PROYECTO BASES DE DATOS 'BICI-GO'

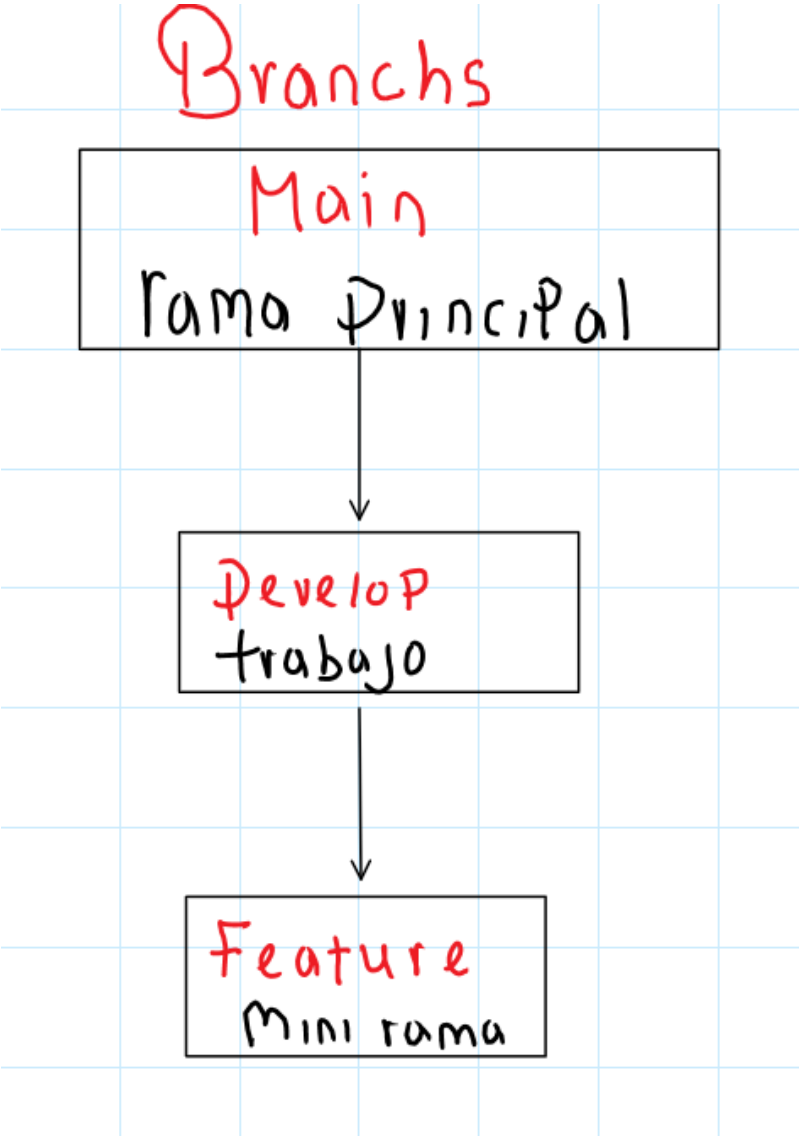
Guía del trabajo – flujos y comandos

Trabajaremos en 3 ramas:

-Main (se va mostrar al profesor)

-Develop

-feature (mini ramas temporales)



¿Por qué dos ramas?

main = estable/entregas. Aquí solo va lo que ya está completo y listo para mostrar (con tag). Evita que el profe vea cosas a medias.

develop = integración. Es la “cocina” donde juntamos el trabajo de las mini-ramas y validamos que todo funcione junto.

¿Por qué una rama temporal (mini-rama feature/*)?

Para trabajar una historia o subhistoria sin romper develop.

Se crea, se trabaja, se revisa por PR y se borra al terminar. Mantiene el historial limpio y reduce conflictos.

1) Objetivo y alcance

Trabajar por iteraciones: cada mini-rama cubre una historia/subhistoria y se integra a develop por PR.

Mantener main estable para entregas al profesor (con tags por versión).

Dejar evidencia en la bitácora (Excel) y en las descripciones de los PR.

¿Entonces, como trabajar una iteración?

Iteración por historia (H1)

H1 conceptual ☑

H1 lógico ☑

H1 físico (SQL) ☑

→ Abrir PR de la mini-rama → merge a develop.

Integración en develop

Cargar seeds, correr consultas/constraints, revisar con el equipo.

Si todo está estable → PR develop → main + tag (ej. v0.1).

Evolución

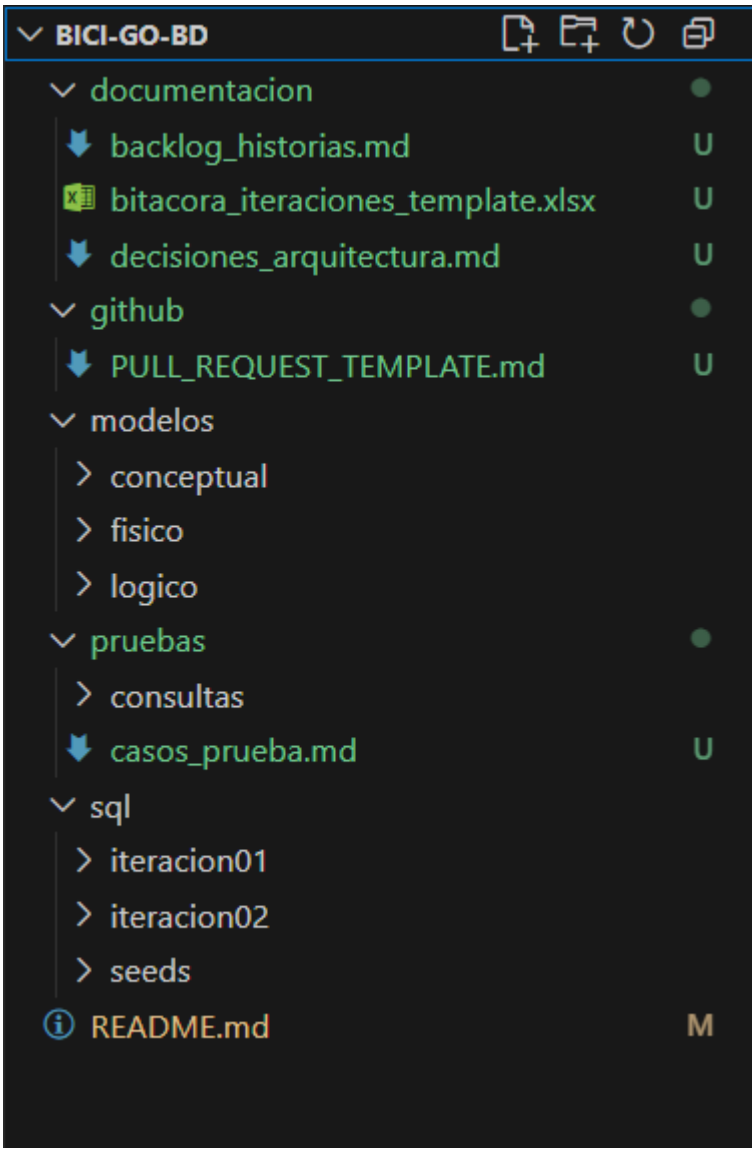
Si otra historia (o un cambio futuro) impacta H1, se crea nueva mini-rama (ajustes de H1), se integra a develop, y al cerrar la nueva iteración se publica otra versión en main (ej. v0.2).

No reescribes la versión anterior; quedas con historial (v0.1, v0.2, …).

Pequeña nota práctica: si una historia es grande, podemos dividir la iteración en sub-iteraciones (H1 conceptual → merge a develop; luego H1 lógico → merge; luego H1 físico

→ merge). Igual, el release a main lo hacemos cuando completemos el bloque previsto de la iteración (por ejemplo, H1 completa).

ARQUITECTURA DE CARPETAS



/ (raíz)

README.md: qué es el proyecto, flujo de trabajo (ramas/PR/tags) y cómo contribuir.

/documentación

backlog_historias.md: listado de historias y subhistorias con criterios.

bitacora_iteraciones.xlsx: registro de PR/iteraciones (links, fechas, cambios)

decisiones_arquitectura.md: políticas de diseño (naming, índices, normalización, constraints, motor DB).

/modelos

/conceptual: diagramas E-R (drawio/pdf/png) y notas de supuestos.

/lógico: modelo relacional (DER), normalización y diccionario de datos.

/físico: diagrama físico (opcional) y notas de implementación (índices, particiones, etc.).

/sql

/iteracion_01 (luego 02, 03…): scripts de esquema por iteración.

01_create_*.sql: creación de tablas.

02_constraints.sql: PK/FK/UNIQUE/CHECK.

03_indexes.sql: índices recomendados.

9x_migraciones.sql: cambios específicos.

/seeds: datos de ejemplo para probar (seed_*.sql).

/pruebas

/consultas: queries de validación (JOINs, filtros, agregaciones).

casos_prueba.md: qué verifica cada consulta/constraint y el resultado esperado.

/.github

PULL_REQUEST_TEMPLATE.md: plantilla que se muestra al abrir un PR (evidencia + checklist).

ALGUNAS REGLAS A TENER EN CUENTA

Mensajes de commit (Conventional Commits)

feat(H1, conceptual): E-R inicial Bicicletas

fix(H1, logico): agregar CHECK anio_fabricacion

chore(docs): actualizar bitácora

test(pruebas): q01 bicicletas disponibles

Convenciones de nombres

Tablas/columnas: snake_case, sin tildes, sin espacios.

Ramas: feature/<historia>-<alcance> (ej. feature/h1-bicicletas-logico).

Bitácora (disciplina)

Una fila por PR/iteración: historia, rama, PR, cambios clave, fechas, tag (si aplica).

Guardar en documentacion/bitacora_iteraciones.xlsx.

PARTE PRACTICA (INICIALIZAR REPOSITORIO POR PRIMERA VEZ, CREACION DE MINI RAMAS, CAMBIO DE RAMAS, ETC ETC)

1) Instalación de Git (primera vez en el PC)

Windows

Descarga Git for Windows: https://git-scm.com/download/win

(incluye Git Bash y Git Credential Manager).

Instala con valores por defecto.

Abre Git Bash o PowerShell y configura:

git --version

git config --global user.name "Tu Nombre"

git config --global user.email "tu@email.com"

Recomendado en Windows para saltos de línea:

git config --global core.autocrlf true

Pull simple (merge por defecto):

git config --global pull.rebase false

gh auth login

Una vez instalado git y tener la sesión sincronizada, seguimos esta serie de pasos.

 # 0) clonar el repositorio (traerlo a tu PC) solo por primera vez en tu pc

git clone https://github.com/andr4f/bici-go-bd.git

cd bici-go-bd

 # 1) partir actualizado (SIEMPRE antes de crear o actualizar tu rama)

git checkout develop

git pull

 # 2) crear tu mini-rama desde develop actualizado

git checkout -b feature/<mi-tarea>

 # 3) trabajar y guardar en tu rama

git add .

git commit -m "feat(Hx): mensaje claro"

git push -u origin feature/<mi-tarea> # (-u solo la PRIMERA vez de esta rama)

 # 4) mantener tu rama al día si develop avanzó

```
git checkout develop
git pull
git checkout feature/<mi-tarea>
git merge develop
git push

# 5) abrir el Pull Request (PR)
# --- Opción 1 (web): en GitHub → "Compare & pull request"
#     base=develop, compare=feature/<mi-tarea> → completar plantilla → Create PR

# 6) tras aprobación: merge del PR a develop (en GitHub) y limpiar ramas
git push origin --delete feature/<mi-tarea>   # borra rama remota
git branch -d feature/<mi-tarea>              # borra rama local

# 7) nueva tarea otro día
git checkout develop
git pull
git checkout -b feature/<nueva-tarea>
```