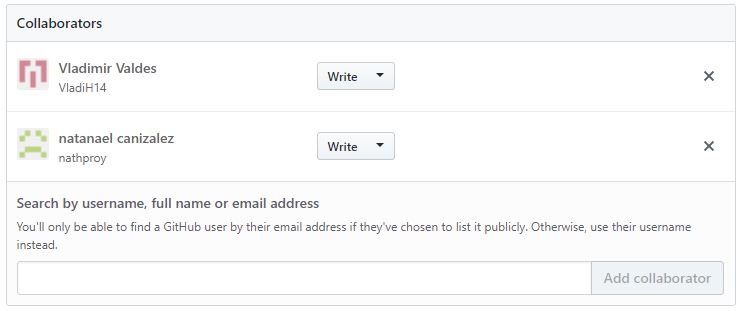
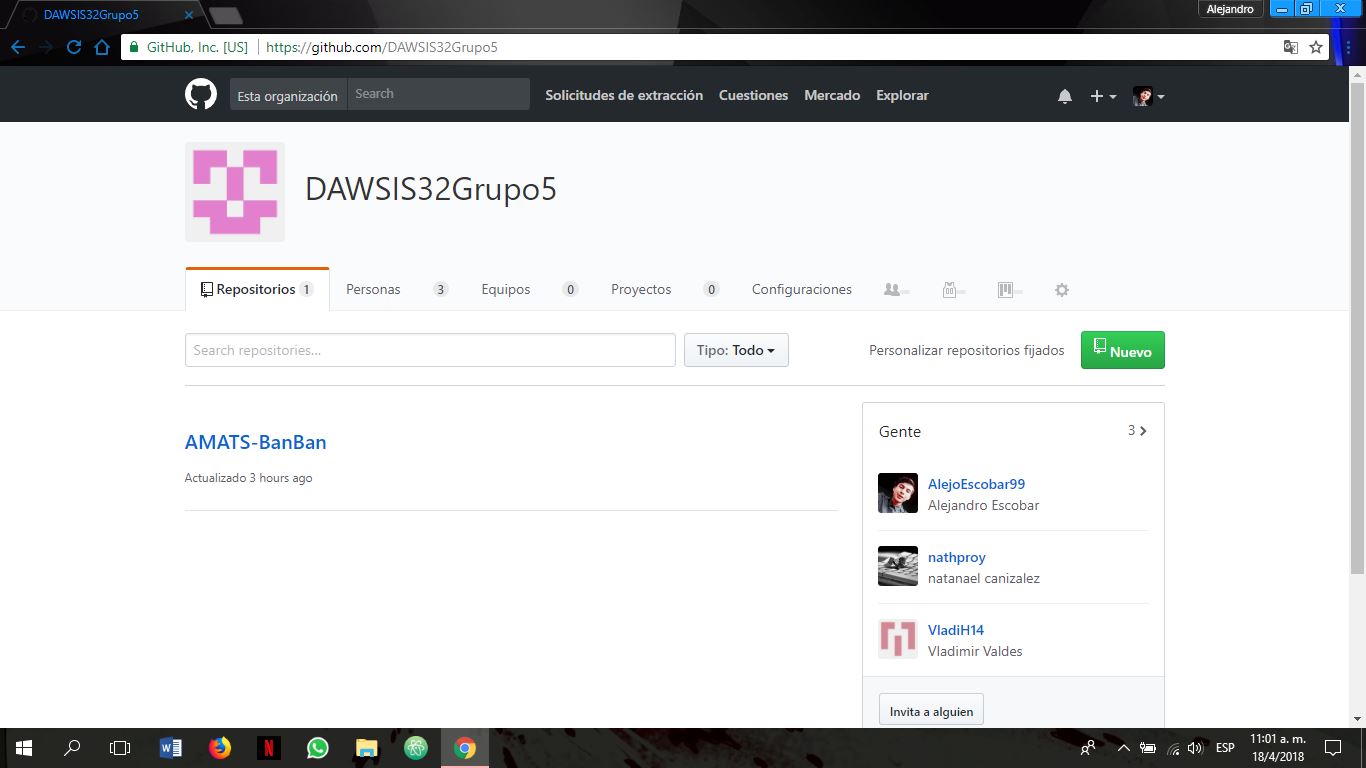
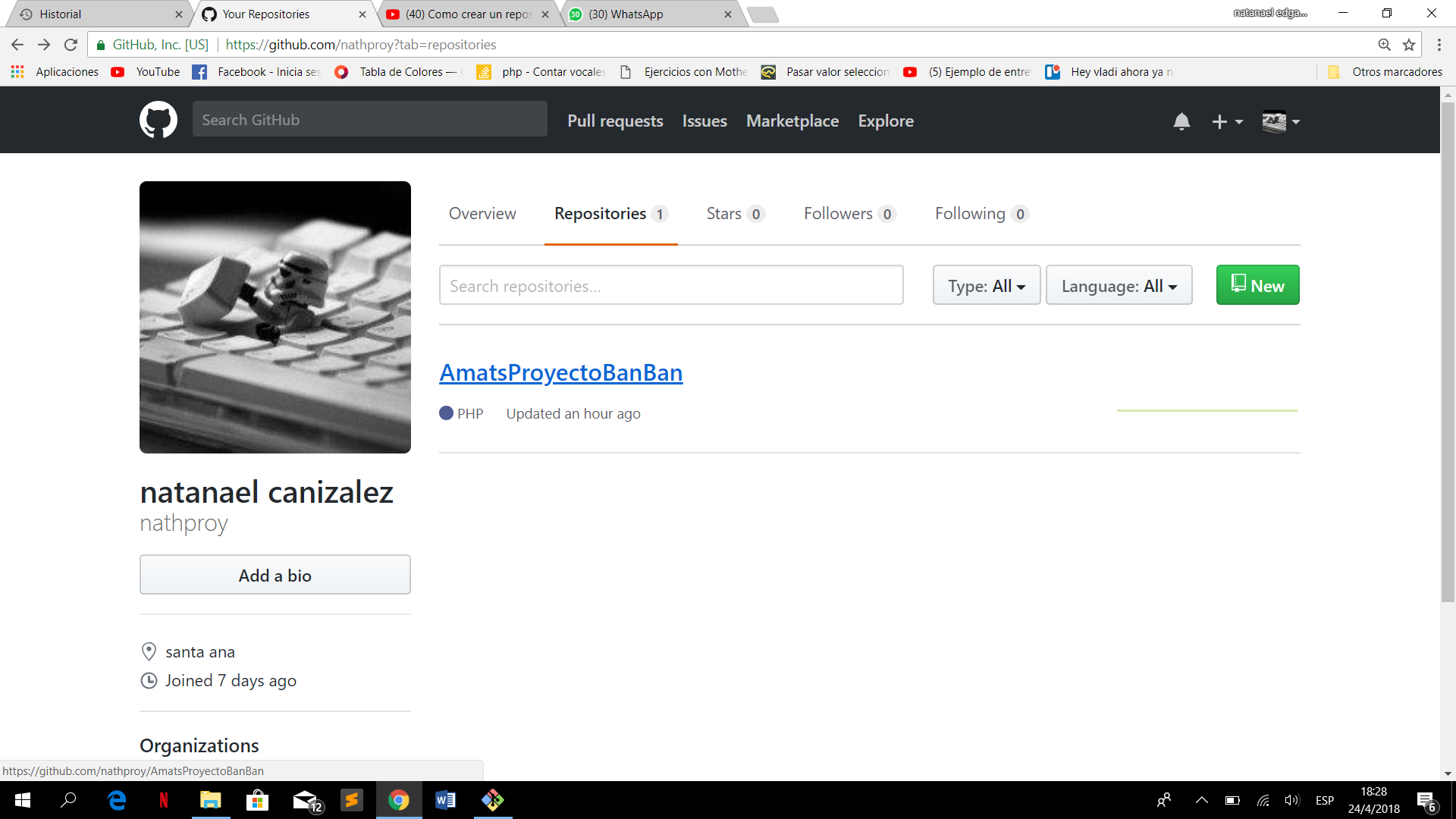
|  |  |  |
| --- | --- | --- |
| **Nombre** | **Traducción** | **Descripción** |
| Fetch | Ir a buscar | Trae los cambios, pero los deja en otro branch |
| Merge | Ir | Es la rama a donde se enviara |
| Pull-request |  | es una petición que el propietario de un fork de un repositorio hace al propietario |
| Issue | problema | un issue es la unidad de trabajo designada para realizar una mejora en un Sistema informático |
| Fork | copia | Hace una copia exacta del repositorio |
| workflow | Flujo de trabajo | Es un flujo de trabajo en desarrollo colaborativo |
| Git client | Cliente git | Software para el control de verciones |
| Branch | Rama | Son atajos para crear ramas |
| Reléase | Lanzamiento | Para preparar la aplicación para su puesta en marcha |
| Tag | Etiqueta | Son etiquetas para marcar puntos especificos |
| Pulse | Pulso | contiene información acerca de otros archivos en un directorio. |
| Readme | Léeme | contiene información acerca de otros archivos en un directorio. |
| Clone | Clonar | Sirve para clonar un archivo |



Imagen que contiene captura de pantalla, interior, ordenador, portátil

Descripción generada con confianza muy alta

**1 ¿Defina las ventajas y desventajas de Git Hub?**

**Ventajas:**

Muy versátil y con figurable.

Distribuido, podemos hace commit en nuestra maquina local, por lo que podemos seguir trabajando sin conexión a Internet subir los cambios más tarde.

Rápido, la mayoría de operaciones son locales, resolución de conflictos a través de herramientas mergetool y el editor meld.

Seguro, se puede trabajar incluso si el servidor este caído sin perder tiempo de trabajo.

La posibilidad de ver rápidamente la historia del proyecto con el comando git log.

**Desventajas:**

Curva de aprendizaje para los que vienen de SVN.

Muchos comandos y el significado de los comandos.

Disponibilidad, el código está en un solo repositorio.

Complejidad, recomendado para proyectos grandes**.**

**2 ¿Es necesario trabajar bajos estos entornos de desarrollo?**

Si por que sirven para:

Controlar el avance del código fuente de diferentes miembros del grupo facilitando el trabajo colaborativo.

Establece versiones en el programa a medida que avance el desarrollo.

Almacenar un historial, en caso de surgir un error y no se pueda solucionar volver a un punto anterior del programa y continuar desarrollando sin tener que volver hacer todo de el programa nuevo

**3 ¿Qué es un commit?**

Commit (locales): Cada commit es un grupo de cambios sobre uno o varios ficheros, con una descripción, una fecha, un autor, etc. La gran diferencia con SVN es que los commit en Git son locales hasta que no se efectúa la subida al servidor. Estos commit locales (importante que sean locales) pueden ser modificados sin peligro (con modificados quiero decir que se les pueden añadir más cambios, actualizar su mensaje o incluso eliminarlos).

Commit (remotos): Cuando se suben cambios al servidor (o como se le llama en Git: el remoto), se considera que estos entran a formar parte del histórico compartido entre los desarrolladores del proyecto y, por lo tanto, no es buena práctica modificarlos del mismo modo en que se hace cuando los commit son locales (además hacerlo puede provocar importantes quebraderos de cabeza).

**4 ¿Qué es un repositorio?**

Almacén o lugar donde se guardan ciertas cosas.

**5 ¿Cuál es la ventaja de trabajar con repositorios privados?**

**Versionar tu código.**

Esto quiere decir 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. En este punto hay que destacar las ventajas frente a herramientas como Subversion y Perforce; que son usadas para este mismo fin, la principal es que no necesitarías montar, mantener, hacer copias de seguridad, etcétera a un repositorio o servidor para proteger el código que escribes, en este caso Github es el plataforma que cumple esta función. Lo único que necesitas es instalar Git en tu equipo, crear tu cuenta en Github y luego de realizar algunas configuraciones sencillas y ejecutar los comandos adecuadas podrás enviar tu código a Github. Es importante mencionar que la visibilidad de los proyectos será público o privado según el tipo de cuenta que adquieras.

**Aprender y experimentar**

Como muchos desarrolladores tienen de forma pública sus proyectos en Github es posible acceder a su código, leerlo, estudiarlo y aprender de el, e incluso podrías hacer cambios y experimentar sin afectar el código original, a esto ultimo se le como como: hacer [fork] .

**Contribuir**

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. Eso lo puedes hacer enviando un [pull request](https://help.github.com/articles/using-pull-requests/) con todas tus modificaciones o novedades. Mediante un pull request el administrador del repositorio original tendrá la posibilidad de revisar, testear o escribir comentarios sobre alguna líneas de tu propuesta, y finalmente si considera interesante tu contribución podrá realizar la fusión de esta con el código original (a esto se le conoce como hacer [merge](https://help.github.com/articles/merging-a-pull-request/) o en caso contrario rechazar los cambios propuestos.

**Trabajo en equipo**

Github es el lugar perfecto para trabajar conjuntamente en una idea con un amigo o colega, después de crear tu proyecto podrás invitar a otros usuarios como colaboradores, permitiendo así que estos lean y escriban directamente sobré tu repositorio.

**Estar informado(a)**

Github tiene un sistema de notificaciones con el cual puedes estar al tanto de las actividades alrededor de un repositorio en el cual estas participando o uno en el cual estés interesado. La notificaciones de los proyectos a los que perteneces o eres dueño, están habilitadas de forma automática, y para aquellos en los cuales no eres participantes pero estas interesado es necesario activarlas haciendo

**Visor de código**

GitHub posee un estupendo visor de código mediante el cual, a través del navegador, podremos consultar en cualquier instante el contenido de archivo determinado, con la sintaxis correspondiente a el lenguaje en el que esté escrito. Este navegador es realmente rápido, y gracias a él podremos hacer pequeñas consultas o copiar porciones de código sin necesidad de bajarse todo el repositorio.

**Mostrar tus conocimientos**

Con Github puedes mostrar tus habilidades como desarrollador(a), puesto que es el código escrito en los archivos, donde reposa el resultado del proceso del desarrollo de software. Al compartir tu cuenta de Github con tu potencial empleador o cliente, este podrá ver la calidad del código que escribes a través de los proyectos públicos que estén en tu cuenta. Todos los proyectos que se escriben para ejecutar una idea, aprender un nuevo lenguaje o tecnología son válidos al momento de exhibir tus conocimientos, así que no dudes en publicarlo en tu cuenta. Como complemento a lo anterior, con [Github Pages](https://pages.github.com/)puedes crear incluso una página como [esta](http://equintana.github.io/) que te sirva como portafolio, en la cual puedes escribir sobre ti, los conocimientos que posees o poner enlaces de los proyectos en los que has participado.

**Registro de incidencias**

Cada proyecto creado en Github incluye un sistema de seguimiento de problemas, del estilo sistema de tickets, este permite a los miembros de tu equipo (o a cualquier usuario de GitHub si tu repositorio es público) abrir un ticket escribiendo en este los detalles un problema que tenga con tu software o una sugerencia sobre una función que le gustaría que fuera implementada.

**Compatibilidad**

Github es una plataforma web, por tanto es independiente del sistema operativo que utilices, y además Git que es la herramient que si requiere instalación es compatible con todos los sistemas; Linux, OSX y Windows.

**Precio**

Github, es completamente gratis e ilimitado para proyectos públicos, es decir que todos podrán ver el código que estos contienen (aunque tu siempre tendrás el control sobre quien subirá cambios), sin embargo, si deseas puedes tener proyectos privados adquiriendo uno de planes que ofrece, los cuales van desde 7 a 50 dólares mensuales, permitiendo crear 5 y 50 repositorios privados respectivamente.

**6 ¿Desventajas de trabajar con repositorios públicos?**

Que todos pueden ver el proyecto.

**7 ¿Que otras herramientas se pueden utilizar para desarrollar** **en entornos colaborativos?**

Git Hub

Bitbucket

Sourceforge

Bazaar

Monotone.