UD2-Desarrollo en Entorno Cliente

GitHub - Patrones de Diseño

Manuel Moya Vadillo - 2ºDAW

https://creativecommons.org/licenses/by-nc-sa/4.0/

Manuel Moya Vadillo - 2ºDAW

Índice

Manuel Moya Vadillo - 2ºDAW

01

Paradigms

Main features & possible limitations.

Topics

01

Structure Programming

Efficiency and readability

→

02

Functional Programming

Composition of Functions

→

03

POO

Representation of elements

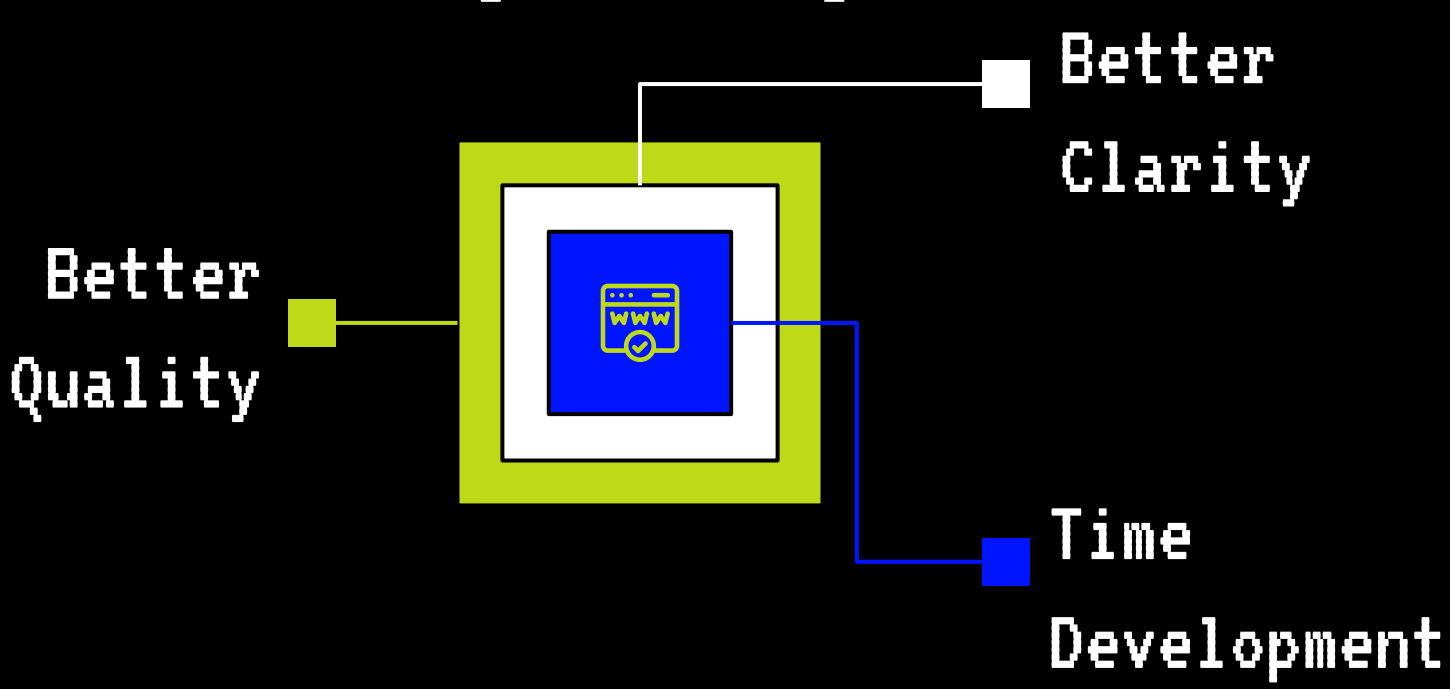
→

04

Reactive Programming

Asynchronous data

Structure Programming



Control Structures

Sequential structure

Calls to language instructions or programmer functions.

Conditional structure

Executes a structure if a Boolean condition is met.

Iterative structure

Executes a structure over and over again if a boolean condition is met.

Topics

01
Structure Programming

Efficiency and readability

→

02
Functional Programming

Composition of Functions

→

03
POO

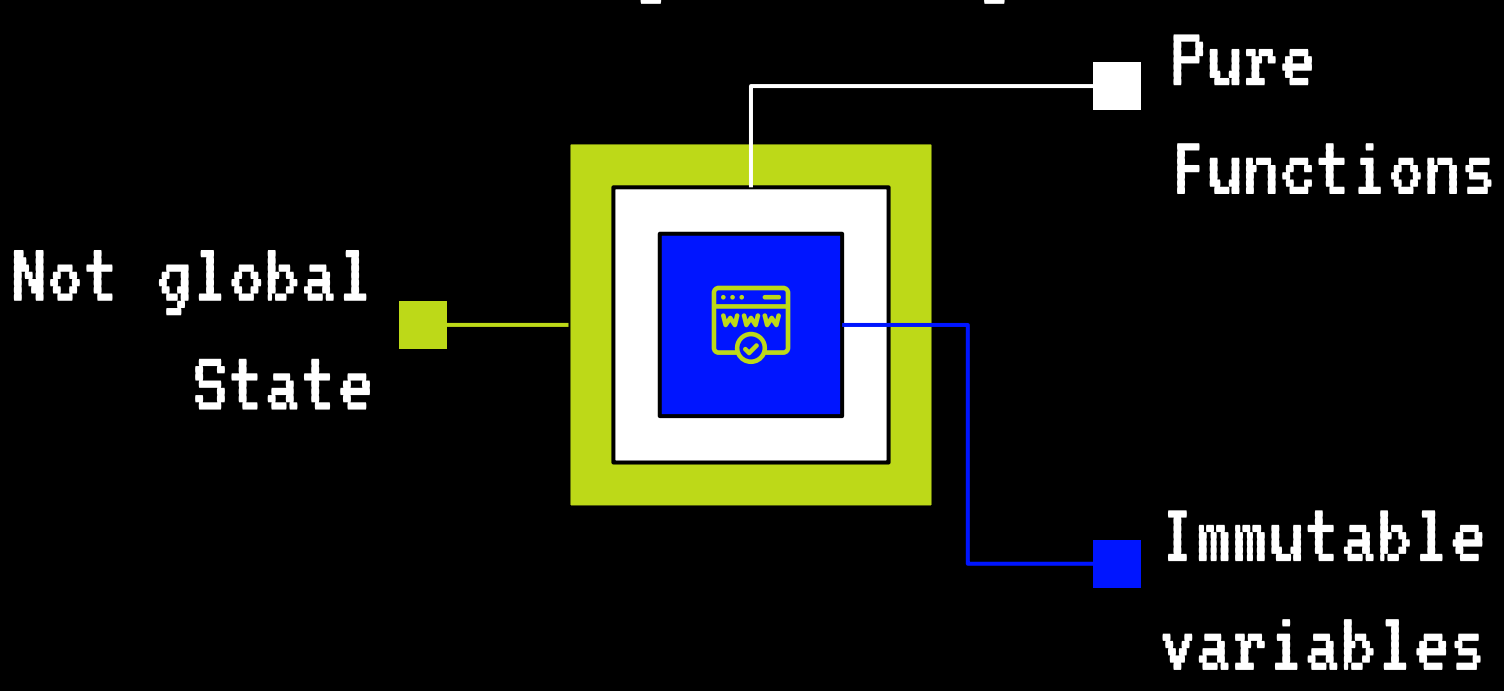
Representation of elements

→

04
Reactive Programming

Asynchronous data

Functional Programming



Limitations

Modular programming issues

- Serious mistakes
- Not suitable for all tasks

Topics

01
Structure Programming

Efficiency and readability

→

02
Functional Programming

Composition of Functions

→

03
POO

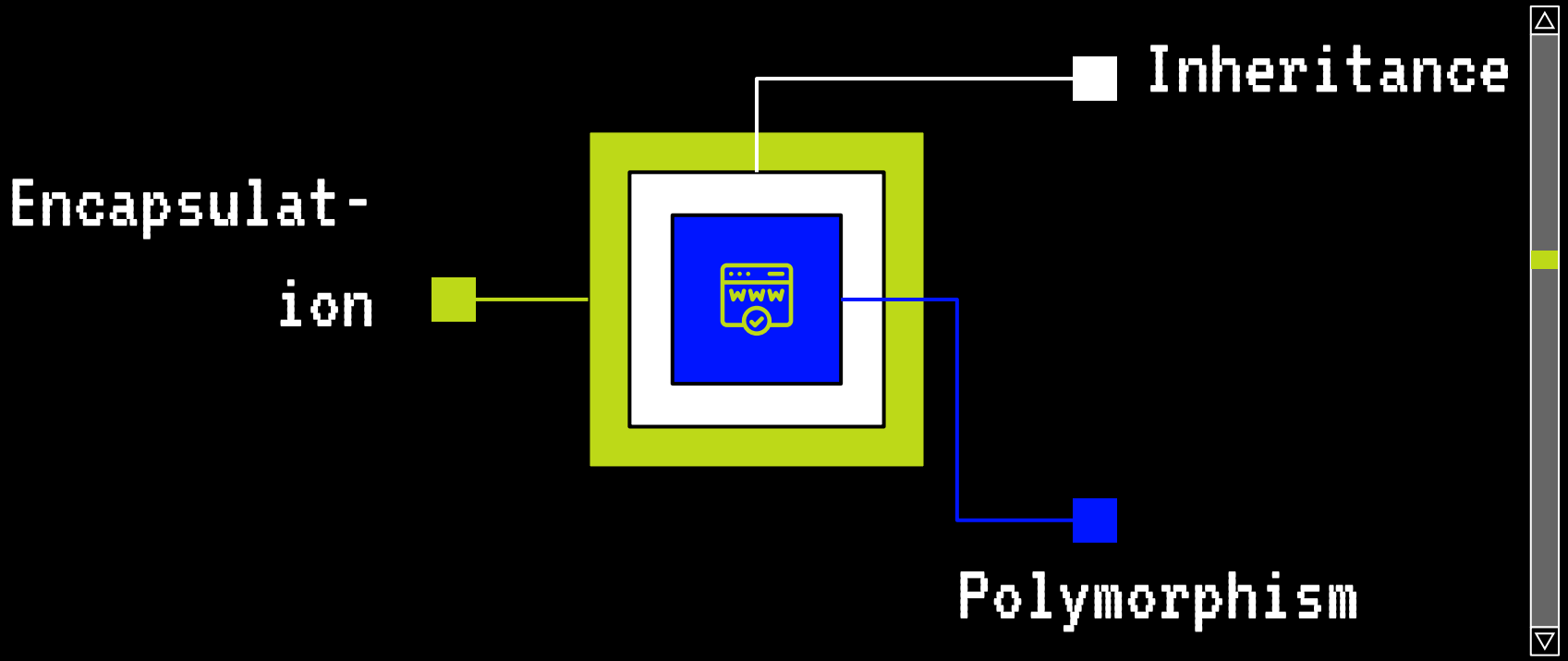
Representation of elements

→

04
Reactive Programming

Asynchronous data

POO



Limitations

- There is not a unique way to resolve the problem. This may can lead to different interpretation of the solutions.
- It is required a extensive documentation to reach the solution.

Topics

01
Programación Estructurada

Efficiency and readability

→

02
Functional Programming

Composition of Functions

→

03
POO

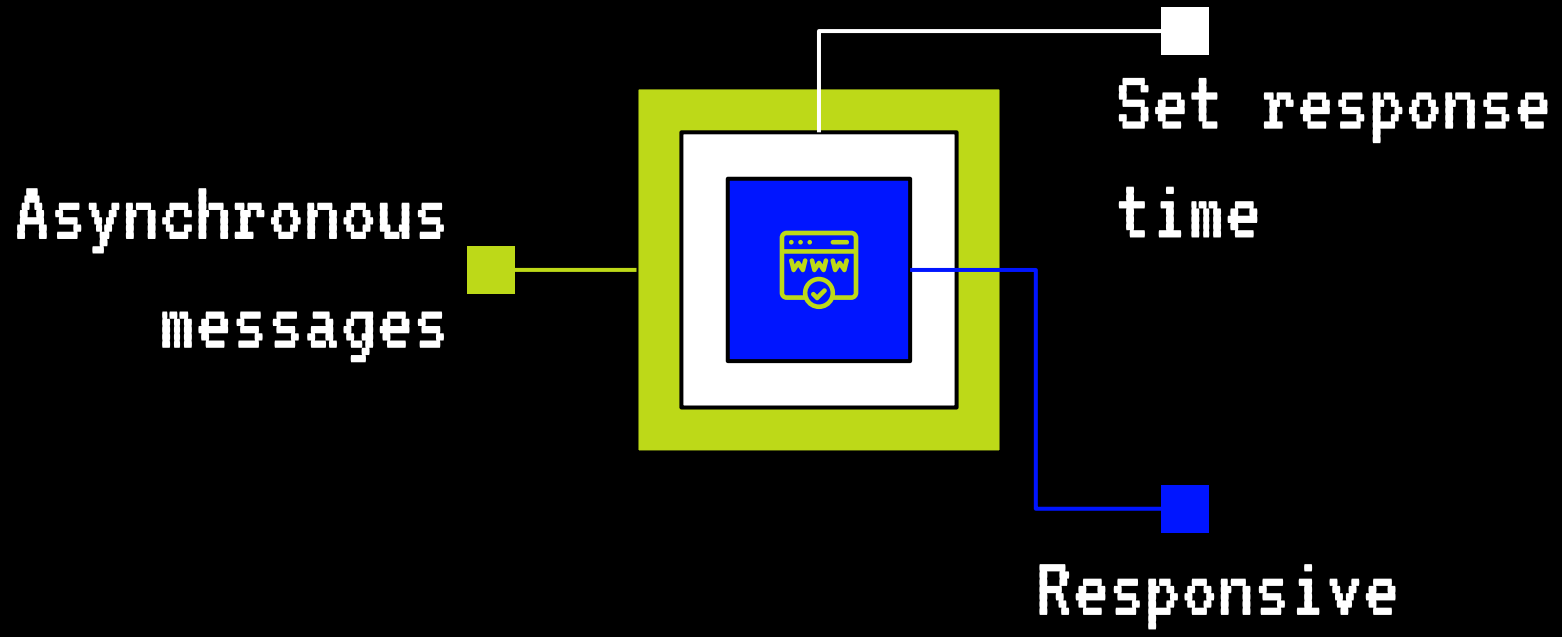
Representation of elements

→

04
Reactive Programming

Asynchronous data

Reactive Programming



This paradigm it is used to fix waste of the CPU, overuse of memory and inefficiency of blocking interactions.

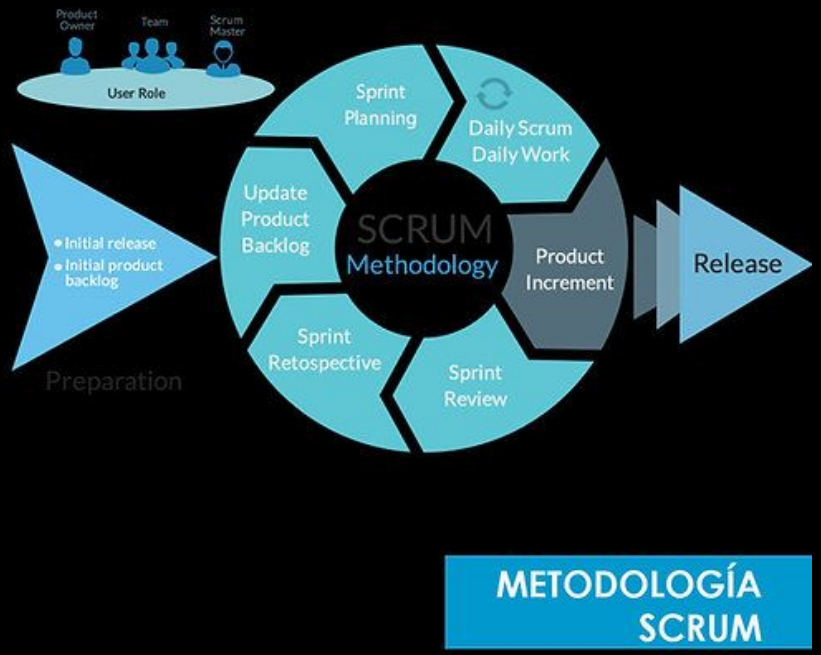
Reactive paradigm

	Programming language
Estructurada	Java, Python, C, C++
Funcional	JS, Visual Basic, LISP
Orientada a objetos	C++, Sharp, PHP, Java, JS, Perl
Reactiva	Scala, Akka, Kubernetes, Kafka, Docker, etc

02

Agile methodologies

XP, Scrum, Kanban, Agile Inception & Design Sprint.



Definition:

Allow to adapt the work to the conditions of the project, obtaining flexibility and immediate in order to change the project.

Topics

01
XP

02
Scrum

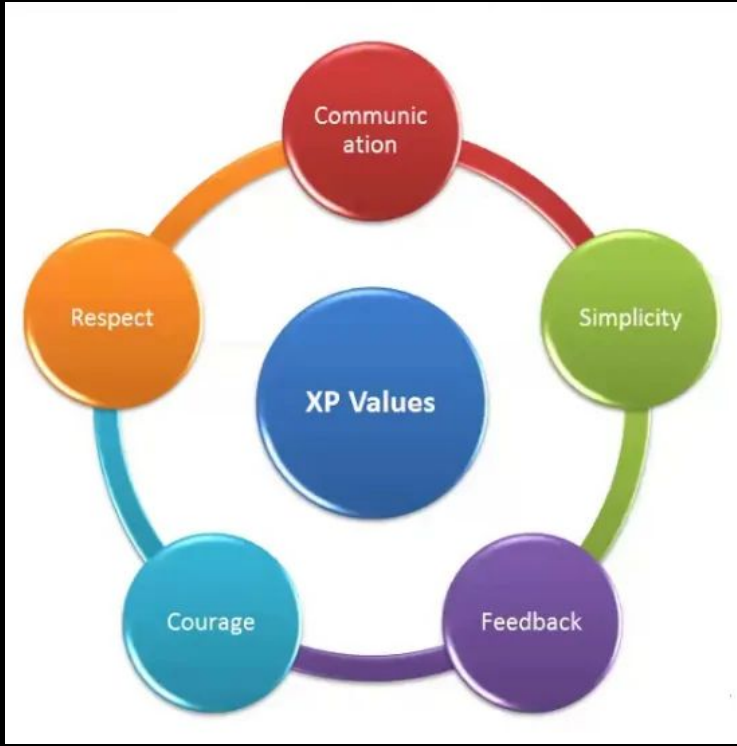
03
Kanban

04
Agile Inception

05
Design Sprint

Methodology - XP

XP is a methodology that have as main objective create high quality systems. This systems must be done with the client interaction. Constant tests and short development cycles. This methodology approximate us to the best quality. In order to get the perfect product, we must do as many changes as we can. If we can gestionate all the changes, the client would be satisfied.



Features

Constant contact with client

Quick response

Schedule of activities

Success factors

Software is above of all

Work - Team

Client: provides the needs and priorities.

Programmers: set durations and times.

Testers: compares the client needs with the program.

Tracker: controls and trace each part of the program

Coach: advise the client and the programmers

MANAGER: coordinate communications between client and the work-team.

Phases

1. Planning: The program is broken down into mini versions which will be reviewed. Every two weeks it must have been created a useful and functional model.

2. Design: The prototype of the software would be created. It would work with a simple code in order to make it works.

3. Coding: The coding must be in teams of two. Sometimes, the co-worker have to be change to make the code more universal.

4. Tests: This part is extremely important due to that the projects are normally really short. So the tests must be constant and automatic.

Topics

01
XP

02
Scrum

03
Kanban

04
Agile Inception

05
Design Sprint

Methodology - Scrum

Scrum is a methodology that allow collaborative work between the team. Scrum encourage the teams to learn through the experience, to organize themself and to improve continually.

Features

Flexible project

Members transparency

Inspection

Change Adaptation

Iterative development

Work - Team

Product owner: the only one who talks with the client (Only one). Manages the product backlog

Scrum Master: takes the responsibility of the scrum techniques. The issues of the team are removed by him.

Development team: In charge of make the development of the task that the product owner prioritizes.

Phases

1. Sprint planning: The team meets and decide what they are going to approach. They take each topic from the backlog and how it would be maden.
2. Daily Meeting: The meets during 15 minutes and explain what they have done, what they are going to do and what issue they have. Here you can make adapt the project to the changes.
3. Sprint Review: The client receive a final result of the product. If the client likes it, he would valid it. Unless he likes it, he would propose new changes. The changes would be added to the backlog by the product owner.
4. Sprint Retrospective: It is the last phase. Here the whole team meets to see the performance of the product and finish all the changes that they should be done the next day.

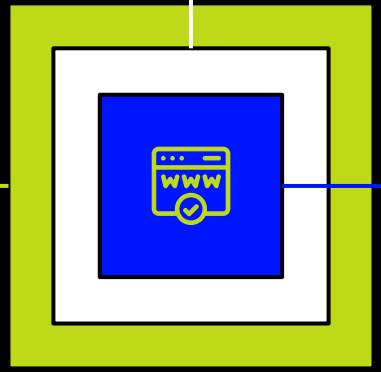
Artifacts

Sprint Backlog

Where the team chose one chore at the sprint planning

Product Backlog

List of chores that involve the whole project



Increment

Merge the finished chores

Topics

01
XP

02
Scrum

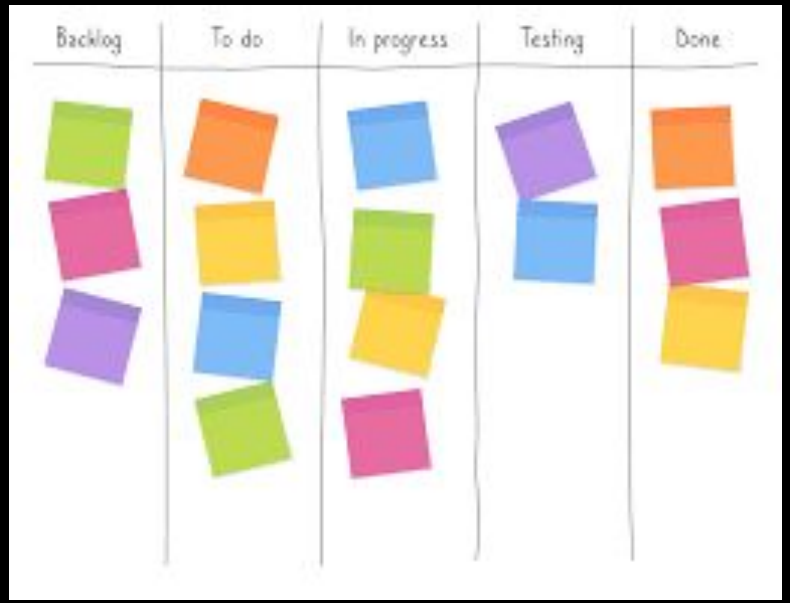
03
Kanban

04
Agile Inception

05
Design Sprint

Methodology - Kanban

It is a very popular way to control the flow of work for define, manage and improve the services of the product. This methodology helps you to display in a better way the product, maximize the efficiency and improve continually.



Features

Control Lists

Expiration dates

Card assignments

Card tags

Card Notes

Principal values

- Display: development moment of the project.
- Priority: the principal tasks must be in order.
- Continue upturn: the competitiveness is lost without updates.
- Same rol: everyone's involvement.
- Quality: quality before speed.

Phases

1. Board tasks: This phase consist on share the state of all the tasks: Pending, in process or Finished. All the team must have access to the tasks.

2. Product display: Consist on add all the info about that task in order to complete it.

3. Stop before Start: the slogan of kanban is “Stop starting, start finishing”. Finish the task before start other.

4. Flow control: Allow us to compile all the information of all the tasks.

Topics

01
XP

02
Scrum

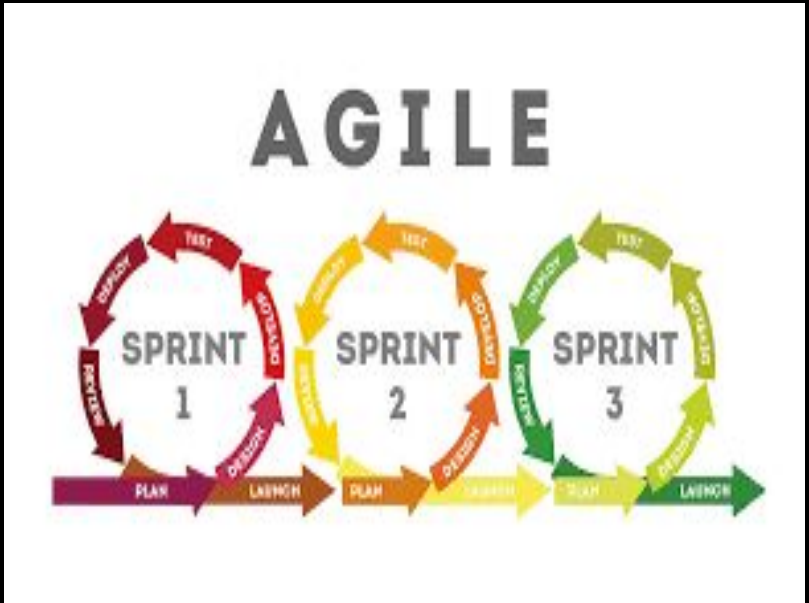
03
Kanban

04
Agile Inception

05
Design Sprint

Agile Inception

It is a very popular way to control the flow of work for define, manage and improve the services of the product. This methodology helps you to display in a better way the product, maximize the efficiency and improve continually. For it to be really useful, it requires the involvement of all the people who will participate in its development. In this way, they can present their ideas and create a joint perspective. As well as the generation of a collaborative informative and documentary process.



Features

Workers and interactions

Multidisciplinary

Customer Collaboration

Response to change

Multidisciplinary

Phases

1. Reasons to carry out this project:
   a. The reasons why we are here, define who our customers are and why we want to create the product.
   b. How the product display would be: what will it have so that the customer wants to buy it just by seeing it.
   c. We do not own the product.
2. We will have to think about aspects related to time, budget, technique:
   a. Devise the solution
   b. Future issues and mark the times.

Topics

01
XP

02
Scrum

03
Kanban

04
Agile Inception

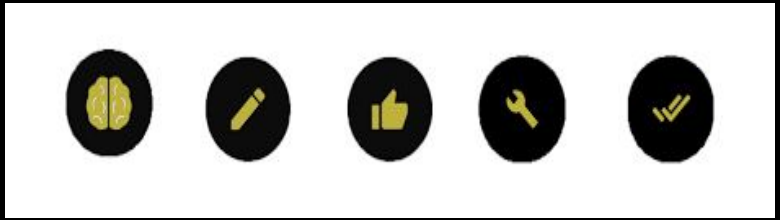
05
Design Sprint

Design Sprint

Design Sprint helps speed up the design process and provides valuable and relevant information that ensures design success.

This type of methodology helps teams work together to solve a specific problem and provide solutions that will be tested with users.

Considerably speeds up decision making and reduces project risk. The purpose is to build a testable prototype with future clients or users.



Features

Fastest-Growing

User - Tests

Reduce risks

Exhausted team

Equitable contribution

Phases

1. Investigate and Define: In this phase, the documentation resulting from the investigations must be provided as:

    - Empathy maps.
    - User Journey.
    - Interview.
    - Surveys.
    - SWOT analysis.

Phases

2. Sketch: Each of the team members must design a solution to a problem. Basically you have to draw a quick solution on paper, without other factors affecting the result

Phases

3. Determinate: In this phase a decision must be made about which idea (or ideas) are going to be carried out in the prototype phase. It is very important to decide how the prototype will be carried out to discover what problems could generate

Phases

4. Prototype: In this phase, you begin to create a prototype of the product. UX and UI begin to design the prototype with High Definition, to which later the necessary animations will be introduced so that the basic functionalities that solve the problem can be understood.

You only have one day to complete this task, so at this point things get serious. At the same time that the prototype is made, the research or research team must specify the schedules for the tests or tests with users.

Phases

5. Validate: It is the last phase and the most important. Tests with Users are carried out. It will be necessary to gather up to a maximum of 20 users and a minimum of 6 to carry out the tests with the prototype.

03

Design Patterns

Abstract Factory & Adapter.

The design patterns are differents ways to fix normal problems in web development.

What is it?

Topics

01

Abstract Factory
& Adapter.

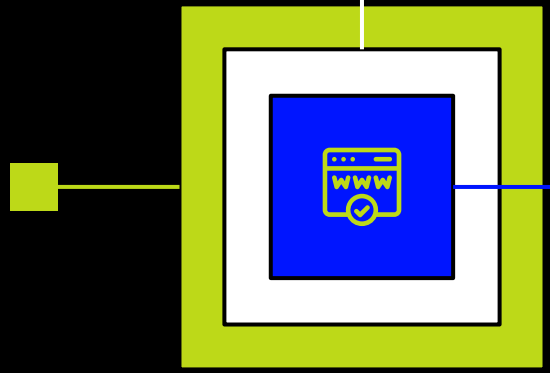
02

Other Patterns

Abstract Factory

Abstract Factory is a creational design pattern that gives us the option to create different objects from different families that are interrelated.

DrawBacks



Mixed objects

Independent Objects

Change code

How to apply

Interfaces

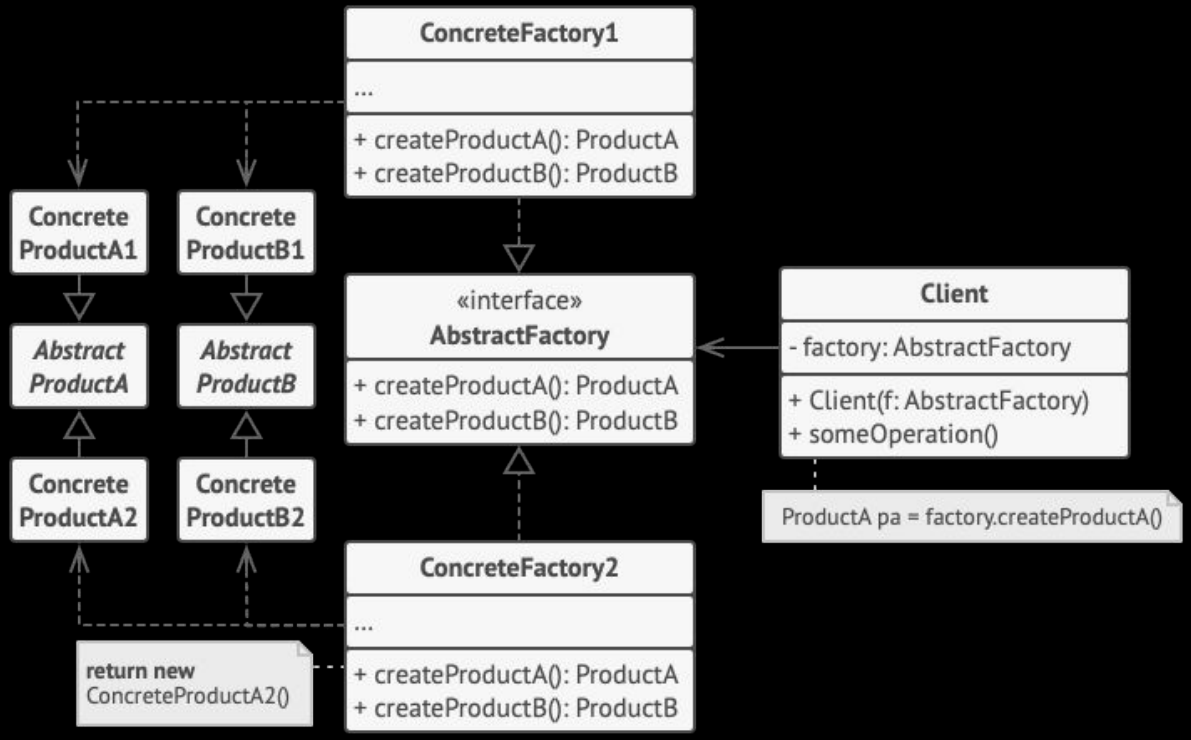
Generic

List methods

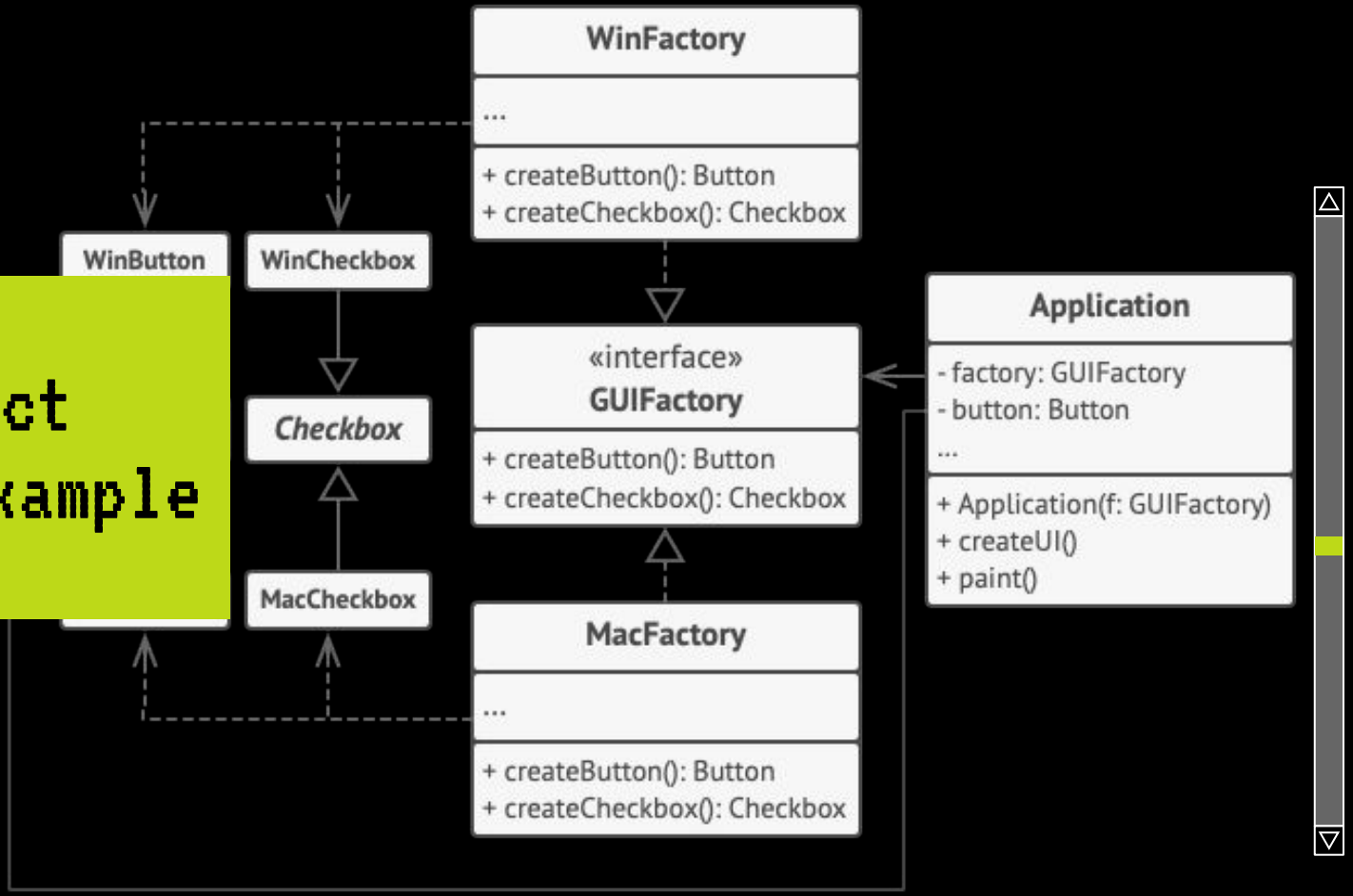
Object properties

Obj Interfaces

For each object

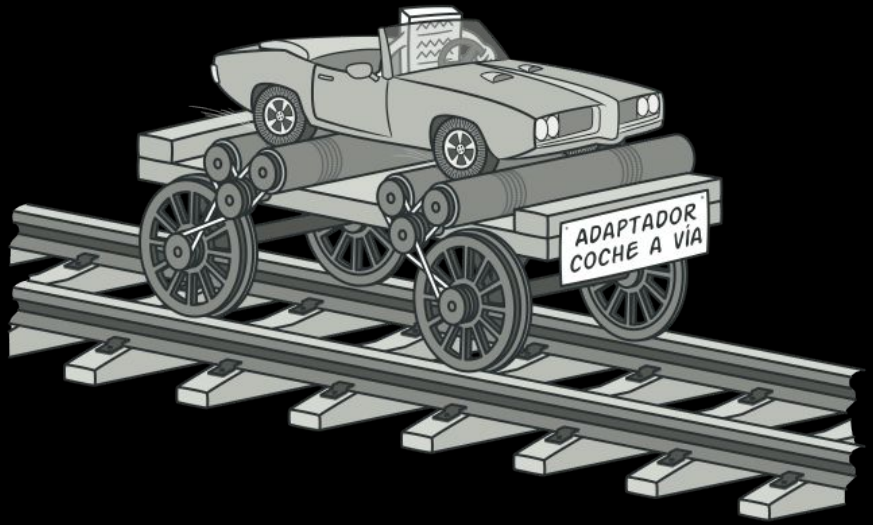

ConcreteFactory1
...
+ createProductA(): ProductA
+ createProductB(): ProductB
Concrete ProductA1
Concrete ProductB1
Abstract ProductA
Abstract ProductB
Concrete ProductA2
Concrete ProductB2
«interface»
AbstractFactory
+ createProductA(): ProductA
+ createProductB(): ProductB
Client
- factory: AbstractFactory
+ Client(f: AbstractFactory)
+ someOperation()
ProductA pa = factory.createProductA()
ConcreteFactory2
...
+ createProductA(): ProductA
+ createProductB(): ProductB
return new ConcreteProductA2()

Abstract Factory Example



Uses

When your code needs to work with several families of related products, but you don't want it to depend on the specific classes of those products, because you either don't know about them beforehand or simply want to allow for future extensibility. However, this may tend to make the code more difficult.



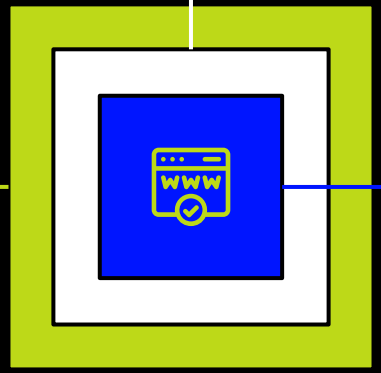
Adapter

Adapter is a structural design pattern that allows collaboration between objects with incompatible interfaces, that is, we can mix objects that have nothing to do with each other.

Pros

Can Mixed objects

Incompatible interfaces



Adapts anything

Adapter Example

RoundHole

- radius: int

+ RoundHole(radius: int)
+ getRadius(): int
): bool

RoundPeg

- radius: int

+ RoundPeg(radius: int)
+ getRadius(): int

+ SquarePeg(width: int)
+ getWidth(): int

SquarePegAdapter

- peg: SquarePeg

+ SquarePegAdapter(peg: SquarePeg)
+ getRadius(): int

return peg.getWidth() * sqrt(2) / 2

Topics

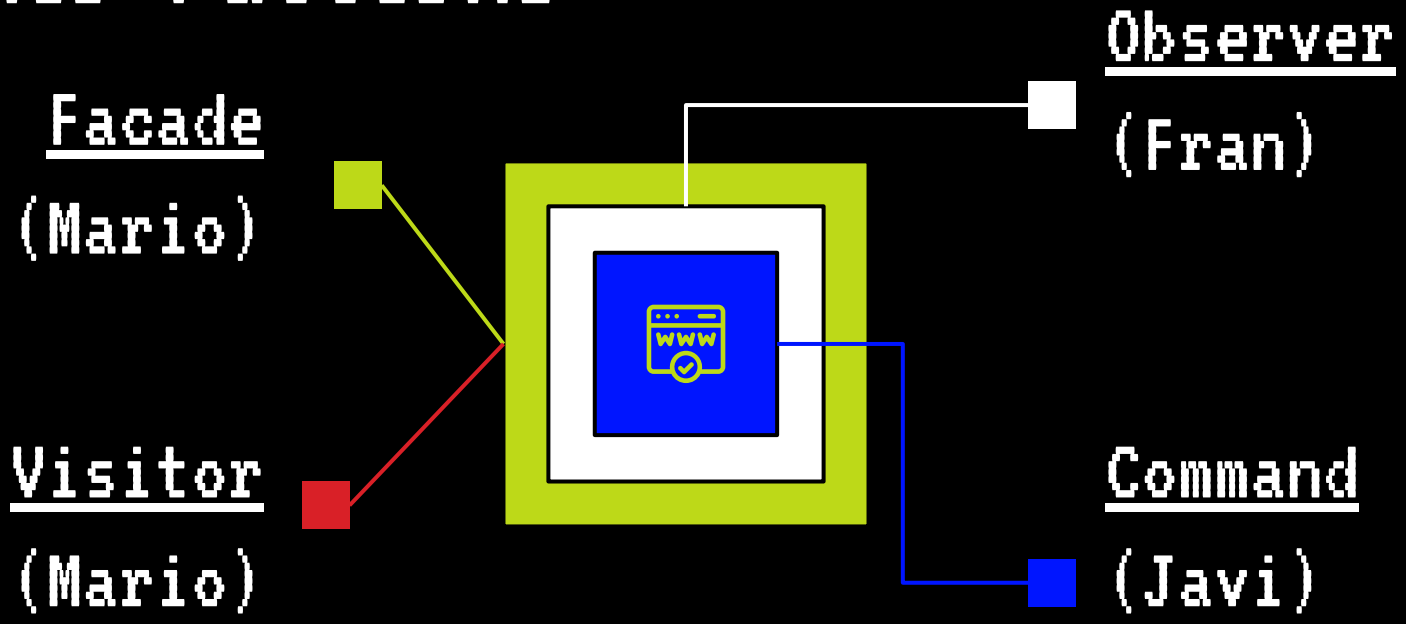
01

Abstract Factory
& Adapter.

02

Other Patterns

Other Patterns



04 GitHub

Uses, repositories, branches & tags.

GitHub is a online repository where many programmers keep up their works giving the opportunity to a lot of people to interact with the code.

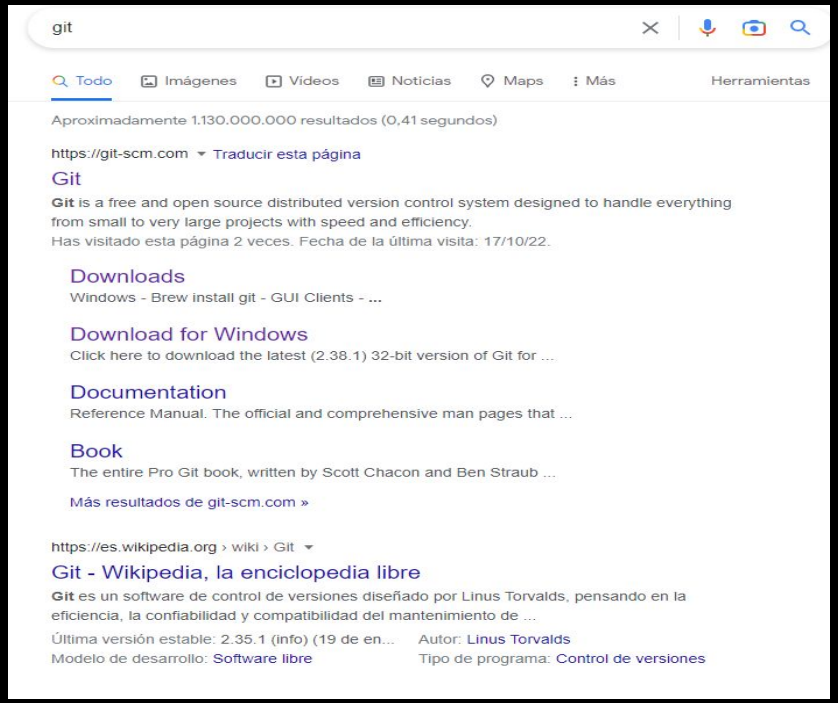
What is it?

Topics

01

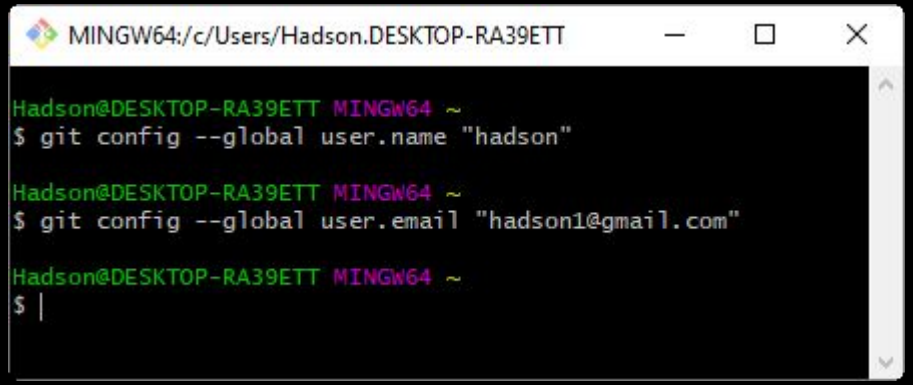
Exercises

E0



For download the git console, we must search on the navigator “git”. After, we have to click on the first link and we would be able to download it.

E0



After we have downloaded git console, we must go at git bash and set the global configuration with two commands:

1. Git config - global user.name “”
2. Git config - global user.email “”

E1

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub
$ mkdir E_GitHub

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub
$ cd E_GitHub/

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub
$ ls

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub
$ git init
Initialized empty Git repository in C:/Users/porta/Documents/manuel/DAW/ILERNA/g
itHub/E_GitHub/.git/
```

To create a local repository, through the git console, we must use the command “git init ‹repository name›”

E1

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ ls
hp01.html  hp02.html  hp03.html

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hp01.html
        hp02.html
        hp03.html

nothing added to commit but untracked files present (use "git add" to track)

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git add -A
warning: in the working copy of 'hp01.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'hp02.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'hp03.html', LF will be replaced by CRLF the next time Git touches it

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git commit -m "Archivos añadidos. Subida1"
[master (root-commit) 4d27381] Archivos añadidos. Subida1
 3 files changed, 37 insertions(+)
 create mode 100644 hp01.html
 create mode 100644 hp02.html
 create mode 100644 hp03.html

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ |
```

Once our files are inserted, we use the following commands:

- git add -A → add changes from working area to staging area
- git commit -m → add staging area changes to repository
- git status → to check if the status of our repository has changed.

E1

Within github we have 3 logical areas:

- Working copy
- Staging area
- Repository

When we add files to our repository, we are unknowingly adding them to the working copy. In order to pass it to the staging area we must use the git add -A command. to check the status of both we use git diff, but since we have not made any changes, they are exactly "same", nothing will appear. However, if we change the content of our files or add a new one and use git diff it will show us the changes.

E1

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git diff

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ nano hp01.html

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hp01.html

no changes added to commit (use "git add" and/or "git commit -a")

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git diff
warning: in the working copy of 'hp01.html', LF will be replaced by CRLF the next time Git touches it
diff --git a/hp01.html b/hp01.html
index 82d7d93..0666e8d 100644
--- a/hp01.html
+++ b/hp01.html
@@ -5,6 +5,7 @@
  </head>
  <body>
 <h1>Harry Porlotes y La Red Neuronal</h1>
+<p>Esto es un cambio para comprobar el git diff</p>
 <p>El niño huérfano Harry vive con sus tíos, que lo tratan muy mal (le hacen utilizar Internet Noexplorer).</p>
 <p>Harry tiene unas gafas redondas a lo John Lennon y una cicatriz de una manzana mordida en la frente.</p>
 <p>Un día, aparece Juan Nieve, el del Muro, y le dice que él también puede ser un programador.

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ |
```

E1

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git reset --soft

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hp01.html

no changes added to commit (use "git add" and/or "git commit -a")

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git diff
warning: in the working copy of 'hp01.html', LF will be replaced by CRLF the next time Git touches it
diff --git a/hp01.html b/hp01.html
index 82d7d93..0666e8d 100644
--- a/hp01.html
+++ b/hp01.html
@@ -5,6 +5,7 @@
  </head>
  <body>
 <h1>Harry Porlotes y La Red Neuronal</h1>
+<p>Esto es un cambio para comprobar el git diff</p>
 <p>El niño huérfano Harry vive con sus tíos, que lo tratan muy mal (le hacen utilizar Internet Noexplorer).</p>
 <p>Harry tiene unas gafas redondas a lo John Lennon y una cicatriz de una manzana mordida en la frente.</p>
 <p>Un día, aparece Juan Nieve, el del Muro, y le dice que él también puede ser un programador.

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git reset --hard
HEAD is now at 4d27381 Archivos añadidos. Subida1

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git status
On branch master
nothing to commit, working tree clean

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$
```

There is a way to undo a commit:

- git reset : If you want to keep the changes we write git reset - -soft and if we don't want to keep them git reset - -hard.

To undo an add it would be git reset HEAD

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git reset HEAD
Unstaged changes after reset:
M       hp01.html

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hp01.html

no changes added to commit (use "git add" and/or "git commit -a")

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$
```

E1

To show the commits made, we must use the git log command:

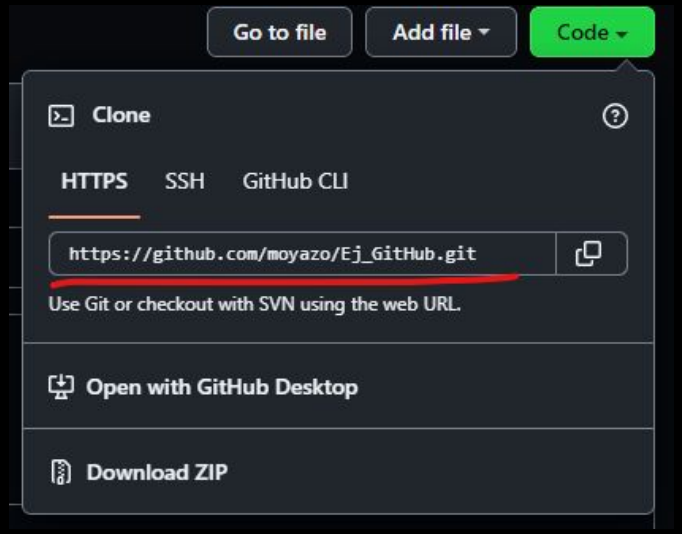
```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git log
commit 043b89f55a3dee89d8a8b6606ae23ffbc81eacda (HEAD -> master)
Author: Manuel Moya <manuel.moyavadillo2@gmail.com>
Date:   Sat Oct 29 14:11:06 2022 +0200

    subida 1.

commit 4d273812220bac33d19532f33e8a8346d7887c8d
Author: Manuel Moya <manuel.moyavadillo2@gmail.com>
Date:   Sat Oct 29 13:10:12 2022 +0200

    Archivos añadidos. Subida1
```

E1

To create a tag we will use the command git tag -a ‹labelname› -m “message”. To list all tags we will use git tag and to remove tags git tag --delete ‹tagName›.

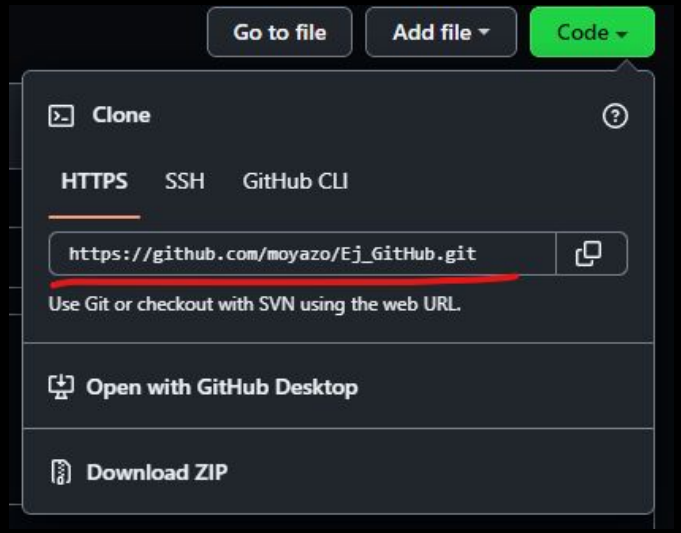
```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git tag -a Trilogía -m "Esta es la etiqueta Trilogía"

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ gi tag
bash: gi: command not found

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git tag
Trilogía

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git tag --delete Trilogía
Deleted tag 'Trilogía' (was 3792569)

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ git tag

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/E_GitHub (master)
$ |
```

E2



To clone a repository from your github account you must use the git clone URL command.

E2



To clone a repository from your github account you must use the git clone URL command.

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub
$ git clone https://github.com/moyazo/Ej_GitHub.git
Cloning into 'Ej_GitHub'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub
$ |
```

E2

```
porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (main)
$ git branch NBranch

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (main)
$ git branch
  NBranch
* main

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (main)
$ git branch -m MBranch

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (MBranch)
$ git branch
* MBranch
  NBranch

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (MBranch)
$ git checkout NBranch
Switched to branch 'NBranch'

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (NBranch)
$ git branch -D NBranch
error: Cannot delete branch 'NBranch' checked out at 'C:/Users/porta/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub'

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (NBranch)
$ git checkout MBranch
Switched to branch 'MBranch'
Your branch is up to date with 'origin/main'.

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (MBranch)
$ git branch -D MBranch
error: Cannot delete branch 'MBranch' checked out at 'C:/Users/porta/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub'

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (MBranch)
$ git checkout NBranch
Switched to branch 'NBranch'

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (NBranch)
$ git branch -D MBranch
Deleted branch MBranch (was 0ad5586).

porta@DESKTOP-NR6JE9F MINGW64 ~/Documents/manuel/DAW/ILERNA/gitHub/Ej_GitHub (NBranch)
$
```

To create branches, we use git branch ⟨branch Name⟩. To switch branches we use git checkout ⟨branch Name⟩. To list the branches and see which one we're on git branch and if we want to rename the branch git branch -m ⟨newName⟩.

To remove branches we use git branch -D ⟨branch Name⟩. YOU CANNOT DELETE THE BRANCH YOU ARE ON.

E3

To work as a couple we must share the repository.
Next, we have to create a new branch where we will add
our files. Each will do the basic iteration: git add -A,
git commit -m "" and git push. However, the person doing the
push later, you'll get a rice that you'll need to fix using git pull to add the
changes your partner made.

To incorporate two different branches we use git merge. It must be used from the main
branch to merge both.

If we don't do git pull, we will get an error. This error happens when we haven't
incorporated our partner's changes and we try to do a git push.

E3

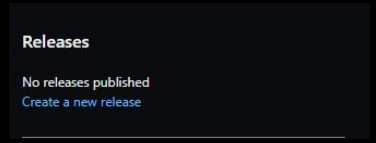
```
DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (hp042)
$ git add -A

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (hp042)
$ git commit -m "First commit."
[hp042 5a3a112] First commit.
 Committer: DAW <DAW@ILERNA.SEVILLA>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 9 insertions(+)
 create mode 100644 hp04.html

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (hp042)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (main)
$ git merge hp042
Updating f60ff92..5a3a112
Fast-forward
 hp04.html | 9 +++++++++
 1 file changed, 9 insertions(+)
 create mode 100644 hp04.html

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (main)
$ git add -A

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (main)
$ git commit -m "First commit"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 552 bytes | 552.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/marioariza/UD2-DWEC-E2.git
   f60ff92..5a3a112  main -> main

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/UD2-DWEC-E2 (main)
$
```

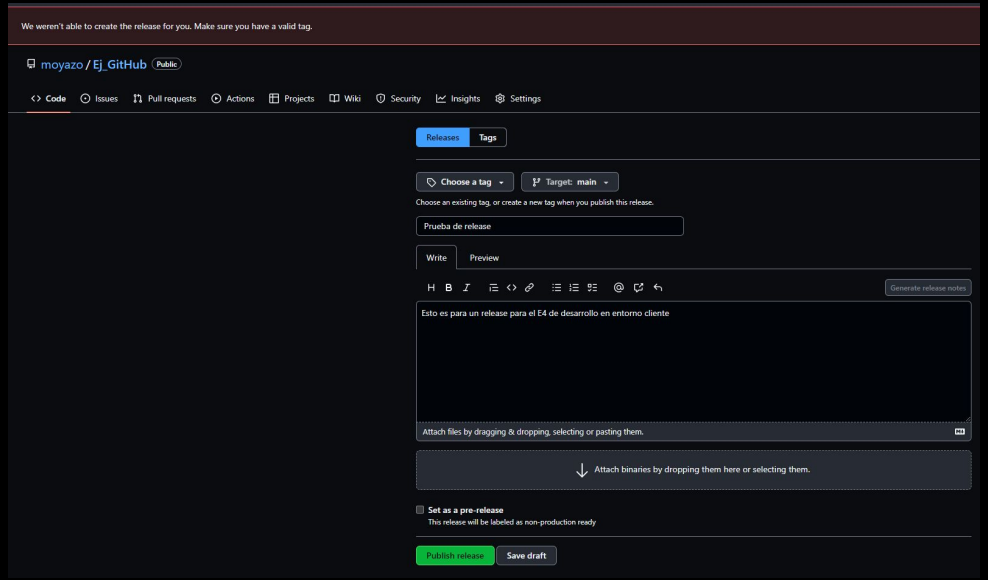
E4

```
AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; nano hp05.html

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; nano hp06.html

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
;

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; nano hp07.html

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; ls
.ICENSE  README.md  hp01.html  hp02.html  hp03.html  hp05.html  hp06.html  hp07.html

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; git add -A
/arning: in the working copy of 'hp05.html', LF will be replaced by CRLF the next time Git touches it
/arning: in the working copy of 'hp06.html', LF will be replaced by CRLF the next time Git touches it
/arning: in the working copy of 'hp07.html', LF will be replaced by CRLF the next time Git touches it

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; git commit -m "hp05- hp07 creados"
,main 5407bf5] hp05- hp07 creados
 Committer: DAW <DAW@ILERNA.SEVILLA>
'our name and email address were configured automatically based
n your username and hostname. Please check that they are accurate.
'ou can suppress this message by setting them explicitly. Run the
'ollowing command and follow the instructions in your editor to edit
'our configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 3 files changed, 4 insertions(+)
 create mode 100644 hp05.html
 create mode 100644 hp06.html
 create mode 100644 hp07.html

AW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
; git push
:numerating objects: 6, done.
:ounting objects: 100% (6/6), done.
Delta compression using up to 6 threads
:ompressing objects: 100% (2/2), done.
/riting objects: 100% (5/5), 392 bytes | 392.00 KiB/s, done.
'otal 5 (delta 1), reused 0 (delta 0), pack-reused 0
'emote: Resolving deltas: 100% (1/1), completed with 1 local object.
'o https://github.com/moyazo/Ej_GitHub.git
   0ad5586..5407bf5  main -> main
```

Para añadir nuestros archivos a github debemos usar: git add -A, git commit -m “” y git push.

E4

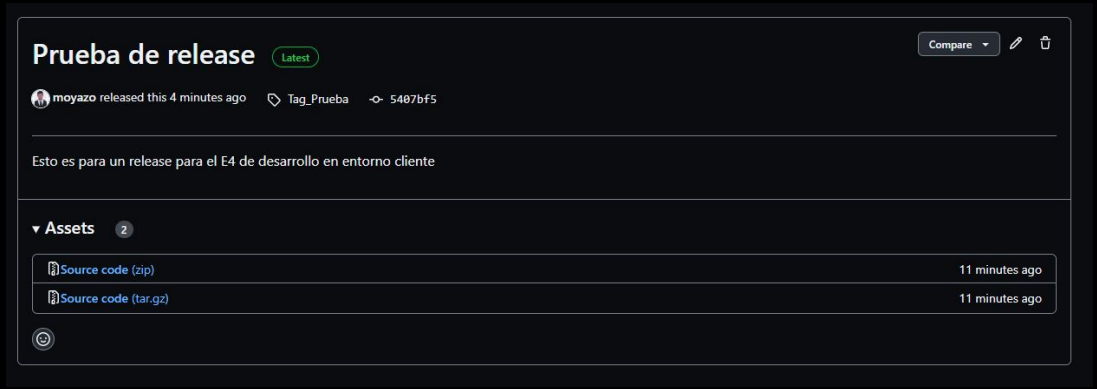




To create a release we must go to our repository. However, we must assign a tag to it so we will have to create one. Basically, a release saves us the version or state of a program that we want to save so as not to lose it.

We will get a message warning us that we must create a tag for the release.

E4



We assign a tag and we will have the zip available with the version of the program

E4

Diferencias Release - Tag	
Una release se crea a partir de un tag existente y expone las notas de la versión y los enlaces para descargar el software o el código fuente de GitHub.	Un tag es un puntero a un commit específico. Este puntero puede estar sobrecargado con alguna información adicional (identidad del creador de la etiqueta, una descripción, una firma GPG, ...).

E4

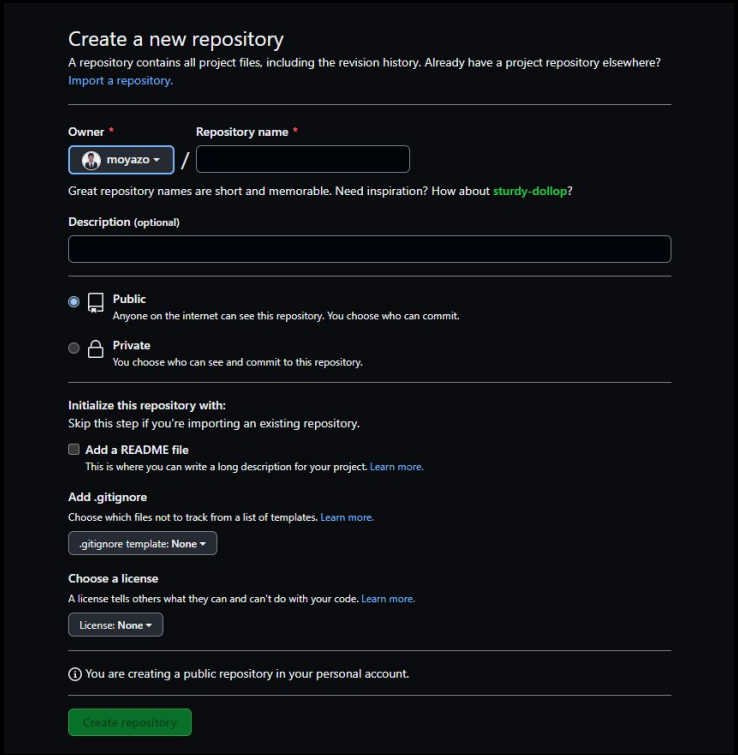
An Issue is a note in a repository that tries to draw attention to a problem. It can be a bug to fix, a request to add a new option or feature, a question to clarify something that is not properly clarified, or many other different things. On GitHub you can tag, search, or assign Issues, making managing an active project easier.

E5

```
DAW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
$ nano stash.txt

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        stash.txt

nothing added to commit but untracked files present (use "git add" to track)

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
$ git add -A
warning: in the working copy of 'stash.txt', LF will be replaced by CRLF the next time Git touches it

DAW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   stash.txt


DAW@PC-S1287 MINGW64 ~/Documents/Manuel/Ej_GitHub (main)
$ git stash
Saved working directory and index state WIP on main: 5407bf5 hp05- hp07 creados
```

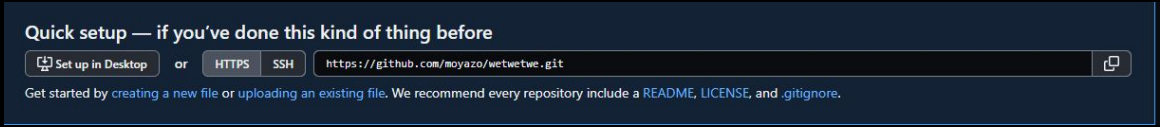
To use git stash, we must create a new file or modify another. When we make changes to it we use git bash to save those changes temporarily, so git status won't detect the changes:

E6



Markdown is a simple and easy-to-write type of plain text code documentation. It can also be used to document and point out important points to keep in mind when reading program code.

E6



If we already have our repository, but we have forgotten to add the Read.me, it will give us the recommendation to create a Read.me

END