

**Documentación: Gestión de Proyectos con Git y GitHub**

CFGS DUAL DESARROLLO DE APLICAIONES WEB - ENTORNOS DE DESARROLLO

Daniel Arqués Toro y Daniel Jonathan Zurita Mena

֎ Índice General

[ **Índice de Ilustraciones** 2](#_Toc24998599)

[**1.** **Introducción** 3](#_Toc24998600)

[**2.** **Entornos de Desarrollo** 4](#_Toc24998601)

[**2.1.** **Registro en Git Hub** 5](#_Toc24998602)

[**2.2.** **Crear un repositorio en GitHub** 6](#_Toc24998603)

[**3.** **Programación** 7](#_Toc24998604)

[**3.1.** **TDD** 7](#_Toc24998605)

[**3.2.** **Sistema SOLID** 7](#_Toc24998606)

[**3.3.** **Barricadas** 7](#_Toc24998607)

[**3.4.** **Parte lógica** 7](#_Toc24998608)

[**4.** **Lenguaje de Marcas** 8](#_Toc24998609)

[**4.1.** **XML** 8](#_Toc24998610)

[**4.2.** **XSD** 8](#_Toc24998611)

[**5.** **Base de Datos** 9](#_Toc24998612)

[**6.** **Webgrafía** 10](#_Toc24998613)

# **Índice de Ilustraciones**

[Ilustración 1 - Introducción: Git Hub 3](#_Toc24998614)

[Ilustración 2 - Introducción: Repositorios y branch 3](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998615)

[Ilustración 3 - Introducción: Presentación Word 3](#_Toc24998616)

[Ilustración 4 - Ejercicio 1: $Git log --oneline --all --graph 4](#_Toc24998617)

[Ilustración 5 - Ejercicio 1: Recetas 4](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998618)

[Ilustración 6 - Registro en Git Hub: Sing up 5](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998619)

[Ilustración 7 - Registro en Git Hub: Menú principal 5](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998620)

[Ilustración 8 - Crear un repositorio en Git Hub - New Repository 6](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998621)

[Ilustración 9 - Crear un repositorio en Git Hub - New 6](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998622)

[Ilustración 10 - Crear un repositorio en Git Hub - Comandos Git Bash 6](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998623)

[Ilustración 11 - Crear un repositorio en Git Hub - Parámetros 6](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998624)

[Ilustración 12- TDD 7](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998625)

[Ilustración 13 - SOLID 7](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998626)

[Ilustración 14 - Barricadas 7](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998627)

[Ilustración 15 - Lógica 7](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998628)

[Ilustración 16 - XML validator 8](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998629)

[Ilustración 17 - XML Schema Generator 8](file:///C:\Users\dajoz\Documents\Entornos%20-%20Receta\recetas\Documentación_ProyectoVLC.docx#_Toc24998630)

# **Introducción**

El objetivo de este proyecto consiste en realizar un programa que permita reproducir canciones aleatorias con el programa *VLC.* Es un proyecto transversal de todos los módulos de 1º DAW Dual. Se realizará un control de versiones (basado en *Commitizen)* mediante línea de comandos *Git Bash* y en entorno gráfico *Git Hub*.

Ilustración 1 - Introducción: Git Hub

Se realiza en parejas, por lo tanto, ambos estaremos trabajando en los mismos archivos, y se encargará de ir documentando las partes del proyecto que realice.

La manera de la que trabajaremos simultáneamente en los mismos archivos será mediante *Git Hub*. El repositorio constará de dos *branches*, la principal, llamada *master* donde se encuentran los archivos ya finalizados, y otra llamada *preprod* donde se hacen los *commit* a medida que se va avanzando en el proyecto. Una vez acabadas todas las partes del proyecto, se hace un *merge* de las dos *branch* para que quede el proyecto listo en la rama master.

La documentación del mismo tiene que estar presente con los puntos que hay en el índice de este documento.

Ilustración 2 - Introducción: Repositorios y branch

No es necesario, pero el documento tiene que estar bien presentado, con una tabla de contenido, uso de estilos, uso de cabeceras y pies de página, el texto justificado y otras opciones que dan los editores de texto.

Ilustración 3 - Introducción: Presentación Word

La fecha de entrega del trabajo será día 2 de Diciembre.

El final del documento consta de una Webgrafía hypervinculada a las direcciones que se han tomado como referencia y/ o consulta para este trabajo.

Algunos aspectos de la presentación no se han seguido al pie de la letra con la intención de obtener un trabajo más original y único. Con la esperanza de que no repercuta a la nota del trabajo.

# **Entornos de Desarrollo**

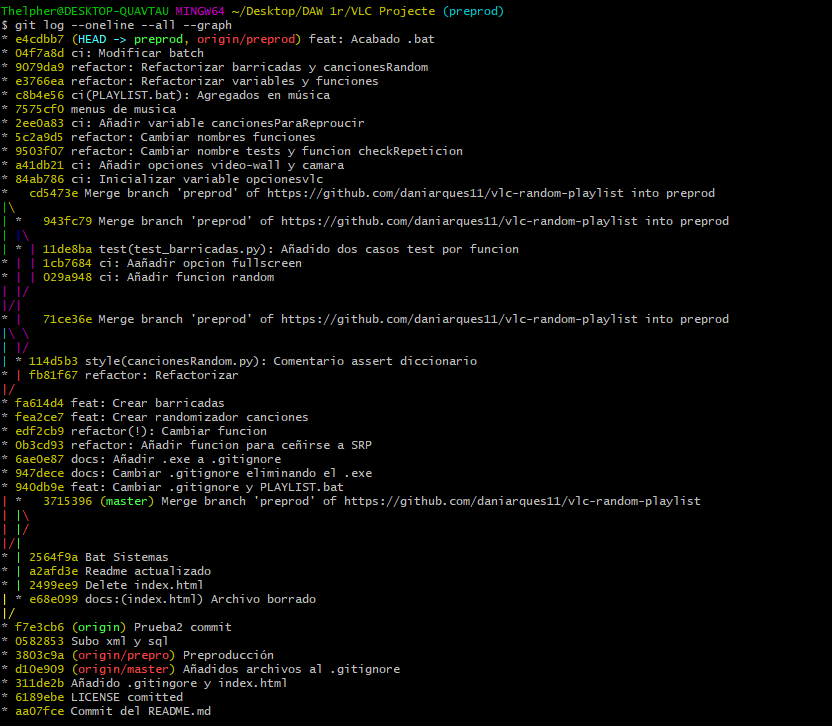
En este proyecto ponemos en práctica lo anteriormente aprendido sobre control de versiones, compartir repositorios, crear *branches* y hacer un *merge* entre las mismas; añadiremos algunas capturas de líneas de comandos en *Git Bash* para que quede constancia de que se han realizado en modo texto. 

Ilustración 4 - $Git log --oneline --all --graph

Vamos a simular que no hayamos creado una cuenta ni un repositorio para realizar una guía rápida de cómo hacerlo.

## **Registro en Git Hub**

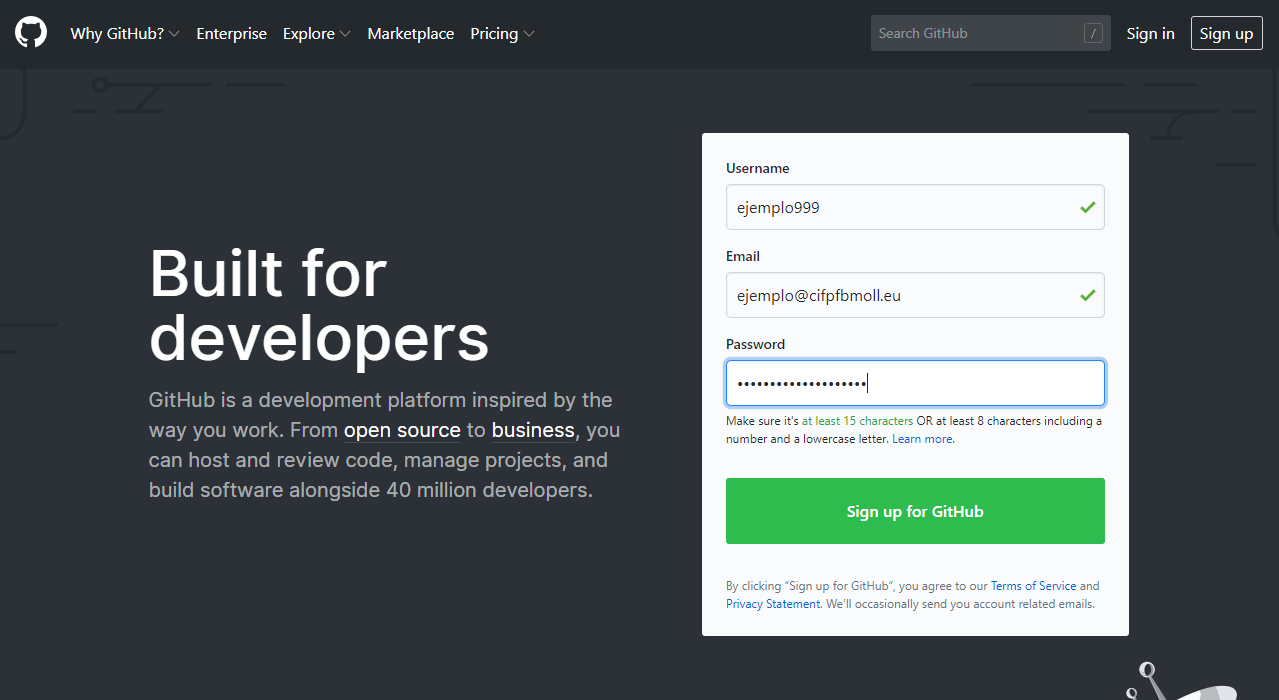
Empezamos registrando nuestro correo electrónico en la página web de Git Hub: [github.com](http://www.github.com)

Ilustración 6 - Registro en Git Hub: Sing up

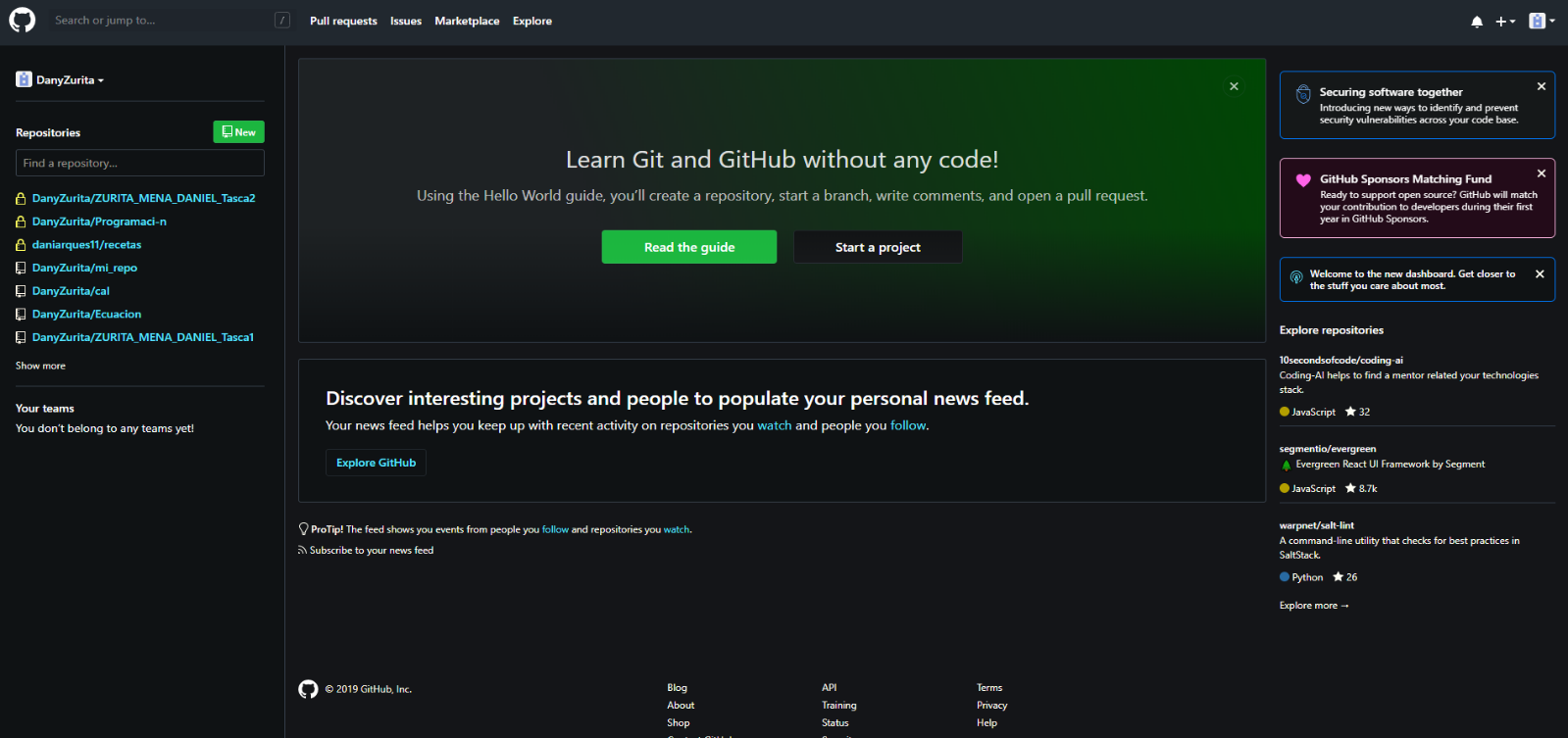
En nuestro caso, utilizamos el correo del coordinador del proyecto, ya que evitaremos usar el dominio del centro para que los profesores que pertenecen al dominio, no reciban correos cada vez que se modifica cualquier repositorio de todos los alumnos que tienen.

Ilustración 7 - Registro en Git Hub: Menú principal

Una vez acabado el registro y verificado el correo accederemos al menú principal de *Git Hub* en el que podemos observar en la parte superior un navegador entre las funciones más utilizadas y a la derecha el panel de repositorios.

## **Crear un repositorio en GitHub**

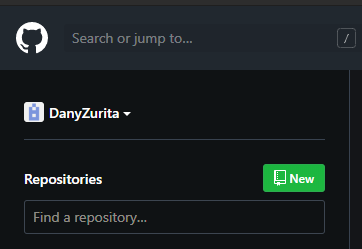
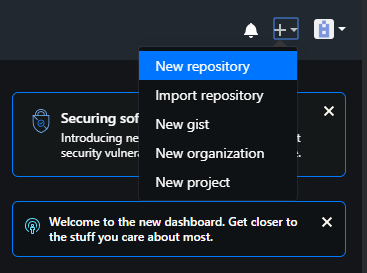
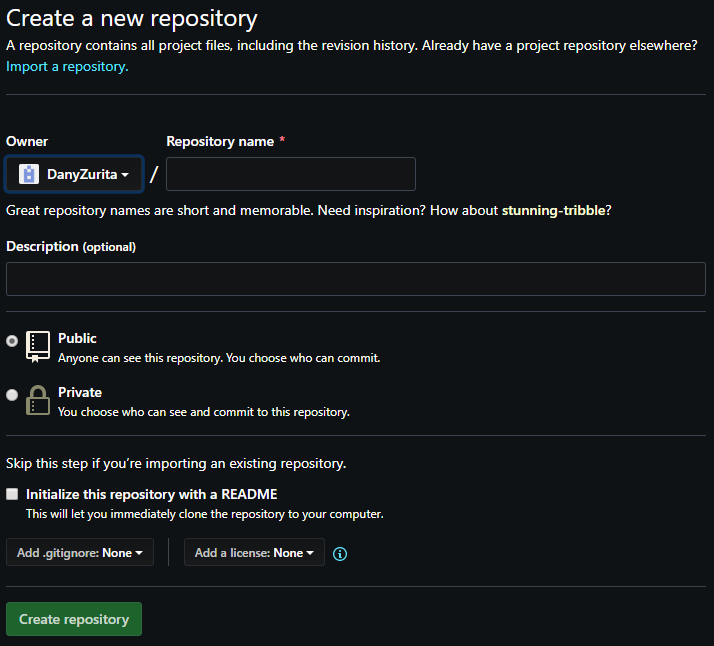
Para crear un repositorio en Git Hub, pendremos dos opciones:

Ilustración 8 - Crear un repositorio en Git Hub - New Repository

Ilustración 9 - Crear un repositorio en Git Hub - New

* Menú principal izquierdo –> Botón verde -> New
* Navegador superior derecho -> + -> New Repository

Una vez hecho esto nos aparecerá un pequeño formulario en el que se introduce el nombre del repositorio, una pequeña descripción opcional, si el público o privado y otras opciones.

Una vez seleccionados estos parámetro nos aparecerá una pequeña guia sobre los comandos básicos de *Git Bash* para crear un repositorio en local desde cero, importar un repositorio de la máquina local a este nuevo o imporar código de otro repositorio existente al que acabamos de crear.



Ilustración 10 - Crear un repositorio en Git Hub - Comandos Git Bash

Ilustración 11 - Crear un repositorio en Git Hub - Parámetros

## **Gestión del proyecto**

## Una vez creado el repositorio, ya tenemos todo listo para empezar a trabajar. De manera que escogemos trabajar en un prototipo

## **Clockify**

# **Programación**

La programación de este proyecto es l, ya que con lo que hemos aprendido durante el curso, debemos programar el acceso a la base de datos, la parte lógica y

## Resultado de imagen de tdd png**TDD**

Ilustración 12- TDD

Para realizar los programas nos basamos en usar ***Test Drive Developement*** que consiste en utilizar casos test, hacer *debugging* al código y refactorización.

## **Sistema SOLID**

En el trabajo nos basamos principalmente en estos dos puntos:

***Single Responsability Principle* (SRP)**, consiste en fraccionar el código en funciones aisladas para que cada método se encargue de una única labor.

***Open/Closed Principle*** **(OCP)**, se define como el código abierto a la extensión, cerrado a la modificación.

## Resultado de imagen de SOLID programming png**Barricadas**

Ilustración - SOLID

Utilizamos código para comprobar que los inputs del programa son válidos y casos test para testear su funcionalidad.

## Resultado de imagen de barricada png**Parte lógica**

Ilustración - Barricadas

Consiste en crear una lista aleatoria de canciones con un diccionario que contiene información de las canciones como input. No se permite usar el orden aleatorio de los diccionarios. Hay que usar un método que utilice el procedimiento *random* que genere un número aleatorio.

Seguimos en proceso de transformar ese output de ID en una lista de rutas para que posteriormente se invoque al programa VLC desde Python



Ilustración - Lógica

# **Lenguaje de Marcas**

Ilustración - XML validator

La introducción a los lenguajes de marcas consistía en explicarnos que el lenguaje XML sirve para crear lenguajes de marcas a partir del mismo y que tiene un esquema que lo soporta.

## Resultado de imagen de xml png**XML**

Realizamos un primer XML a mano que contiene una librería de canciones junto con la información de estos.

Una vez hecho esto, comprobamos el XML con una herramienta web.

## Resultado de imagen de xsd png**XSD**

Creamos un esquema de definición mediante una herramienta web. Nos daba el XSD ya creado y tuvimos que añadir comentarios de porqué se utilizaban algunos atributos y el porqué de sus valores.



Ilustración - XML Schema Generator

# **Base de Datos**

La asignatura de BBDD no aporta mucho en este proyecto. Aún así, aprendimos como exportar una base de datos en MySQL a un documento XML con comandos.

Ilustración 18 - Exportar BBDD a XML

# **Sistemas Informáticos**

Mediante comandos MS-DOS creamos las carpetas correspondientes del repositorio local, la librería de música a la que llamaremos desde el programa para acceder a las canciones e instalamos el mismo programa VLC.

Llegamos a la conclusión de que el programa VLC, lanza un error si la sintaxis del comando para ejecutarlo es incorrecto. Tampoco funciona si tratamos de lanzar cualquier comando vlc si no accedemos primero al directorio en el cual se encuentra mediante una ruta absoluta (otra opción es añadir esa ruta al *path*).

La sintaxis correcta para ejecutar el programa VLC remotamente mediante línea de comandos es la siguiente:

>>> <ruta comando *vlc*> <archivos a ejecutar> [opciones]

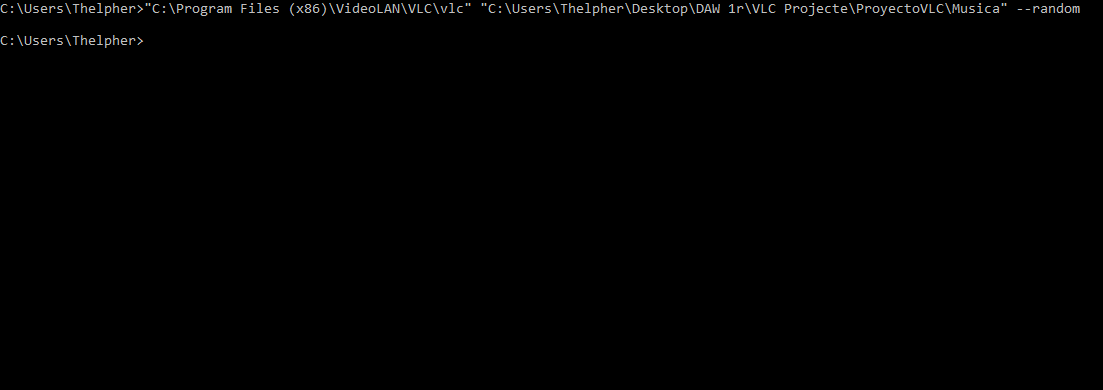


Ilustración 19 – Ejemplo comando vlc

# **Webgrafía**

* Pull Request
  + <https://help.github.com/en/articles/about-pull-requests>
* Como volver a un commit anterior
  + <https://es.stackoverflow.com/questions/41489/volver-a-commit-anterior>
* Slides Git Classroom
  + <https://classroom.google.com/u/1/c/NDMyNjA1NjcwNDda/m/NDMzMjIwNzg0Nzda/details>
* Repositorio compartido
  + <https://classroom.google.com/u/1/c/NDMyNjA1NjcwNDda/m/NDMzMjIwNzg0Nzda/details>
* XML Validator
  + <https://www.xmlvalidation.com/index.php?id=1&L=0>
* XML Schema Generator
  + <https://xml.mherman.org/>
* Exportar BBDD a XML
  + <http://www.tic2.org/WebTecnica/Programas/Aplicaciones/MySQL/exportar_XML.html>
* Comandos de VLC
  + <https://wiki.videolan.org/Documentation:Command_line/>