Regola d’oro: *un task = un branch = una Pull Request (PR).*
Non si lavora mai su `main` o `dev`. Si lavora **solo** sul proprio branch `feature/...`.

Passo 0 — Accetta l’invito

- Controlla l’email/notification su GitHub e **accetta** l’accesso alla repo.

Passo 1 — Clona e prepara

```
git clone https://github.com/leacappello/basilica-platform-project.git
cd basilica-platform-project

# spostati sul ramo di integrazione
git checkout dev
git pull origin dev
```

Passo 2 — Vai sul tuo branch di feature

```
# scarica tutti i riferimenti remoti
git fetch origin

# crea e traccia il tuo branch locale collegandolo al remoto
git checkout --track origin/feature/<il-tuo-branch>
# Esempio:
# git checkout --track origin/feature/01-hero-banner

(In alternativa: git switch -c feature/01-hero-banner --track origin/feature/01-
hero-banner)
```

Passo 3 — Lavora, committa, pusha

```
# modifica i file del tuo task
git add .
git commit -m "feat(hero): implementazione layout e overlay"
git push
```

Passo 4 — Apri la Pull Request (PR) verso `dev`

- Vai su GitHub → repo → **Compare & pull request** (compare in alto).
- **Base**: `dev` — **Compare**: `feature/...` (il tuo).
- Titolo: `feat(<sezione>): breve descrizione`
- Descrizione: scrivi **Closes #<numero-issue>** (collega la tua Issue).

- Invia la PR e **richiedi 1–2 review**.

Passo 5 — Itera con le review

- Se ricevi commenti: fai le modifiche → `git add .` → `git commit -m "fix: ..."` → `git push`.
- La PR si aggiorna da sola. Quando approvata e i check passano, verrà mergiata in `dev`.

Passo 6 — Tieniti allineato con `dev` (se la PR tarda)

Se `dev` avanza mentre lavori:

```
# salva il lavoro
git add .
git commit -m "chore: savepoint"

# porta nel tuo branch gli ultimi cambi di dev
git checkout dev
git pull origin dev
git checkout feature/<il-tuo-branch>
git merge dev          # oppure: git rebase dev (se sai già usarlo)
# risolvi eventuali conflitti, poi:
git add .
git commit -m "merge: allineamento con dev"
git push
```

Chi fa cosa (abbinamento branch ↔ studenti)

- `feature/01-hero-banner` → Cancilla, Ventura
- `feature/02-header-nav` → La Bruna, Mistretta
- `feature/03-news` → Giordano
- `feature/04-places` → Pandolfo
- `feature/05-main-links` → Arpaia
- `feature/06-main-media` → Lovero
- `feature/07-main-refine` → Carparotta
- `feature/08-sidebar-cta` → Chianese
- `feature/09-footer` → Trapanese, Giordano
- `feature/10-internal-pages` → Palmigiani

Troubleshooting rapido

- **“permission denied” / “protected branch”** quando fai push: stai provando a pushare su `main`/`dev`.
  → Soluzione: assicurati di essere sul tuo `feature/...` e fai push lì.
- **“Updates were rejected because the remote contains work…”**
  → Fai prima `git pull --rebase` o allinea con `dev` (vedi Passo 6), poi riprova il push.
- **Non trovi il branch remoto**
  → `git fetch origin` poi `git branch -r` deve mostrare `origin/feature/<tuo-branch>`.
  → Se non c’è, chiedi al docente oppure crealo:
  `git checkout -b feature/<tuo-branch>` → `git push -u origin feature/<tuo-branch>`.

Mini-checklist

- Lavoro **solo** sul mio `feature/...`
- Commit piccoli e descrittivi (`feat:`, `fix:`, `docs:`…)
- PR verso `dev` con “Closes #issue”
- Rispetto: semantica, accessibilità, responsive, performance
- Se `dev` avanza, mi riallineo (merge/rebase) prima di continuare