**2.4 HERRAMIENTAS, ENTORNO E INFRAESTRUCTURA**

Se explicará las herramientas, el entorno y la infraestructura que se usará para llevar a cabo la SCM.

**2.4.1  HERRAMIENTAS**

**2.4.1.1  GITHUB**

Github es una plataforma creada para facilitar el desarrollo colaborativo de software que nos permite alojar proyectos como repositorios en la web gratuitamente, por lo general de forma pública, aunque podemos alojar los proyectos de modo privado, si pagamos una pequeña suscripción mensual.

**CARACTERÍSTICAS**

* Guardar en determinado momento los cambios efectuados a un archivo o conjunto de archivos, con la oportunidad que tener acceso a ese historial de cambios, ya sea para regresar a una de esas versiones o para hacer comparaciones entre ellas.
* Si luego de copiar un proyecto (hacer Fork) haces ajustes que arreglan bugs o introducen una nueva funcionalidad, puedes proponerle al dueño del proyecto que integre tus cambios en su código a través de un Pull Request.
* Permite trabajar conjuntamente en una idea con un amigo o colega, además se puede invitar a otros usuarios como colaboradores, permitiendo así que estos lean y escriban directamente sobre el repositorio.
* Posee un estupendo visor de código mediante el cual, a través del navegador, podremos consultar en cualquier instante el contenido de un archivo determinado.
* Es una plataforma web, por tanto, es independiente del sistema operativo que utilices, y además Git que es la herramienta que si requiere instalación es compatible con todos los sistemas; Linux, OSX y Windows.
* Cuenta con una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero.

**DESVENTAJAS**

* Si queremos tener proyectos de modo privado debemos tener una suscripción mensual.

**2.4.1.2  GIT**

Es un sistema de control de versiones distribuído (scvd) escrito en C, el cual permite la creación de una historia para una colección de archivos el incluye la funcionalidad para revertir la colección de archivos a otro estado.

**CARACTERÍSTICAS**

* El desarrollo de la aplicación será únicamente nuestro, pudiendo decidir qué parte de nuestro proyecto compartimos y con quién.
* El control de versiones se puede realizar dentro de la propia red con una mayor velocidad de acceso y escritura, así como eliminando el requisito de contar con una conexión a internet obligatoria.
* Se puede crear diferentes ramas sobre las que aplicar nuestras modificaciones en entornos aislados de la línea principal de desarrollo.
* Si en la creación de una rama del proyecto encontramos que uno de los cambios incluidos se integra sin presentar conflictos con las diferentes partes de nuestra aplicación, podremos hacer converger dicha ramificación con el desarrollo principal de forma sencilla y segura.
* Las ramificaciones nos preparan un entorno aislado de pruebas sobre el desarrollo de la línea central de nuestra aplicación.
* Se hace uso de sistemas de árbol SHA1, lo que asegura que hasta que no se realice la comprobación del cifrado o firma, los cambios no se escribirán en el servidor.
* Es un sistema de control de versiones libre de código abierto.

**DESVENTAJAS**

* No permite indexar directorios.
* Teniendo una gran curva de aprendizaje, con solo 152 comandos por aprender, la documentación de muchos de estos comandos es obsoleta.

**2.4.2   ENTORNO**

El ambiente de trabajo estará compuesto por:

* **Desarrolladores:** Las personas involucradas en el proyecto, tienen acceso para poder modificar los documentos del repositorio.
* **Administrador:** Se encarga de verificar los cambios de los documentos, y revisar que se trabaja en el branch establecido. Así mismo debe dar los permisos necesarios a los desarrolladores para realizar el desarrollo.
* **Repositorio:** Github.

**2.4.3  INFRAESTRUCTURA**

Se manejan 2 tipos de ramas o branch, que nos va servir para controlar mejor los commit, se va desarrollar en la rama de desarrollo (branch development) que corresponderá 1 branch para cada miembro del equipo y la rama maestra (branch master) la cual aloja las versiones cuyos cambios hayan sido aprobados y estén listos para su respectivo release.

**Branch master**: Esta rama será la principal, donde se pondrá los cambios aprobados por el administrador.

**Branch Development**: Esta rama será para los desarrolladores, donde podrán hacer sus cambios previa aprobación, sin alterar la rama principal.