Guía de Git para el Equipo - Proyecto La Palma

1. Configuración inicial (Solo la primera vez)

Instalar Git

- Descargar desde: https://git-scm.com/download/win
- Durante la instalación, usar configuraciones por defecto

Configurar identidad en Git

bash

```bash
git config --global user.name "Tu Nombre Completo"
git config --global user.email "tu-email@ejemplo.com"
```

Clonar el repositorio

bash

```bash
cd Desktop  # O donde quieras tener el proyecto
git clone https://github.com/MrVargas00/La-Palma.git
cd La-Palma
```

Configurar la base de datos

1. Copiar `backend/LaPalma/LaPalma/web.config.example` como `web.config`
2. Editar `web.config` y cambiar `TU-SERVICIO-DE-SQL` por tu servidor local
3. Ejecutar el script SQL de la carpeta `database/` en tu base de datos

2. Flujo de trabajo con ramas

Antes de empezar a trabajar (SIEMPRE)

bash

```bash
# Ir a la rama principal
git checkout main

# Traer los últimos cambios
git pull origin main
```

Crear rama para nueva funcionalidad

bash

```bash
# Crear y cambiar a nueva rama
git checkout -b feature/nombre-de-tu-funcionalidad

# Ejemplos de nombres de rama:
git checkout -b feature/sistema-login
git checkout -b feature/crud-productos
git checkout -b bugfix/error-conexion-db
git checkout -b hotfix/validacion-formularios
```

Trabajar en tu rama

bash

```bash
# Ver en qué rama estás
git branch

# Hacer cambios en tus archivos...

# Ver qué archivos cambiaste
git status

# Agregar cambios
git add .

# Hacer commit
git commit -m "Descripción clara de lo que hiciste"

# Subir tu rama a GitHub
git push origin feature/nombre-de-tu-funcionalidad
```

Fusionar tu trabajo con la rama principal

Opción A: Pull Request (RECOMENDADA)

1. Ve a GitHub.com → MrVargas00/La-Palma
2. Click en "Compare & pull request"
3. Describe qué hiciste
4. Asigna a alguien para revisar
5. Una vez aprobado, se fusiona automáticamente

Opción B: Merge directo (solo para cambios pequeños)

bash

```bash
git checkout main
git pull origin main
git merge feature/tu-rama
git push origin main
git branch -d feature/tu-rama  # Eliminar rama local
```

3. Comandos útiles diarios

Información del estado

bash

```bash
git status              # Ver estado actual
git branch              # Ver todas las ramas locales
git branch -a           # Ver todas las ramas (locales y remotas)
git log --oneline       # Ver historial de commits
```

Cambiar entre ramas

bash

```bash
git checkout main                    # Ir a rama principal
git checkout feature/mi-rama         # Ir a mi rama
git checkout -b feature/nueva-rama   # Crear y cambiar a nueva rama
```

Actualizar tu trabajo

bash

```bash
git pull origin main           # Traer cambios de la rama principal
git pull origin tu-rama        # Traer cambios de tu rama específica
```

4. Buenas prácticas del equipo

Nombres de ramas

- `feature/nombre` - Para nuevas funcionalidades
- `bugfix/nombre` - Para corregir errores
- `hotfix/nombre` - Para correcciones urgentes
- Usar guiones en lugar de espacios: `feature/sistema-login`

Mensajes de commit

✅ Buenos ejemplos:

bash

```
"Agregado sistema de autenticación de usuarios"
"Corregido error en validación de formulario de productos"
"Actualizada interfaz de dashboard principal"
"Implementada conexión con API de pagos"
```

❌ Malos ejemplos:

bash

```
"cambios"
"fix"
"update"
"asdf"
```

Flujo de trabajo recomendado

1. **Una rama por funcionalidad** - No mezclar diferentes características
2. **Commits frecuentes** - No esperar días para hacer commit
3. **Pull antes de push** - Siempre actualizar antes de subir cambios
4. **Pull Requests** - Para cambios importantes, usar revisión de código
5. **No trabajar directo en main** - Siempre usar ramas

5. Resolución de problemas comunes

Error: "Your branch is behind"

bash

```
git pull origin main
```

Error: "Merge conflict"

bash

```
# Git te dirá qué archivos tienen conflictos
# Editar manualmente esos archivos
# Buscar líneas como <<<<<<< HEAD y >>>>>>>
# Decidir qué código mantener
git add .
git commit -m "Resuelto conflicto de merge"
```

Deshacer cambios no guardados

bash

```bash
git checkout -- archivo.cs        # Deshacer cambios en un archivo
git checkout -- .                 # Deshacer todos los cambios no guardados
```

Ver diferencias antes de commit

bash

```bash
git diff                          # Ver todos los cambios
git diff archivo.cs               # Ver cambios en archivo específico
```

6. Ejemplo completo de flujo de trabajo

bash

```bash
# 1. Actualizar proyecto
git checkout main
git pull origin main

# 2. Crear rama para nueva funcionalidad
git checkout -b feature/registro-usuarios

# 3. Trabajar en archivos (Visual Studio, etc.)

# 4. Guardar progreso
git add .
git commit -m "Implementada página de registro de usuarios"
git push origin feature/registro-usuarios

# 5. Continuar trabajando y hacer más commits...
git add .
git commit -m "Agregada validación de email en registro"
git push origin feature/registro-usuarios

# 6. Cuando esté listo, crear Pull Request en GitHub

# 7. Después de que se apruebe el PR, limpiar
git checkout main
git pull origin main
git branch -d feature/registro-usuarios
```

7. Contacto para dudas

Si tienes problemas o dudas:

1. Revisar esta guía

2. Buscar el error en Google
3. Preguntar en el chat del equipo
4. En último caso, hacer screenshot del error y compartir

¡Importante! Nunca hagas `git push --force` en la rama `main`. Siempre usa ramas para trabajar.