Invitar a colaboradores a un repositorio personal

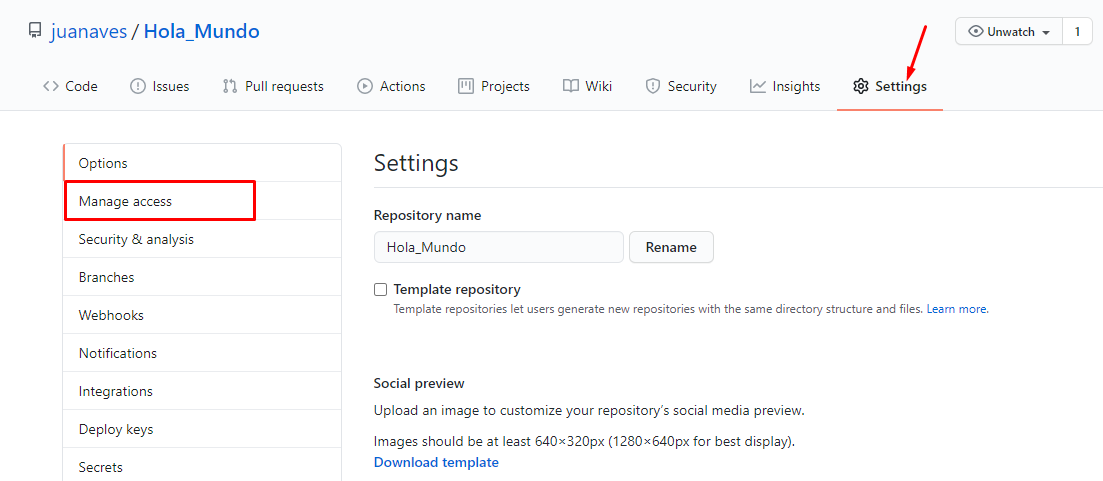
Puede **invitar a los usuarios** a convertirse en colaboradores de su repositorio personal.

Si está utilizando GitHub Free, puede agregar colaboradores ilimitados en repositorios públicos y hasta tres colaboradores en repositorios privados propiedad de su cuenta personal. Para agregar a más de tres personas como colaboradores del repositorio, actualice a GitHub Pro. Para obtener más información, consulte " [Productos de GitHub](https://help.github.com/articles/github-s-products) ".

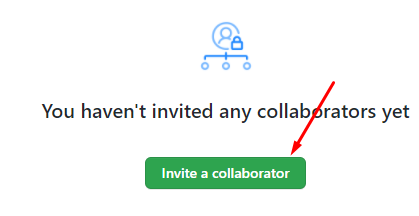
Los repositorios propiedad de una organización pueden otorgar un acceso más granular. Para obtener más información, consulte " [Permisos de acceso en GitHub](https://help.github.com/en/articles/access-permissions-on-github) ".

**Nota:** GitHub limita la cantidad de personas que pueden ser invitadas a un repositorio dentro de un período de 24 horas. Si excede este límite, espere 24 horas o cree una organización para colaborar con más personas.

1. Solicite el nombre de usuario de la persona que está invitando como colaborador. Si aún no tienen un nombre de usuario, pueden registrarse en GitHub. Para obtener más información, consulte " [Registrarse para obtener una nueva cuenta de GitHub](https://help.github.com/en/articles/signing-up-for-a-new-github-account) ".
2. En GitHub, navegue a la página principal del repositorio.
3. Debajo de su nombre de repositorio, haga clic en **Configuración**.

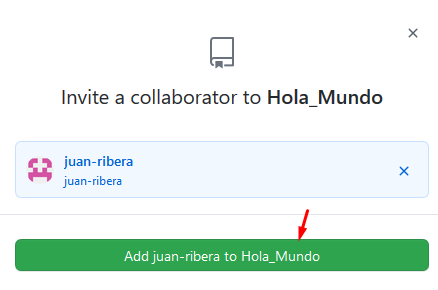


Luego tendremos que seleccionar la opción invitar a colaboradores



En donde introduciremos los colaboradores a los que queremos invitar

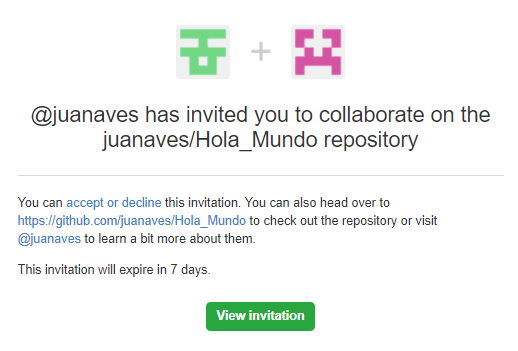
1. En "Colaboradores", comience a escribir el nombre de usuario del colaborador.
2. Seleccione el nombre de usuario del colaborador en el menú desplegable.



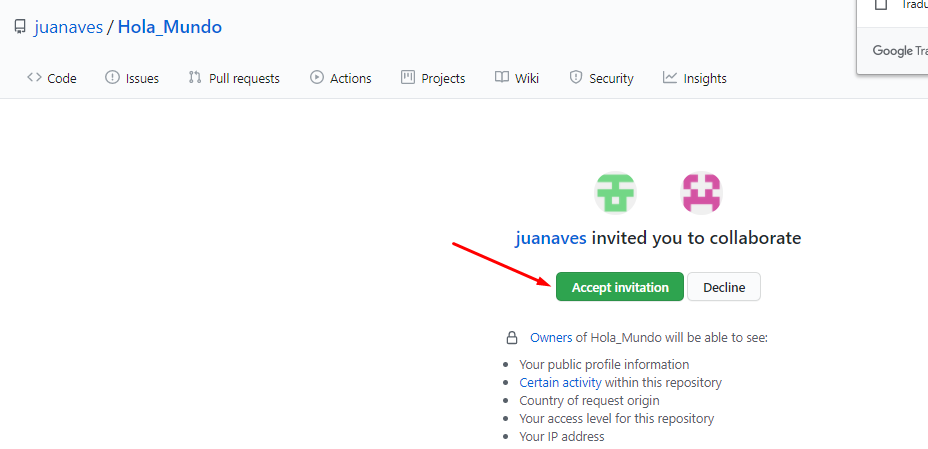
1. Haz clic en **Agregar collaborador** .
2. El usuario recibirá un correo electrónico invitándolo al repositorio. Una vez que acepten su invitación, tendrán acceso de colaborador a su repositorio.



Pulsando sobre el correo nos saldrá una pantalla parecida a esta

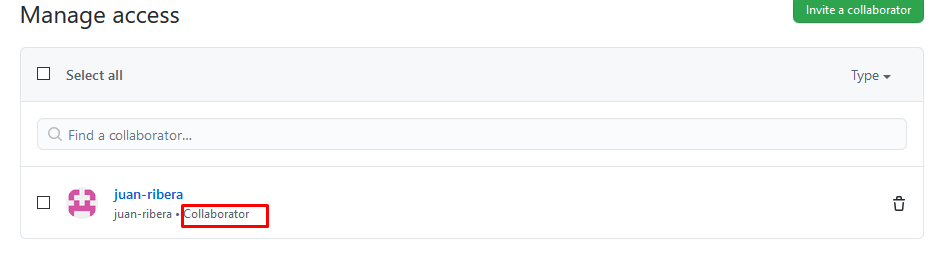


Pulsamos ver Invitación y nos llevará al repositorio de la persona que nos ha invitado



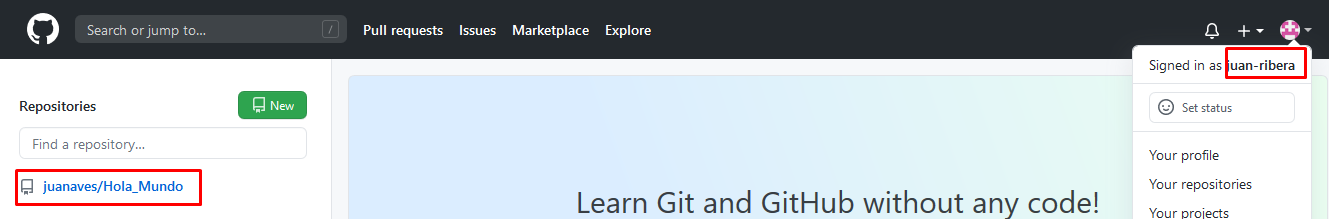
Y pulsamos aceptar invitación.

Ahora veremos que en la pantalla de invitados a nuestro repositorio ya la persona que hemos invitado NO está pendiente, sino que es ya colaborador



Y así con todas las personas a quien queramos invitar a nuestro repositorio.

Si nos vamos al repositorio de la persona que ha recibido la invitación le aparecerá un enlace al repositorio del que le ha invitado. Y podremos trabajar como si fuese nuestro proyecto. Nos lo podríamos clonar en nuestro ordenador y trabajar sobre el.



### Ejercicio 2: Desarrollo con Git empleando un modelo centralizado con GitHub

**Organizar una barbacoa con Git empleando un modelo centralizado con varias ramas de desarrollo.**

Objetivo

En este proyecto debe crear un repo y añadir algún(os) **colaborador**(es) con el que elaborará una barbacoa. **Uno de los miembros** del equipo hace de ***coordinador***.

* Organizar una barbacoa con Github
* Modelo: centralizado, **todos tienen permisos** para modificar el repositorio.
* Repo inicial con varios archivos:
  + **comida.txt**: comidas a llevar
  + **bebida.txt**: bebidas a llevar
  + **materiales.txt**: materiales necesarios (sillas, mesas, etc.)
  + **asistentes.txt**: lista de personas que van a ir. Quien pueda llevar coche que lo indique y que se apunten abajo sus acompañantes.
  + **juegos.txt**: propuestas de juegos para llevar a la barbacoa.

Datos del repositorio

En el fichero README.md se deberá poner: (Ver anexo final)

Cada colaborador trabaja en una rama con su nombre en la que trabaja y cuando termina crea un [**pull request**](https://help.github.com/articles/about-pull-requests/) pidiéndole al **coordinador de proyecto** que **mezcle** los cambios en la rama en la rama master.

El trabajo se realizará por filas, siendo un integrante de la cada fila el coordinador, el resto serán colaboradores. Deberéis meter al profesor como colaborador en cada uno de vuestros proyectos.

El nombre de los repositorios se dará de la siguiente forma:

**Barbacoa\_filaX**, donde la X es el número de la fila 1,2,3…,por ejemplo Barbacoa\_fila1

Flujo de trabajo

* Clonar repo remoto (clone).
* La rama **master**  contiene la versión definitiva, sin errores.
* Los cambios **locales** se realizan sobre la rama de **desarrollo de cada uno de nosotros**.
* Resueltos los conflictos, si los hubiese, los cambios se integran en **nuestra rama del servidor**.
* Los cambios serán informados al **coordinador** mediante un **pull request**, el cual será el que proceda a hacer el merge en la rama master o no.

Tareas mínimas a realizar **por cada alumno**

* Clonar repositorio.
* **Modificar lista de asistentes** para **incluirse en la misma**.
* **Añadir** algún **dato** o hacer algún cambio en: bebidas, comidas y materiales.
* Repetir estos cambios dos o tres veces.
* Finalizar con una copia actualizada del repositorio.

Anexo final

# Buenas prácticas para tener un portafolio atractivo en Github

Para una programadora front-end, una herramienta muy importante para visibilizar su trabajo de cara a un nuevo empleo es su perfil en GitHub. Durante el curso vamos a realizar varios ejercicios, evaluaciones y proyectos donde cada una demuestra lo que sabe y lo que puede aportar a un proyecto. Por tanto, es muy importante visibilizar ese trabajo de cara a un posible empleador.

En esta guía vamos a dar algunos **tips** para dar visibilidad a vuestro perfil de GitHub y vuestros proyectos.

# Buenas prácticas para la página de un repositorio

Para cada repositorio que queramos mostrar en nuestro perfil (como mínimo las evaluaciones de cada sprint y los proyectos) debemos comprobar que cumplimos este checklist.

* El repositorio tiene un fichero **README.md** con la descripción del proyecto y por qué se realiza (contexto en que se hace, por ejemplo)
* En la página principal del repositorio añade una **descripción**, **enlace al proyecto en GitHub Pages** y pon algún tag para que sea encontrable
* Si se trata de un proyecto de equipo, comprueba que tu contribución al proyecto aparece bien representada en la sección de contributors, con el número de commits que has realizado

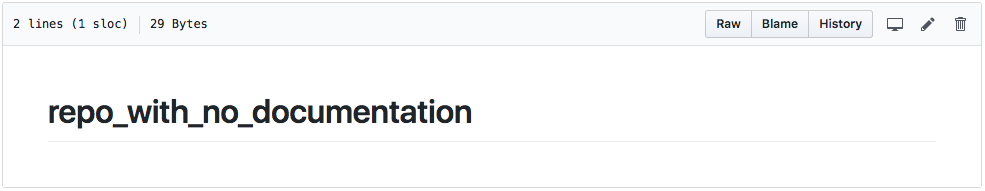
# Buenas prácticas por la página del perfil

En nuestro perfil debemos comprobar este otro checklist. Al finalizar comprueba cómo se ve tu perfil en una sesión del navegador donde no estés logueada (por ejemplo, una pestaña de incógnito) para ver cómo lo ven otras personas.

* Escribe tus datos de contacto (nombre, email), pon una foto (algo distinto a la foto por defecto) y una breve bio
* En las organizaciones a las que perteneces, activa que tu membresía sea **pública**, ya que por defecto solo es visible para ti pero no para el resto
* Personaliza los repositorios pinneados (6 máximo) que aparecen destacados en tu perfil. Puedes seleccionar repositorios a los que hayas contribuido.
* Revisa que tienes una actividad constante en GitHub (grid de cuadraditos verdes), que cuenta desde commits (mergeados a master) como PRs, issues, etc.
* Demuestra actividad siguiendo a otros perfiles interesantes y marcando proyectos molones como favoritos
* *BONUS*: crea tu propia página personal en un repositorio con nombre user.github.io para enseñar al mundo lo que sabes hacer como front

# Acerca de los archivos README

Hace no mucho tiempo llegué a un repositorio de un proyecto y me encuentro con esto:



Ni siquiera tenían una descripción de lo que hacía, para qué era y peor aún, ni siquiera cómo se instala localmente para probarlo o agregar nuevas cosas; en base a esto llegué a la conclusión de que la documentación mínima para un proyecto debe tener al menos un [**README.md**](https://en.wikipedia.org/wiki/README)que tenga sentido, para evitar que alguien llegue a un nuevo proyecto y se tope con algo como lo que les mencionaba.

A continuación, les describo una lista de lo que personalmente creo que un README.md debería contener para que cualquier nuevo desarrollador al llegar a un proyecto, con solo ver este archivo, pueda darse una idea general de lo que trata y cómo funciona.

Puedes agregar un archivo README a tu repositorio para comentarle a otras personas por qué tu proyecto es útil, qué pueden hacer con tu proyecto y cómo lo pueden usar.

Un archivo **README** suele ser el primer elemento que verá un visitante cuando entre a tu repositorio. Los archivos **README** habitualmente incluyen información sobre:

* Qué hace el proyecto.
* Por qué el proyecto es útil.
* Cómo pueden comenzar los usuarios con el proyecto.
* Dónde pueden recibir ayuda los usuarios con tu proyecto
* Quién mantiene y contribuye con el proyecto.
* Librerías utilizadas /requeridas
* Test
* ……………………..

## ¿Qué es Markdown?

[**Markdown**](http://daringfireball.net/projects/markdown/) es una forma de [aplicar](http://daringfireball.net/projects/markdown/) estilo al texto en la web. Usted controla la visualización del documento; dar formato a las palabras en negrita o cursiva, agregar imágenes y crear listas son solo algunas de las cosas que podemos hacer con Markdown. En su mayoría, Markdown es solo texto normal con algunos caracteres no alfabéticos, como #o \*.

Puede usar **Markdown** en la mayoría de los lugares de GitHub:

* Archivos con la extensión .md o .markdown

# Markdown: el lenguaje de estilos para los README.md

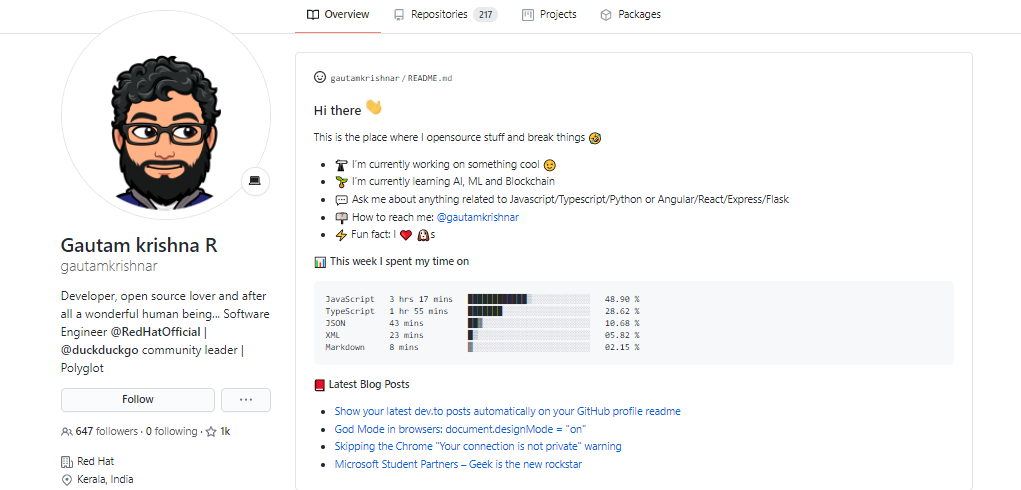
A continuación una cheat-sheet con las **marcas** disponibles para la creación de los [*README.md*](http://readme.md/)



Por último, es darle vida a nuestro archivo o fichero README.md editando a tu gusto. Los ficheros README.md los editas con el formato [GitHub Markdown](https://guides.github.com/features/mastering-markdown/) puedes utilizar <https://stackedit.io/> para tener una vista previa o utilizar un editor de código como <https://code.visualstudio.com/> para editar el archivo README.md después de clonar tu repositorio.

**NOTA:** si no tienes mucho conocimiento en Markdown te recomiendo utilizar [github Profile Readme Generator](https://rahuldkjain.github.io/gh-profile-readme-generator/) para que partas de una base y puedas ir personalizando tu contenido.

Un ejemplo de **perfil**



# Un ejemplo de Readme.md

**## Synopsis**

This PHP project pretends ...

**## External libraries used**

- Library for to query Jira with a PHP API: https://github.com/FayP/JIRA-php

- Composer

- PHP Gitlab API: https://travis-ci.org/m4tthumphrey/php-gitlab-api.svg?branch=master)](https://travis-ci.org/m4tthumphrey/php-gitlab-api

- Monolog

- Buzz http client: https://github.com/kriswallsmith/Buzz

**## Contributors**

Name Surname1 (mail@domain.com), Name Surname1 (mail2@domain.com)

**## Coding conventions**

**### Coding Style:**

For this project, every developer must follow the PSR coding conventions. It does not apply to third party library (PSR-1 and PSR-2 specifications should be enough for the most part of the project):

- PSR-1 (Basic Coding Standard): http://www.php-fig.org/psr/psr-1/

- PSR-2 (Coding Style Guide): http://www.php-fig.org/psr/psr-2/

- PSR-3 (Logger Interface): http://www.php-fig.org/psr/psr-3/

- PSR-4 (Autoloader):  http://www.php-fig.org/psr/psr-4/

- PSR-6 (Caching Interface): http://www.php-fig.org/psr/psr-6/

- PSR-7 (HTTP Message Interface): http://www.php-fig.org/psr/psr-7/

**### Method names:**

When we are setting or getting data, we must to start the method name with the action. Examples:

getIssueMetadata()

setIssueMetadata

**### Documentation:**

For documentation, we must follow PHP documentor convention: http://www.phpdoc.org/docs/latest/index.html

**## License**