

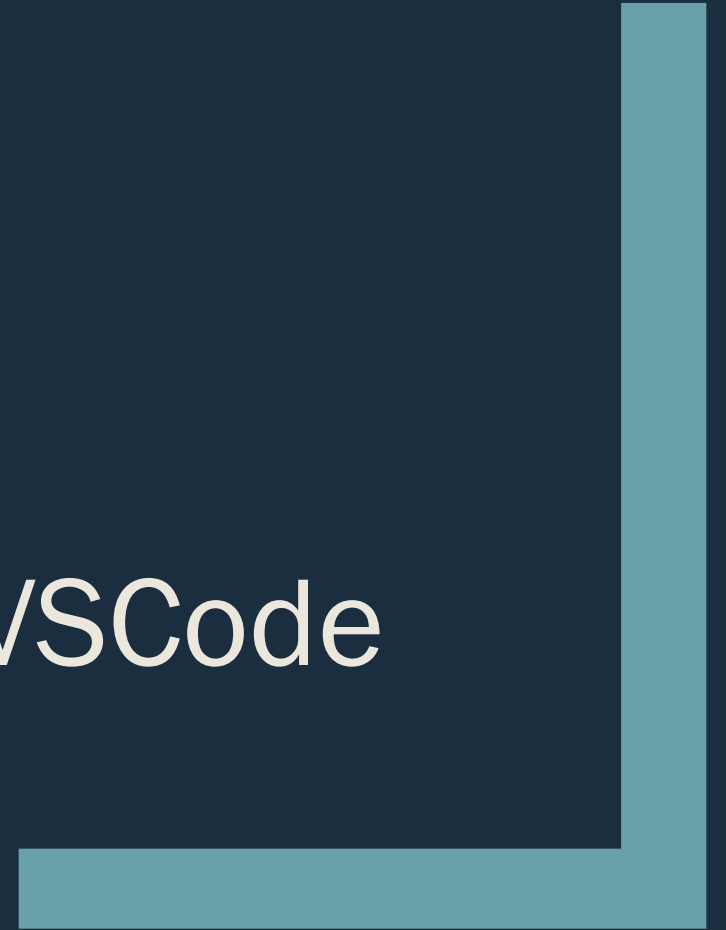


BACKEND CON PYTHON



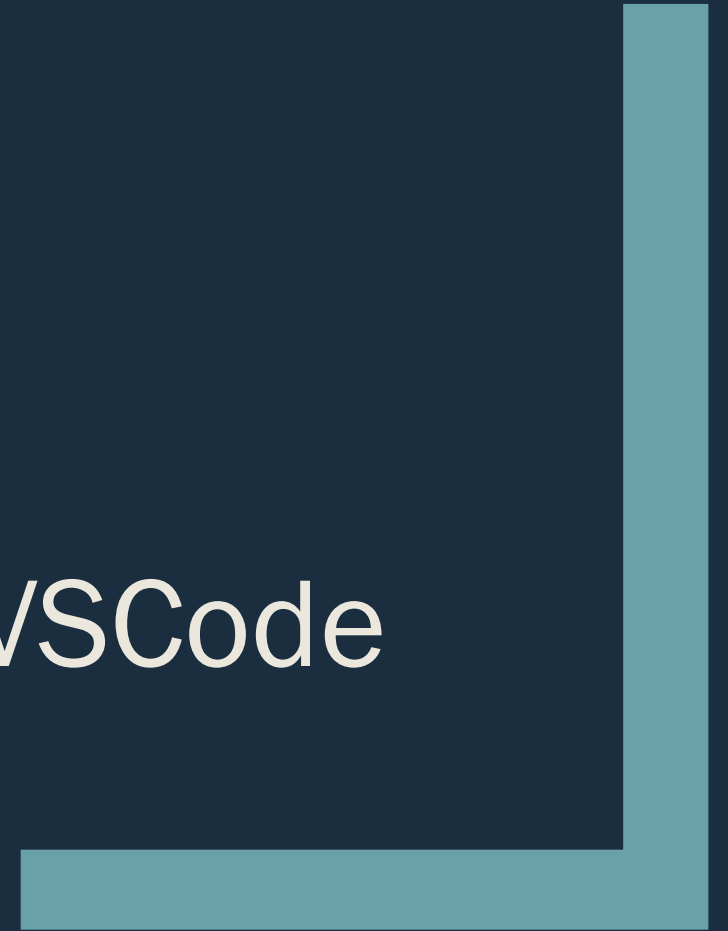
TRABAJO COLABORATIVO

- Manejo de versiones
- Comandos de Git
- Integración de GitHub con VSCode



TRABAJO COLABORATIVO

- Manejo de versiones
- Comandos de Git
- Integración de GitHub con VSCode

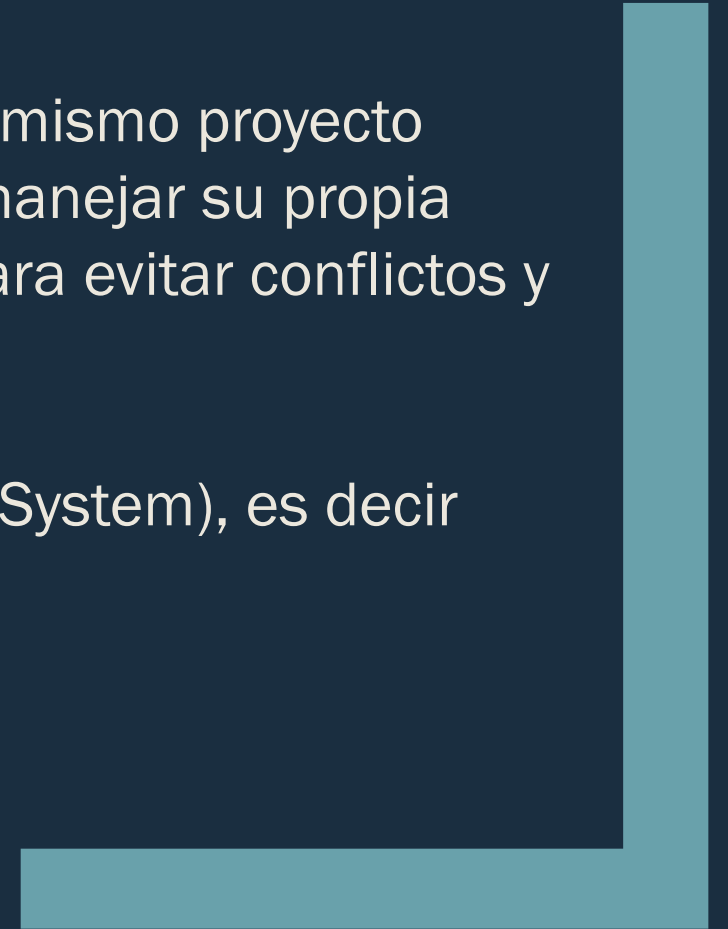


MANEJO DE VERSIONES

El desarrollo de software por lo general, por no decir siempre, se realiza en grupos de trabajo.

Para permitir a varios miembros de un grupo trabajar en el mismo proyecto simultáneamente, existen herramientas que les permiten manejar su propia versión a cada uno y luego comparar y unificar el trabajo para evitar conflictos y errores al unirlos.

Estas herramientas se conocen como VCS (Version Control System), es decir manejador de versiones.



MANEJO DE VERSIONES

Ventajas de usar un VCS:

- Control de versiones: se almacenan versiones del proyecto en diferentes momentos durante su desarrollo, esto permite hacer un seguimiento a las modificaciones realizadas.
- Colaboración: permite a diferentes personas trabajar sobre el mismo proyecto simultáneamente.
- Manejo de conflictos: si dos o más personas realizan cambios a la misma parte del código, al momento de unificar permite comparar las diferentes versiones y facilita la toma de decisiones.
- Reversión de cambios: si los cambios generan problemas, se puede devolver a una versión anterior libre de errores.
- Ramificación: permite crear ramas (branch), para crear nuevas características sin afectar la rama principal (trunk/master)

MANEJO DE VERSIONES

En internet encontraremos diversos VCS:

- Subversion: Utiliza modelo cliente-servidor y sirve para manejar versiones de manera centralizada.
- Git: es un VCS distribuido, es decir, cada colaborador tiene su propia copia. Facilita el trabajo descentralizado y realizar cambios locales, guardar el historial de estos, antes de compartarlos con el resto del equipo.
- Perforce: sistema centralizado y se usa mayormente en sistemas empresariales.
- Bazaar: sistema distribuido, busca ser flexible y darle una experiencia fácil de usar al usuario.

Por su facilidad de uso, su amplia comunidad y soporte, usaremos Git para este curso. Además, es gratis.

MANEJO DE VERSIONES



Git nos permitirá manejar commits localmente. Los commits son las versiones que guardan todos los cambios en uno o más archivos.

Podremos crear ramas para tener ramas diferente y cambiar entre ellas, unificar, comparar, entre otros.

Sin embargo, para trabajar en un equipo de trabajo tendremos que usar alguna plataforma que funcione como servidor y permita que varios usuarios se conecten envíen los cambios, sus versiones, sus ramas y se pueda unificar la información de todos los miembros de trabajo.

Con este fin utilizaremos GitHub, del que hablaremos más adelante.



TRABAJO COLABORATIVO

- Manejo de versiones
- Comandos de Git
- Integración de GitHub con VSCode



COMANDOS DE GIT

- `git init`: Inicializa un nuevo repositorio de Git en el directorio actual.
- `git clone <url>`: Clona un repositorio existente en la máquina local desde una URL.
- `git add <archivo>` o `git add .`: Añade cambios de un archivo específico, o de todos los archivos del directorio (si usa “.”)
- `git commit -m "mensaje"`: Guarda los cambios en una versión (llamado commit) con un mensaje descriptivo.
- `git status`: Muestra el estado actual del repositorio, indicando archivos modificados, añadidos o pendientes de confirmar.

COMANDOS DE GIT

- `git log`: Muestra el historial de commits con detalles como autor, fecha y mensaje.
- `git pull`: Obtiene y fusiona cambios desde un repositorio remoto a la rama (branch) actual.
- `git push`: Envía cambios locales a un repositorio remoto.
- `git branch`: Muestra una lista de ramas y resalta la rama actual.
- `git checkout <rama>`: Cambia a la rama especificada.
- `git merge <rama>`: Fusiona los cambios de otra rama en la rama actual.

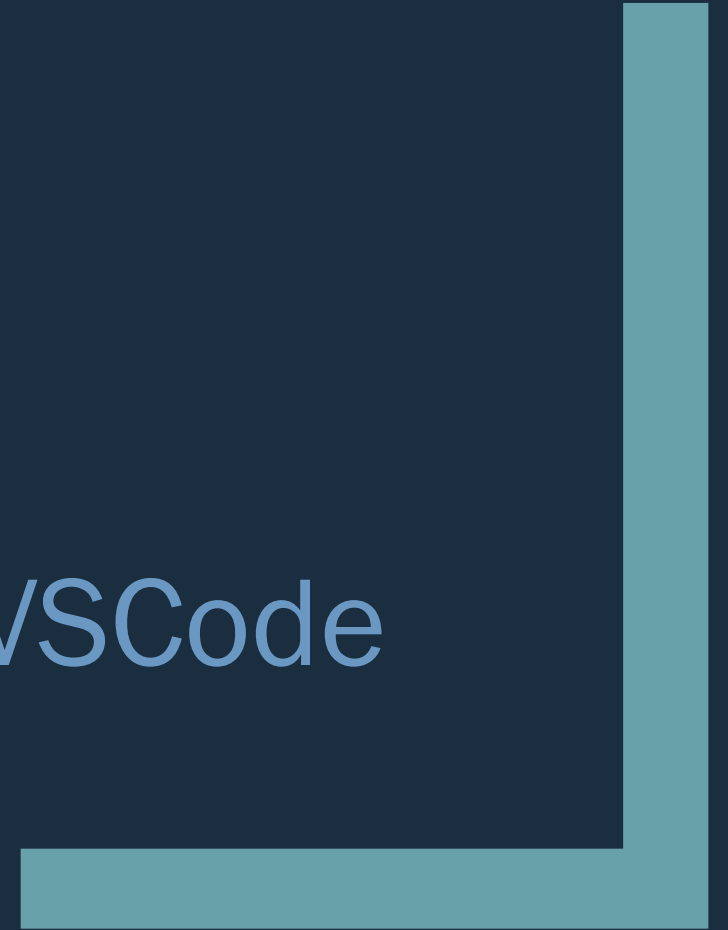
COMANDOS DE GIT

- `git remote -v`: Muestra las URL de los repositorios remotos configurados.
- `git fetch`: Recupera cambios del repositorio remoto, pero no los fusiona automáticamente.
- `git diff`: Muestra las diferencias entre los cambios sin confirmar y la última confirmación.
- `git reset <archivo>`: Deselecciona un archivo del área de trabajo.

Hay muchos otros comandos en git, y pueden ser consultados en la documentación oficial.

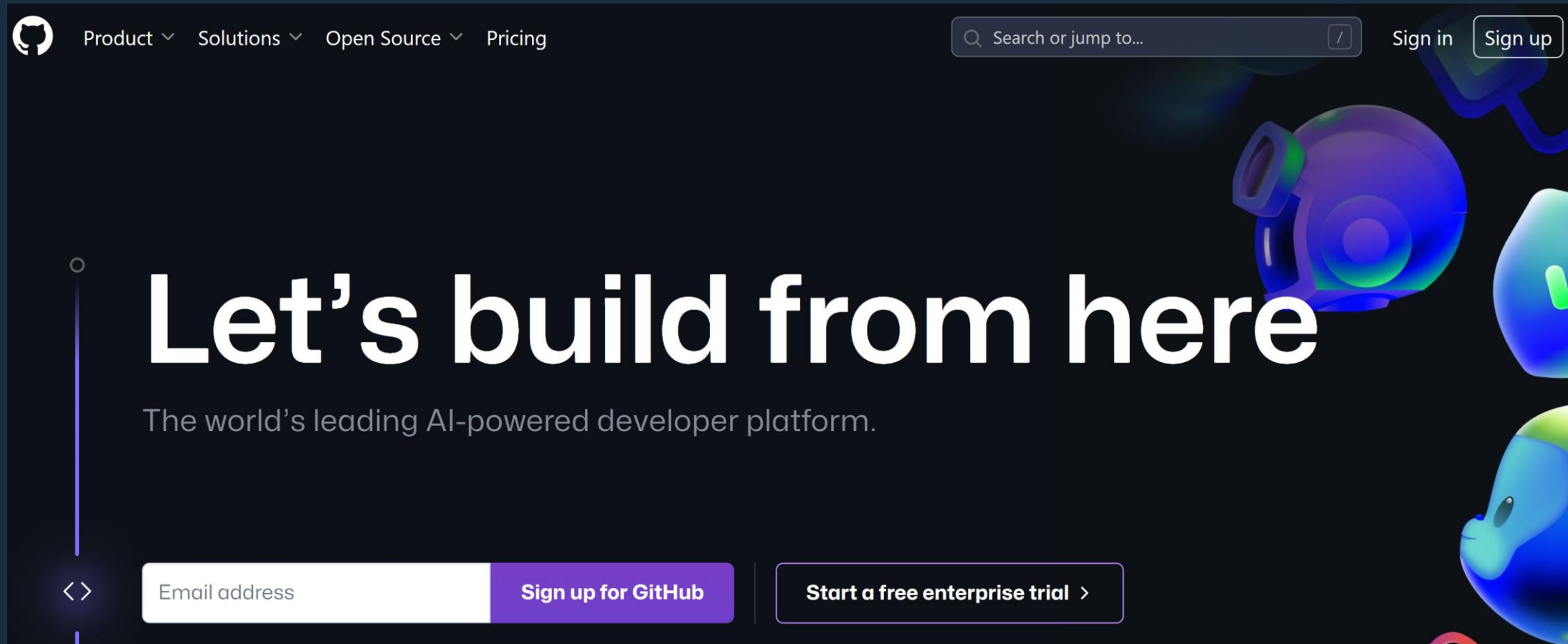
TRABAJO COLABORATIVO

- Manejo de versiones
- Comandos de Git
- Integración de GitHub con VSCode



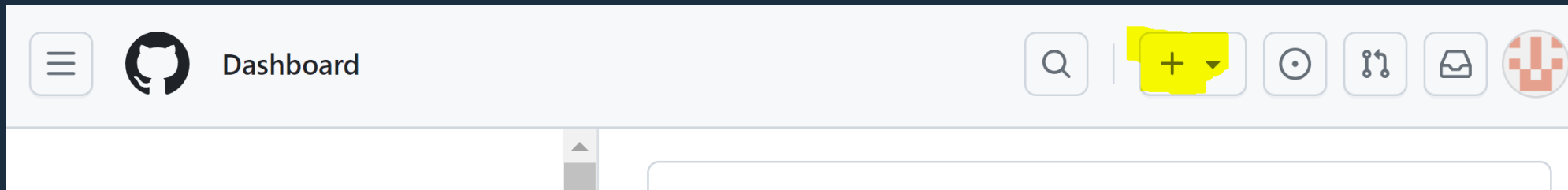
INTEGRACIÓN GITHUB Y VSCODE

Comencemos entrando a github.com, creando una cuenta, si ya teníamos, iniciamos sesión.



INTEGRACIÓN GITHUB Y VSCODE

Creamos un nuevo repositorio, ejemplo: ejemplo_division



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository else [Import a repository.](#)


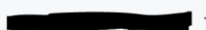
Required fields are marked with an asterisk ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * **Repository name ***

  /

✔ ejemplo_division is available.

Great repository names are short and memorable. Need inspiration? How about [studious-octo-bassoon](#)

INTEGRACIÓN GITHUB Y VSCODE

En la Terminal de VSCode seguir las instrucciones que ofrece el repositorio de Github recientemente creado

...or create a new repository on the command line

```
echo "# ejemplo_division" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/germanromero/ejemplo_division.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/germanromero/ejemplo_division.git
git branch -M main
git push -u origin main
```

MANOS A LA OBRA

Taller 3 disponible en BloqueNeon

