GITHUB

Plataforma de colaboración formal/informal de desarrollo de software o social coding.

Para publicar repositorios que funcionan bajo el sist de ctrl de versiones de Git. Configura los proyectos nuevos como open source = los ve cualquier persona (reconfigurable).

<http://github.com/>

*Repositorio = parte de un proyecto*

**Incidencias – issues:** unidad de trabajo designada para realizar alguna mejora en un sistema informático: tarea, bugfix, característica pedida, solicitud de documentación especifico, ideas o sugerencias, etc. Cada incidencia puede ser etiquetada bajo una categoría (bug, doc, duplicado, inválido, etc.) para que permita identificar a qué tipo de incidencia corresponde cada una (se pueden crear más a gusto).

**Hitos – milestones:** grupos de incidencias que ayudan a seguir el progreso de estas:

1. Descripción del hito proporcionada por el user: info descriptiva de proyecto, equipos relevantes, fechas de vencimiento proyectadas.
2. Fecha de vencimiento del hito.
3. Porcentaje de finalización del hito.
4. Cantidad – lista de incidencias abiertas y cerradas asociadas al hito.

Proyectos GitHub

Permiten organizar incidencias y notas en categorías mediante tarjetas en columnas, las cuales se pueden arrastrar entre columnas según el estado de cada tarea.

Paneles de proyectos = personalizables y adaptables. Dentro de cada uno se pueden reordenar columnas y cartas según criterios del proyecto/organización.

Dentro de columnas o tarjetas se pueden crear notas o comments para entender incidencias.

Se pueden vincular hasta 25 repositorios a cada tablero de proyecto, lo que facilita agregar informes de problemas a través del botón + o desde la barra lateral de la pestaña Issues.

**Tableros de proyecciones**

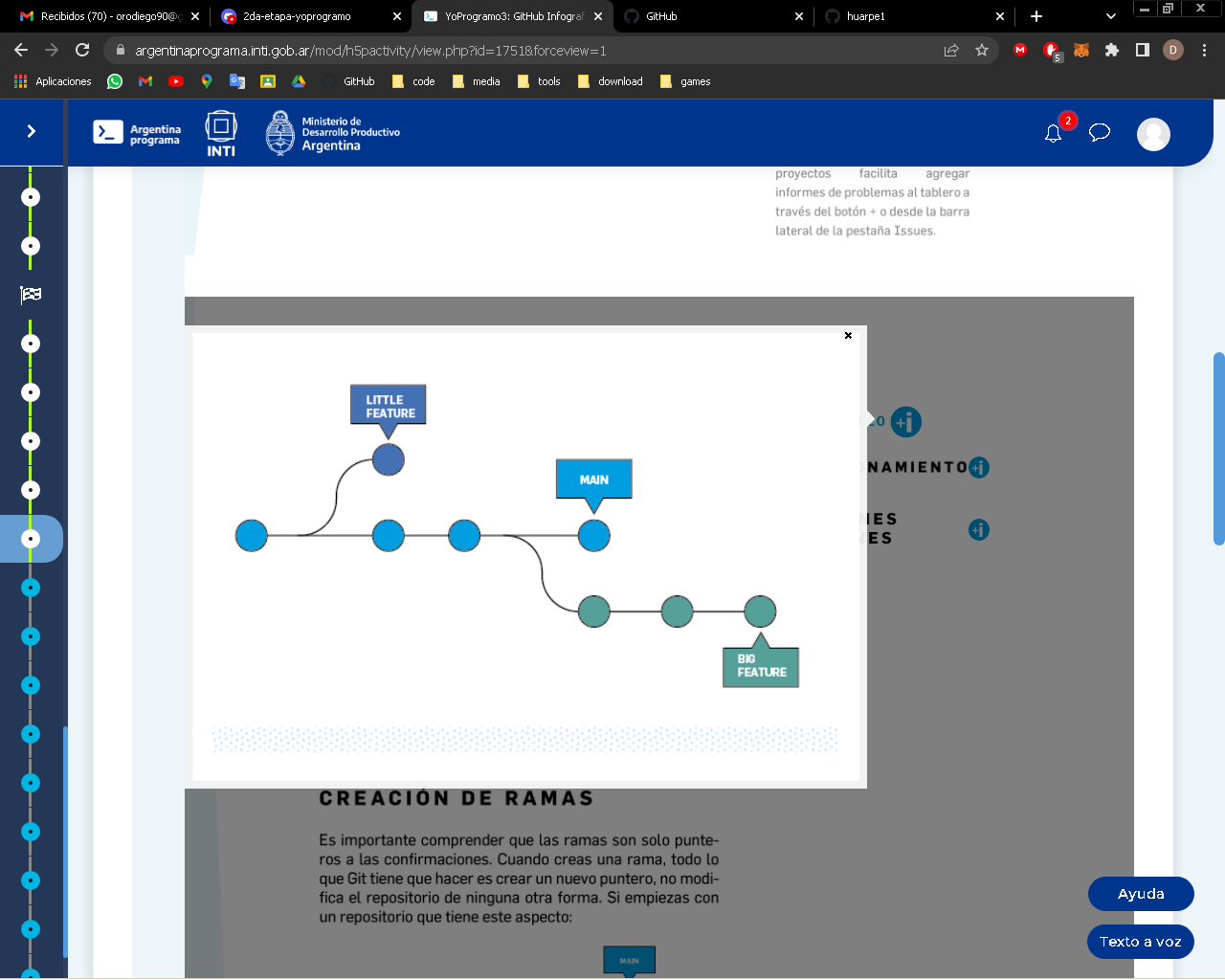
Pertenecientes al user: cualquier repositorio personal.

De proyectos a nivel de organización: repositorios de organización.

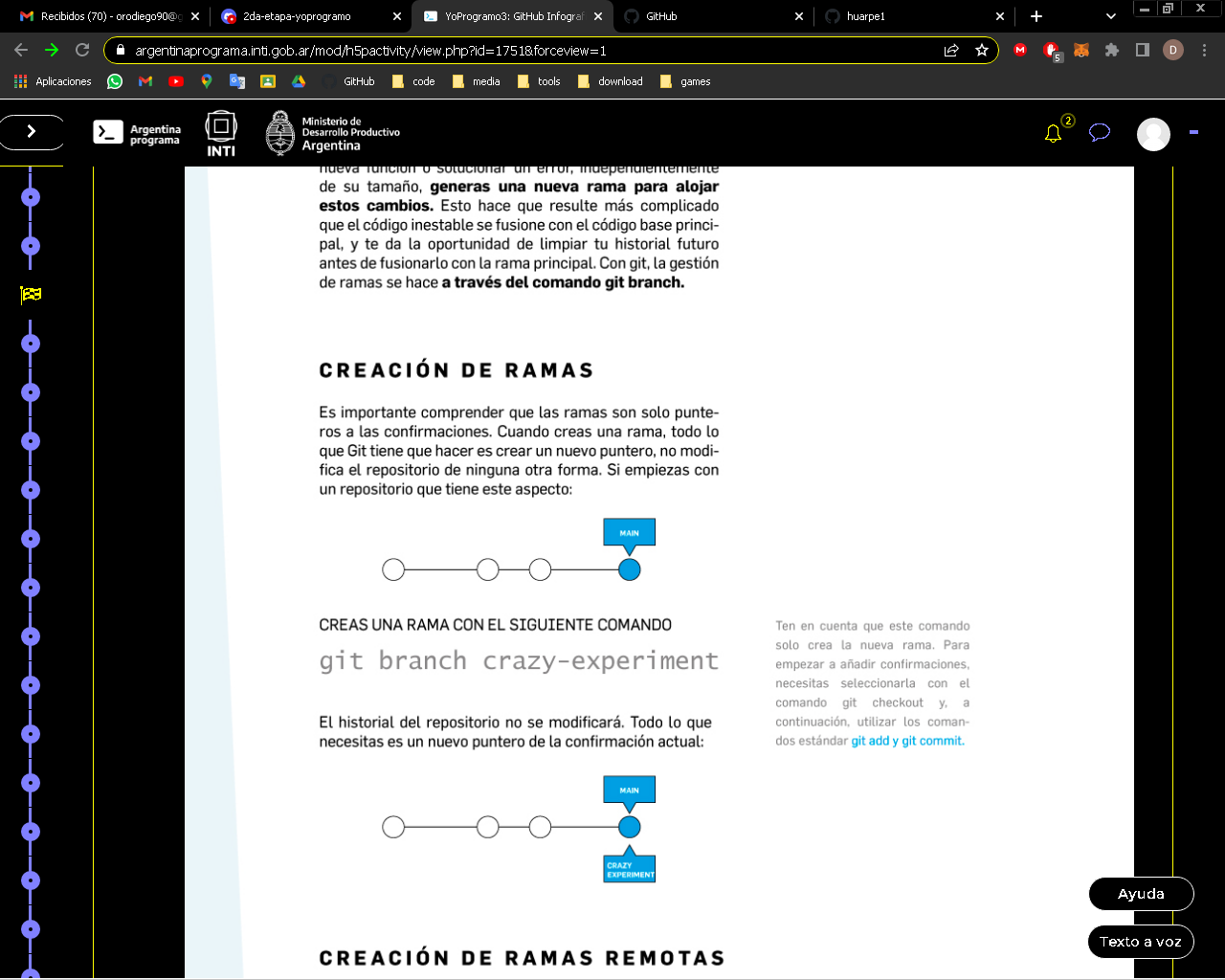
Tableros por repositorio: incidencias de un único repositorio, a veces con referencias a incidencias de otros repositorios.

Workflow - branches

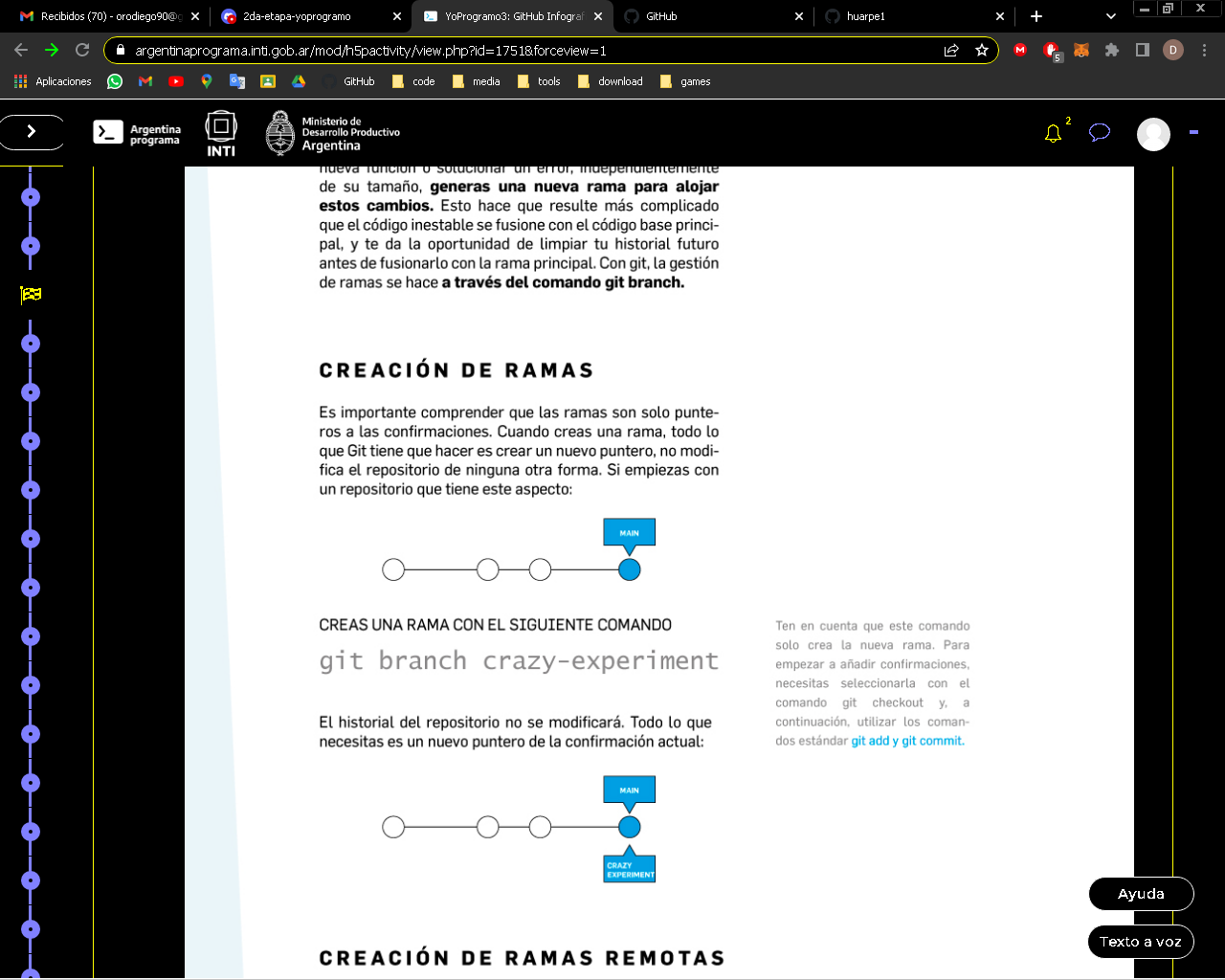
Función disponible en la mayoría de sistemas de control de versiones moderno, son parte del proceso de desarrollo diario en Git como un puntero para instantáneas de cambios. A añadir una solución, función o modificación se genera una nueva rama para alojar esos cambios = hace complicado que el código inestable se fusione con el base code principal y da la oportunidad de limpiar mi historial a futuro antes de fusionarlo con la main branch (comando *git branch*).



**Rama - Branch:** línea independiente de desarrollo, sirviendo como abstracción de procesos de cambio, preparación y confirmación; como una forma de solicitar un nuevo directorio de trabajo, entorno de ensayo o historial de proyecto. Nuevas confirmaciones se registran en historial de rama actual como bifurcación. Son solo punteros a las confirmaciones, cuando se crea una todo lo que Git hace es crear un nuevo puntero sin modificar el repositorio de ninguna forma. Si empiezo con un repositorio que se ve así:



Se crea una rama con el comando: git branch crazy-experiment, y el historial de repos. no se modifica, solo hay un nuevo puntero de la confirmación actual:



**Comandos básicos:**

git init: comando para inicializar directorio como repositorio Git, ejecutado dentro de directorio de proyecto y, como resultado, crea subdirect .git que contiene files para realizar seguimiento de cambios, etiquetas, etc.

git add <file>: después de mod., creación o eliminación de file, los cambios queda únicamente en área de trabajo, por lo que se pasan a área de preparación con este comando para incluir en la siguiente confirmación-commit.

git status: permite conocer en qué estado se encuentran los files.

git commit: confirmar cambios registrados en área de preparación, se pasan cambios a repositorio local.

git push: para enviar confirmaciones registradas en repositorio local a remoto.

git pull: inverso de git push, trae cambios a repositorio local y los deja disponibles para su modificación o revisión en área de trabajo. Se usa cuando ya se tiene repositorio local vinculado a uno remoto, al igual que con git push.

git clone: en caso de necesitar “bajar” un repositorio remoto de algún proyecto existente. Genera directorio (con nombre de repositorio o uno especificado) que contiene todo lo del proyecto, y el subdirectorio .git necesario para gestionar cambios y lo pertinente al repositorio Git.

git branch: permite crear, enumerar, renombrar, borrar o listar (todas las del repositorio) ramas. No permite cambiar entre ramas o reunificar historial ya bifurcado. Muy relacionado a comandos git checkout y git merge.

git branch <branch>: para crear rama y su nombre.

git branch –d <branch>: una vez se haya terminado de trabajar en una rama y se haya fusionado con código base principal, se puede eliminar una rama sin perder ninguna historia. Si la rama no se ha fusionado el comando anterior muestra un error “error: Te branch ‘crazy-experiment’ is not fully merged. If you are sure you want to delete it, run ‘git branch –D crazy-experiment’.” Esto protege ante la pérdida de acceso a una línea de desarrollo completa, el último comando elimina sin aviso previo.

git push origin –delete crazy-experiment / git push origin \_crazy-experiment: envía señal de eliminación de rama de repositorio remoto o de origen.

git branch –m <branch>: para renombrar rama sobre la cual se está trabajando.

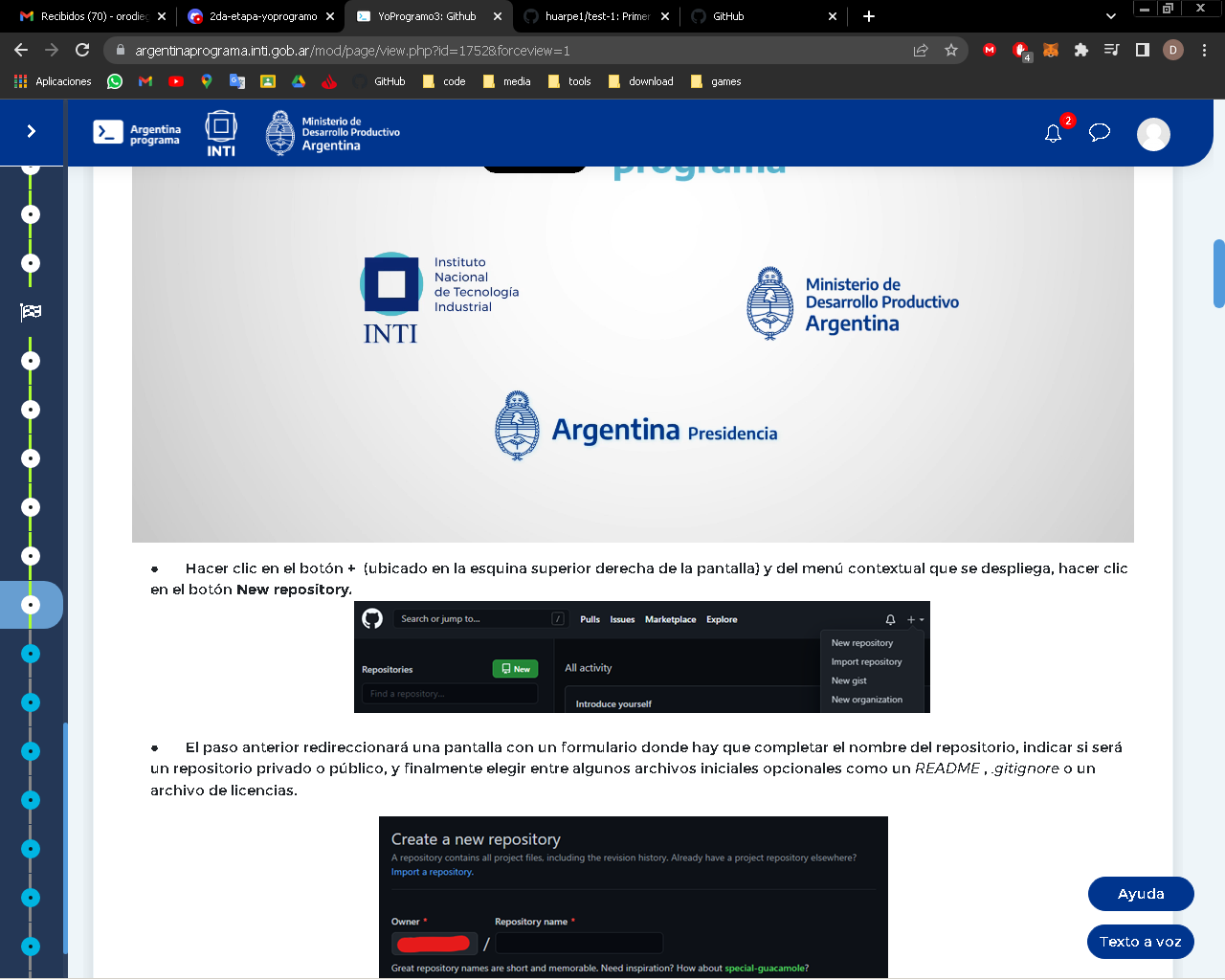
git branch –a: listar todas las ramas en el repositorio remoto.

git checkout: seleccioar

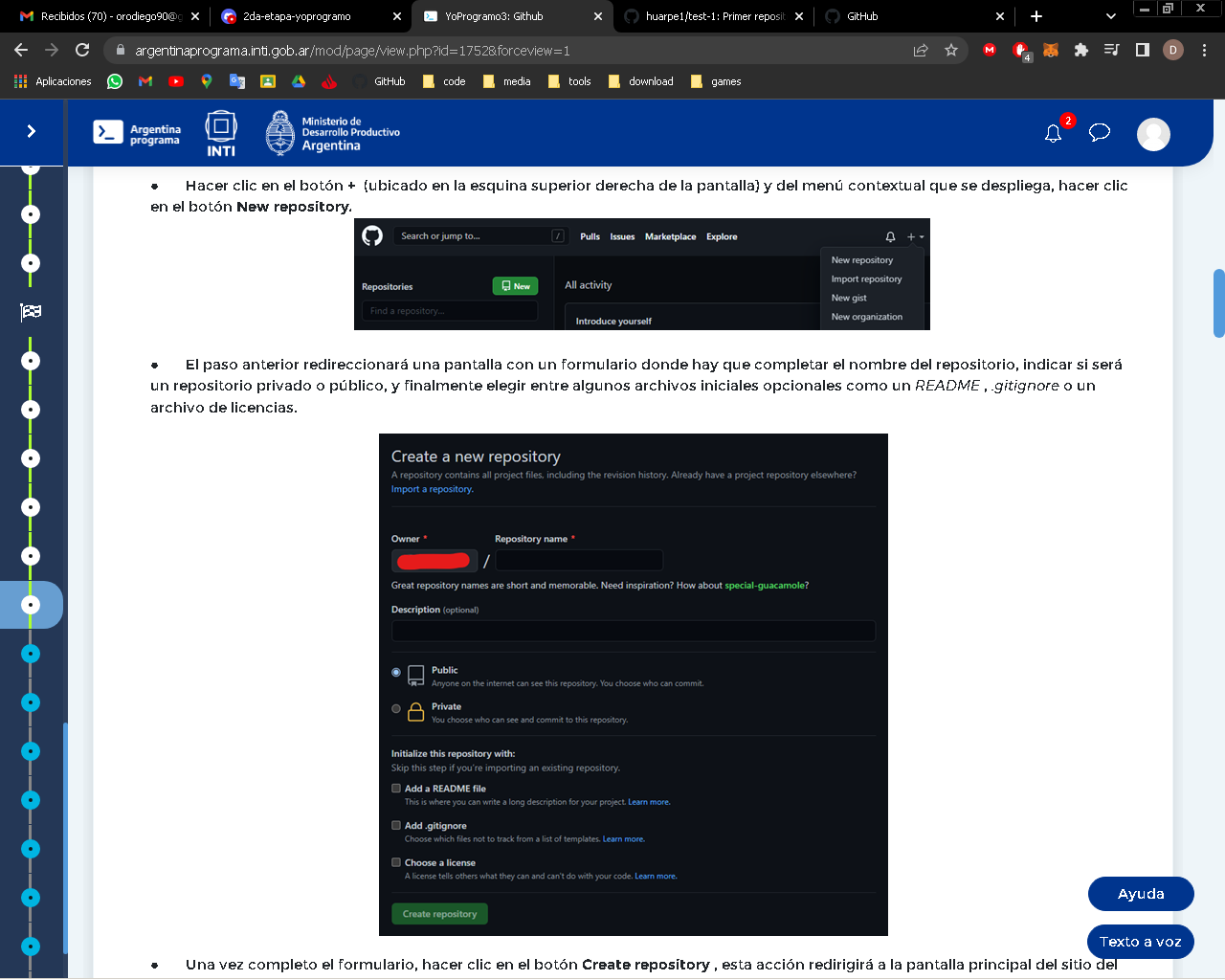
git remote add <remote-repo> <remote-repo-URL>: configurar repositorio remoto y añadirlo a al config de repositorio local.

git push <remote-repo> crazy-experiment-: envía copia de local branch crazy-experiment a repositorio remoto <remote-repo>.

**Creación de repositorio remoto en GitHub:** hacer clic en el botón + (ubicado en la esquina superior derecha de la pantalla) y del menú contextual que se despliega, hacer clic en el botón New repository.

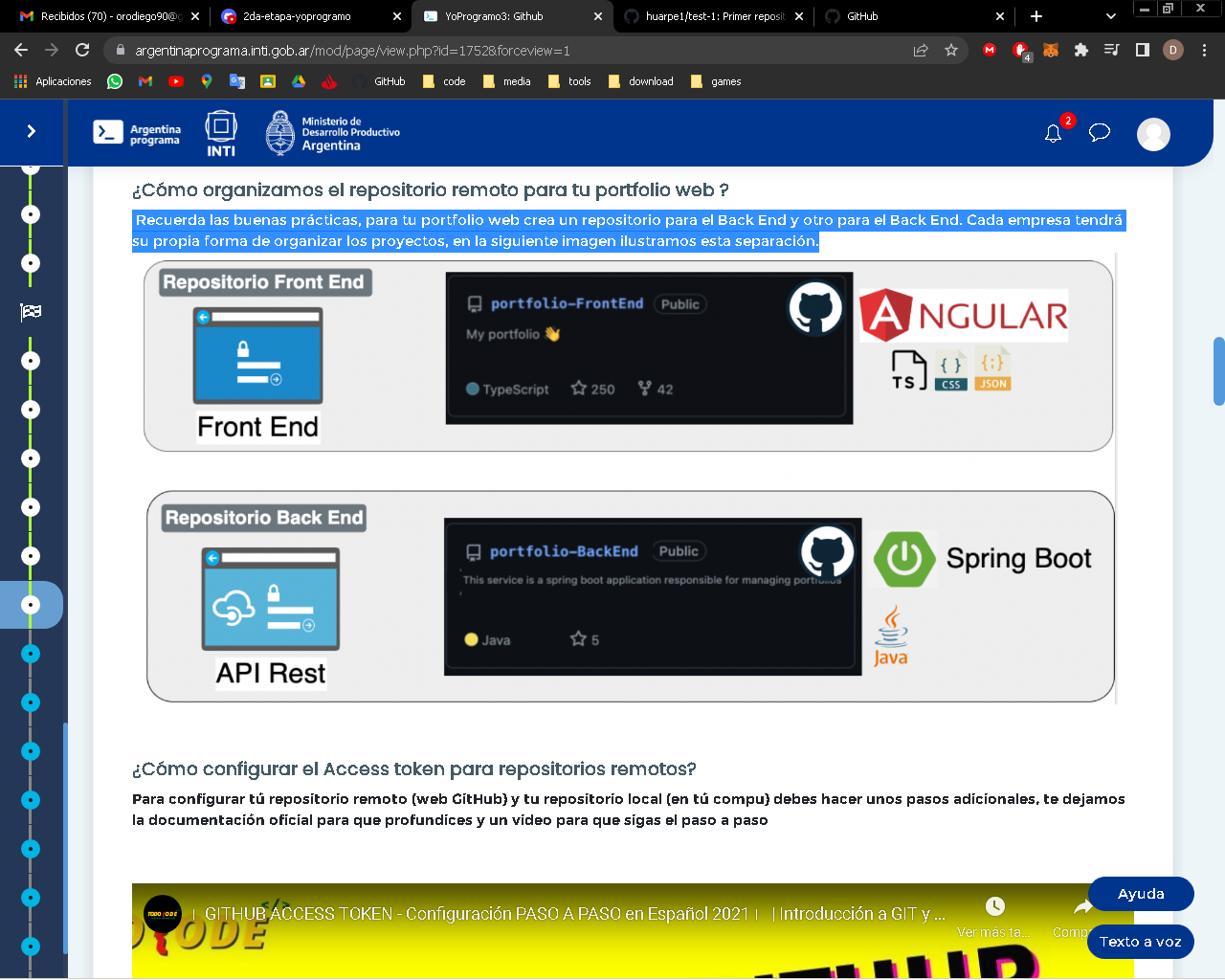


El paso anterior redireccionará una pantalla con un formulario donde hay que completar el nombre del repositorio, indicar si será un repositorio privado o público, y finalmente elegir entre algunos archivos iniciales opcionales como un README, .gitignore o un archivo de licencias.



Una vez completo el formulario, hacer clic en el botón Create repository, esta acción redirigirá a la pantalla principal del sitio del repositorio, donde se pueden observar diferentes instrucciones según la situación en la que se encuentre.

**Organización de repositorio remoto para portfolio web:** recuerda las buenas prácticas, para tu portfolio web crea un repositorio para el Back End y otro para el Front End. Cada empresa tendrá su propia forma de organizar los proyectos, en la siguiente imagen ilustramos esta separación:



**Configurar Access token para repositorios remotos:** para configurar tú repositorio remoto (web GitHub) y tu repositorio local (en tu compu) debes hacer unos pasos adicionales, te dejamos la documentación oficial para que profundices y un video para que sigas el paso a paso: <https://youtu.be/2nzOI-ynXF4>