
```

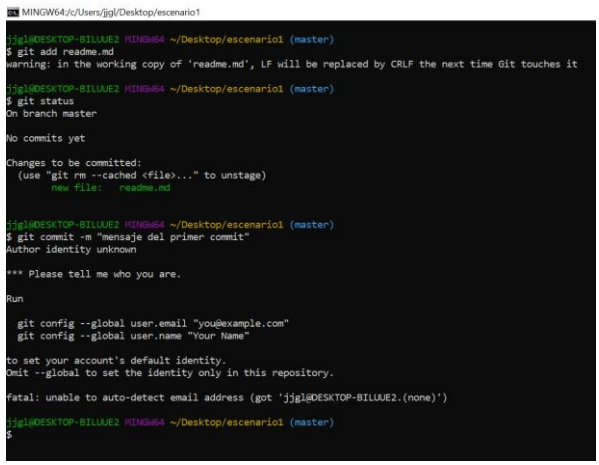
```
jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ ls -lart .git/
total 11
drwxr-xr-x 1 jjgl 197121   0 ago. 11 13:38 ../
-rw-r--r-- 1 jjgl 197121  73 ago. 11 13:38 description
drwxr-xr-x 1 jjgl 197121   0 ago. 11 13:38 info/
drwxr-xr-x 1 jjgl 197121   0 ago. 11 13:38 hooks/
drwxr-xr-x 1 jjgl 197121   0 ago. 11 13:38 refs/
-rw-r--r-- 1 jjgl 197121  23 ago. 11 13:38 HEAD
drwxr-xr-x 1 jjgl 197121   0 ago. 11 13:38 objects/
-rw-r--r-- 1 jjgl 197121 112 ago. 11 13:38 config
drwxr-xr-x 1 jjgl 197121   0 ago. 11 13:38 ./

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$
```


16

Git Repositorio local

Primer fichero, estado y primer commit





17

Git Repositorio local

Configuración previa necesaria

```
jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ git config --global user.email "jjgarcia@vivantia.net"

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ git config --global user.name  "jjgl"

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ git commit -m "mensaje del primer commit"
[master (root-commit) 4f7cc2a] mensaje del primer commit
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$
```


18

Git Repositorio local

Trabajo con varios ficheros y directorios .gitignore

```
MINGW64:/c/Users/jjgl/Desktop/escenario1

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ echo "fichero 1" > fichero1

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ echo "fichero 2" > fichero2

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ echo "fichero 3" > fichero3

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ echo "fichero 4" > fichero4

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$ ls -l
total 5
-rw-r--r-- 1 jjgl 197121 10 ago. 11 14:01 fichero1
-rw-r--r-- 1 jjgl 197121 10 ago. 11 14:01 fichero2
-rw-r--r-- 1 jjgl 197121 10 ago. 11 14:01 fichero3
-rw-r--r-- 1 jjgl 197121 10 ago. 11 14:02 fichero4
-rw-r--r-- 1 jjgl 197121 39 ago. 11 13:42 readme.md

jjgl@DESKTOP-BILUUE2 MINGW64 ~/Desktop/escenario1 (master)
$
```

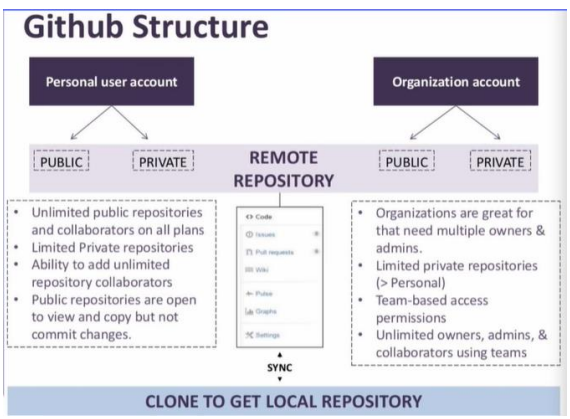
.gitignore incluye máscaras de extensión

```
Lucia@LAPTOP-LVO1PEG0 MINGW32 ~/Desktop/test/escenario1 (master)
$ git status
On branch master
nothing to commit, working tree clean

Lucia@LAPTOP-LVO1PEG0 MINGW32 ~/Desktop/test/escenario1 (master)
$ git log --oneline
00a01dc (HEAD -> master) revision 2 con el .gitignore
7651c55 mensaje de mi primer commit

Lucia@LAPTOP-LVO1PEG0 MINGW32 ~/Desktop/test/escenario1 (master)
$ ls -l
total 5
-rw-r--r-- 1 Lucia 197121 11 dic.  4 19:18 ficch1.tmp
-rw-r--r-- 1 Lucia 197121 11 dic.  4 19:18 ficch2.tmp
-rw-r--r-- 1 Lucia 197121 11 dic.  4 19:18 ficch3.tmp
-rw-r--r-- 1 Lucia 197121 11 dic.  4 19:19 ficch4.tmp
-rw-r--r-- 1 Lucia 197121 40 dic.  4 19:09 readme.md
```


19

GitHub

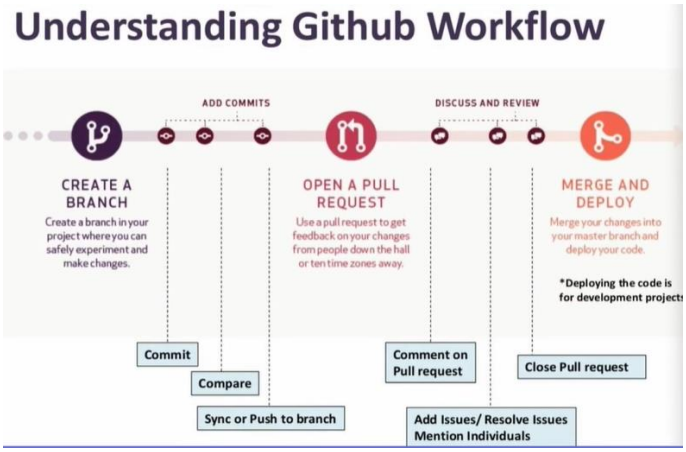
Estructura



20

GitHub

Flujo de trabajo



21

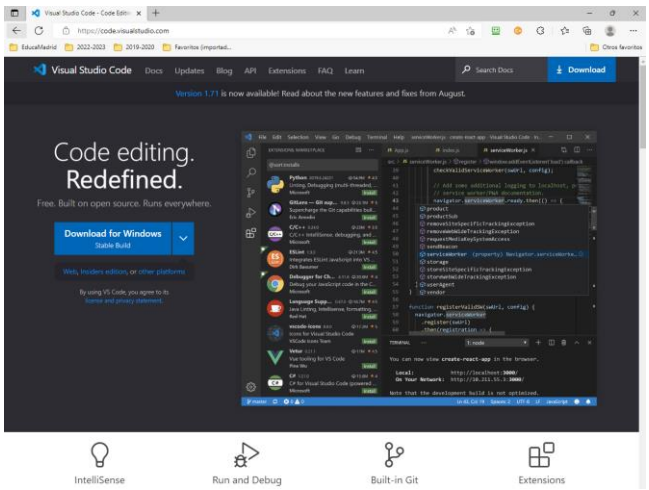
Herramientas durante el curso

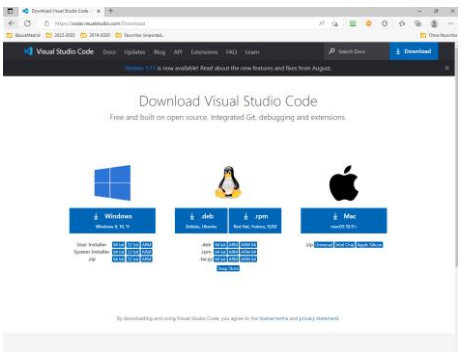
- Visual Studio Code (+ plugin)
- GitHub
- Navegadores Mozilla, Chrome, Edge
- Notepad ++
- Online Javascript Editor (tutorialspoint.com)

22

VSC

Visual Studio Code





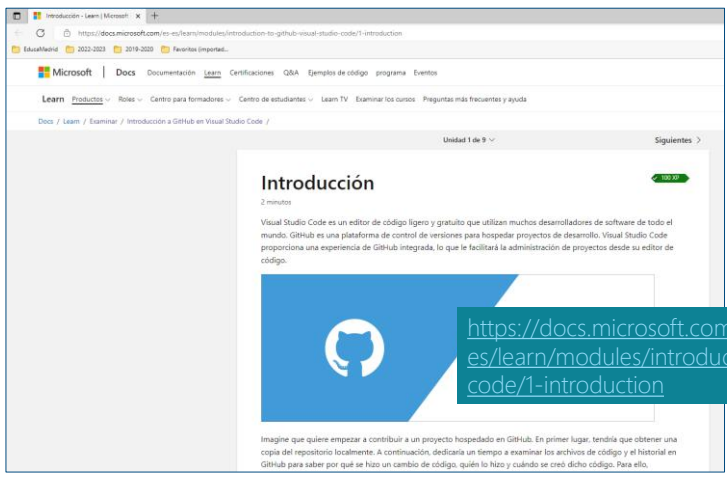
23

VSC y GitHub

1. Crear cuenta en GitHub
2. Crear nuevo repositorio
3. Autorizar a VSC a GitHub
4. Instalar plugins necesarios en VSC si no se tienen (GitHub Pull Request and Issues)

24

VSC y GitHub
Curso básico



https://docs.microsoft.com/es-es/learn/modules/introduction-to-github-visual-studio-code/1-introduction