GITFLOW E CLEAN CODE
COME ABBIMO RISOLLEVATO UN PROGETTO?

Bacaro Tech

CODE AND FUN

CIAO
CI PRESENTIAMO

Giorgio Basile
FE Developer

Michele Scarpa
Full stack developer

...VI PRESENTO BACARO TECH

Bacaro Tech è un'inizitiva che ha il compito di ricreare quell’atmosfera gioiosa e di gruppo, tipica del bacaro veneziano, nel mondo dell’informatica, attraverso la divulgazione sui social, eventi e workshop

Bacaro
Tech

CODE AND FUN

Bacaro
Tech

CODE AND FUN

CHE COS'È
BACARO TECH

PAGINA INSTAGRAM

ORA INIZIAMO CON LA STORIA (DELL'ORRORE)

Bacaro
Tech

CODE AND FUN

ATTENZIONE
ALZO LE MANI

Questa potrebbe non essere necessariamente la strada migliore per affrontare un progetto, ma il motivo per cui sono qui davanti a voi è per offrirvi un metodo di gestione e sapere anche la vostra opinione o cosa avreste fatto.

Bacaro
Tech

CODE AND FUN

VI RACCONTO UNA STORIA FORSE GIÀ SENTITA

- Cambio di progetto
- Forte cambio di dominio
- 6 persone che non avevano mai lavorato insieme
- Progetto senza documentazione
- Codebase confusa
- Nessuna guideline
- Gestione dei branch fallace

Bacaro
Tech

CODE AND FUN

IDEA “DAL BASSO”
MIGLIORARE LA CODEBASE

Bacaro
Tech

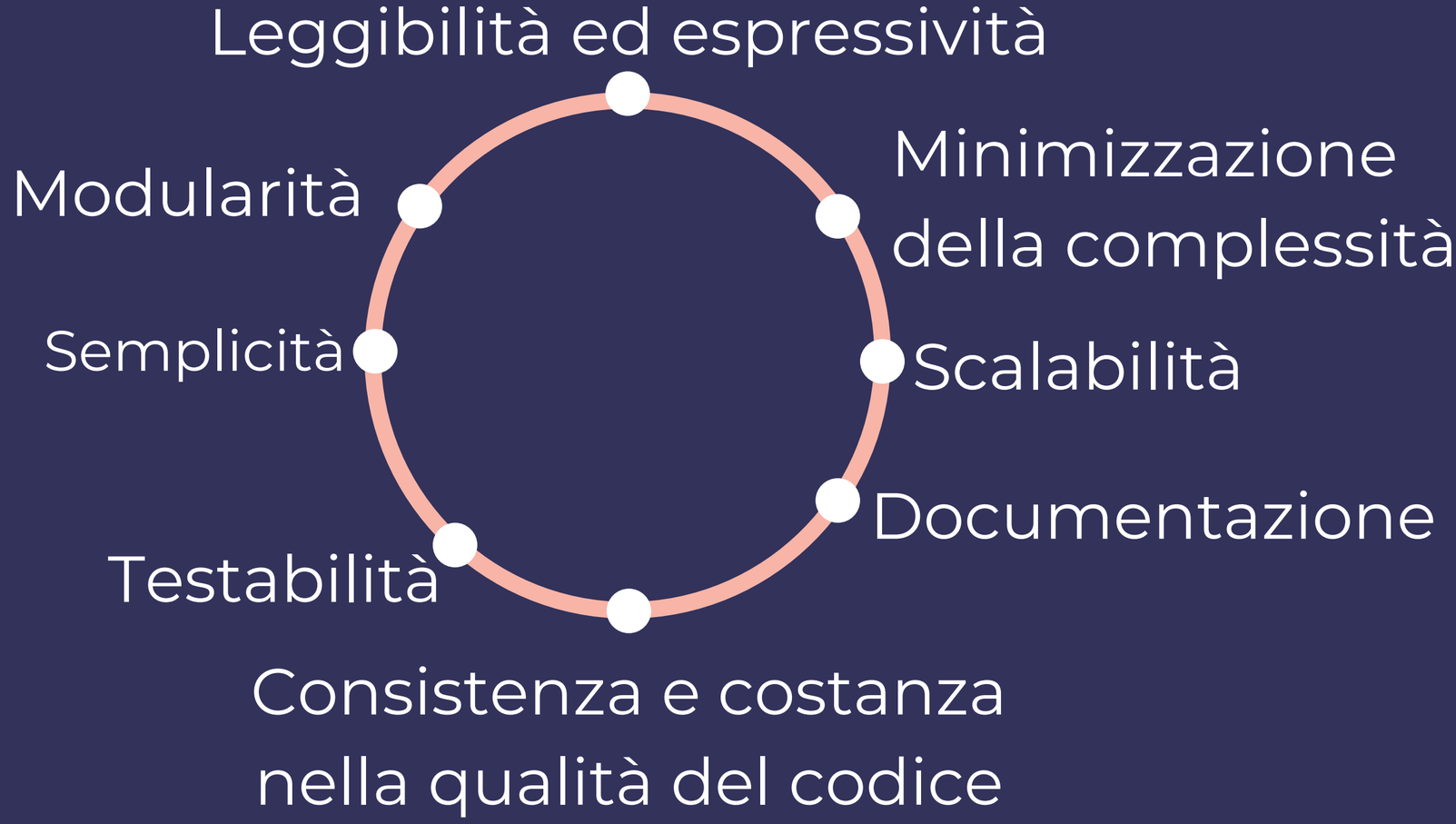
CODE AND FUN

PRINCIPI DEL CLEAN CODE
QUALI SONO?(ACCORPATI)

1. Leggibilità ed espressività
2. Semplicità
3. Consistenza e costanza nella qualità del codice
4. Minimizzazione della complessità
5. Testabilità
6. Modularità
7. Scalabilità
8. Documentazione

PRINCIPI DEL CLEAN CODE

CERCHIO DELLA VERITÀ



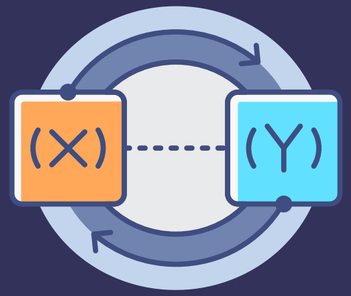
LEGGIBILITÀ ED ESPRESSIVITÀ

CHE COS'È?

Il **codice** deve essere scritto in modo che sia **facile da leggere e da comprendere per altri programmatori**.

Ciò significa utilizzare nomi significativi per variabili, funzioni e classi, evitare l'ambiguità e mantenere una struttura logica.

LEGGIBILITÀ ED ESPRESSIVITÀ
COME MIGLIORARLA?



Nome variabili

$$f(x)$$

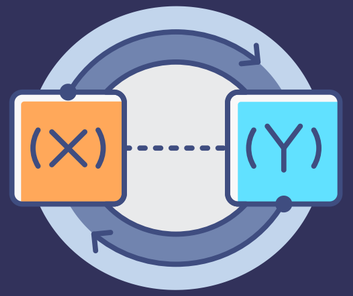
Nome funzioni

Codice strutturato

Strutture dati

Evitare duplicato

LEGGIBILITÀ ED ESPRESSIVITÀ
TEAM E PROGETTO

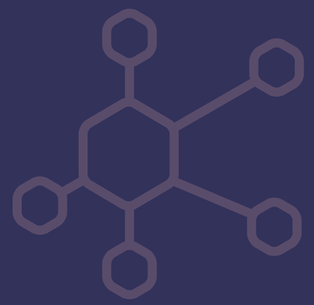


Nome variabili

$$f(x)$$

Nome funzioni

Codice strutturato



Struttur e dati

Evitare duplicato

SEMPLICITÀ
CHE COS'È?

Il codice deve seguire il principio **Keep It Simple, Stupid,** rendendolo il **più semplice possibile** senza sacrificare la funzionalità.

Evitare l'eccessiva complessità e l'implementazione di funzionalità non necessarie.

SEMPLICITÀ
SPAGHETTI CODE

Codice disorganizzato e difficile da seguire a causa della sua struttura intricata e annidata.

Cause: nomi poco significativi, mancanza di struttura, ecc...

CONSISTENZA E QUALITÀ
CHE COS'È?

Consiste nel mantenere uno **stile di scrittura costante in tutto il progetto**, come per esempio l'uso coerente di:

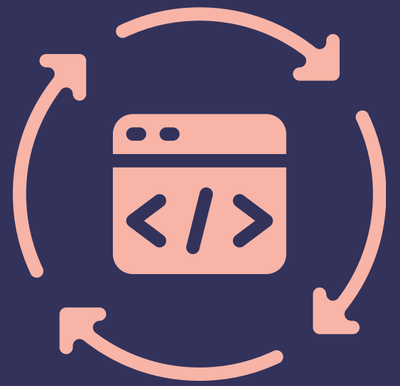
- spazi bianchi
- formattazione del codice
- nomenclatura delle variabili

e molto altro ancora...

QUALITÀ
COME MANTENERLA?

Code Review

Pair programming



Refactoring

Bacaro
Tech

CODE AND FUN

REFACTORING
SITO UTILE!

refactoring.guru

Bacaro Tech

CODE AND FUN

MINIMA COMPLESSITÀ CHE COS'È?

Evitare la **duplicazione di codice**, **ridurre al minimo le dipendenze tra i moduli**(*) e mantenere **ogni componente** del sistema il **più semplice** possibile.

Anche perchè chi è **responsabile dei moduli** esterni che inseriamo nella nostra code base?

Bacaro
Tech

CODE AND FUN

MINIMA COMPLESSITÀ
PRINCIPIO SOLID 1

Single Responsibility Principle

Ogni classe o funzione dovrebbe avere una sola responsabilità.

Open/Closed Principle

Il codice dovrebbe essere aperto per estensioni, ma chiuso per modifiche.

Bacaro Tech

CODE AND FUN

MINIMA COMPLESSITÀ
PRINCIPIO SOLID 2

Liskov Substitution Principle

Le classi derivate dovrebbero poter sostituire le classi base senza alterare il comportamento del programma.

Interface Segregation Principle

Le interfacce dovrebbero essere specifiche e non generiche.

Bacaro
Tech

CODE AND FUN

MINIMA COMPLESSITÀ
PRINCIPIO SOLID 3

Dependency Inversion Principle

Dipendere da astrazioni, non da classi concrete

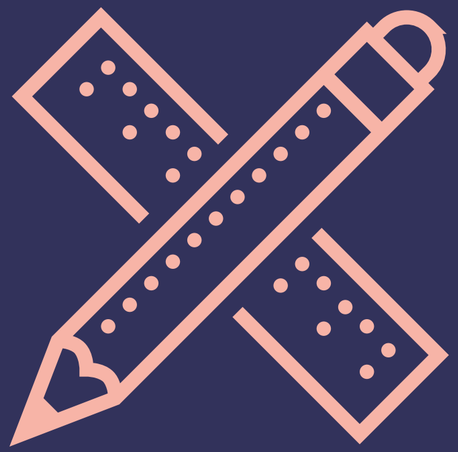
- Poliformismo
- Riusabilità++

MODULARITÀ
CHE COS'È?

Consiste nel **suddividere il codice in moduli separati,** ciascuno dei quali si occupa di una singola responsabilità.

Questo facilita la manutenzione del codice e favorisce il riuso nel lungo periodo.

MODULARITÀ
COME APPLICARLA?



Sfruttare design pattern

Studiare la struttura delle librerie

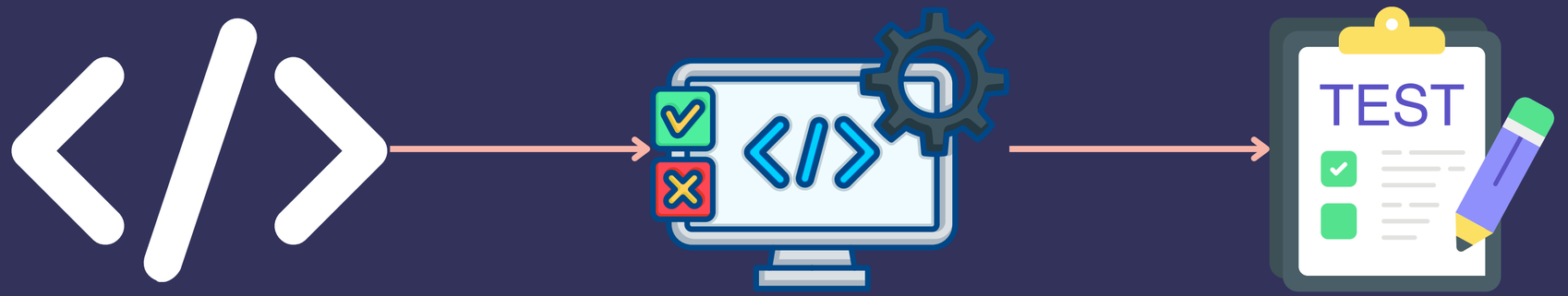
Best practice folder by framework

TESTABILITÀ
CHE COS'È?

Scrivere codice che sia **facile da testare da chiunque.**

Questo significa progettare il codice in modo che sia possibile scrivere test automatici per verificarne il corretto funzionamento.

TESTABILITÀ A LIVELLI



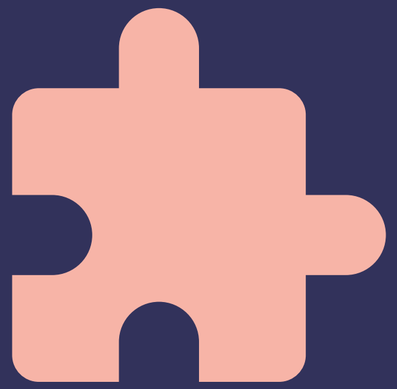
Codice → Unit Test → Test Manuali

- Check delle dipendenze
- Evitare lo stato condiviso
- Modularità e struttura

SCALABILITÀ
CHE COS'È?

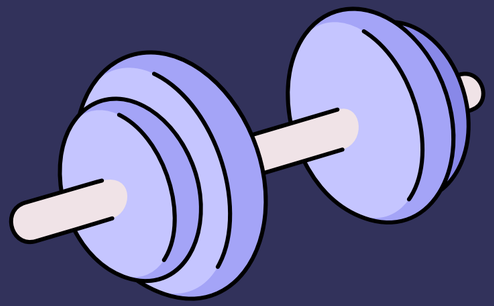
Scrivere il codice in modo che sia in **grado di gestire un aumento della complessità** o del carico senza richiedere una riscrittura completa.

SCALABILITÀ
COME GARANTIRLA?



Progettazione a moduli

Separazione delle responsabilità



Bilanciamento del carico

Ottimizzazione delle performance

RESPONSABILITÀ
COME SEPARARLE?

CONTROLLER

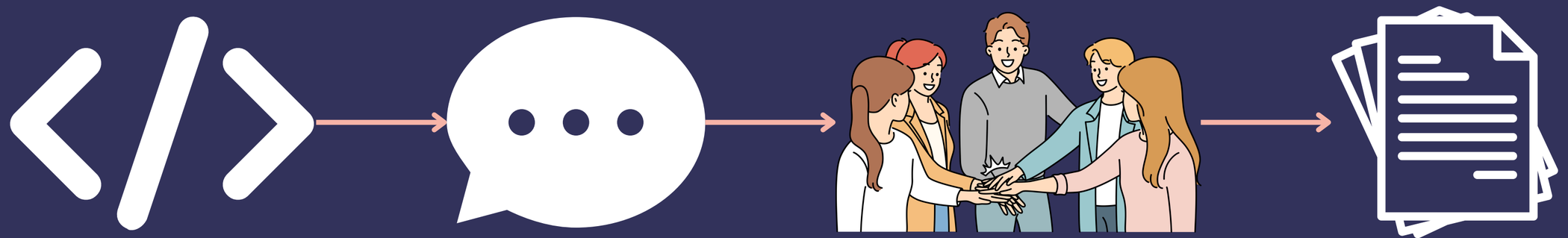
VIEW MODEL

DOCUMENTAZIONE
CHE COS'È?

Scrivere commenti chiari e significativi per spiegare parti complesse del codice o decisioni di progettazione **non ovvie**.

Tuttavia, il codice dovrebbe essere preferibilmente auto esplicativo, limitando il bisogno di commenti e documentazione eccessivi.

DOCUMENTAZIONE A LIVELLI



Codice → Commenti → Colleghi → Doc

Come faccio a far “parlare il codice”?
=> applicare i principi clean code!

TUTTO BELLO MA...
UN ALTRO PROBLEMA

Nonostante avessimo migliorato, il possibile, la codebase e la documentazione annessa, comunque la gestione dei branch continuava ad essere non ottimizzata.

come ci dovevamo sentire

Vs

come ci sentivamo

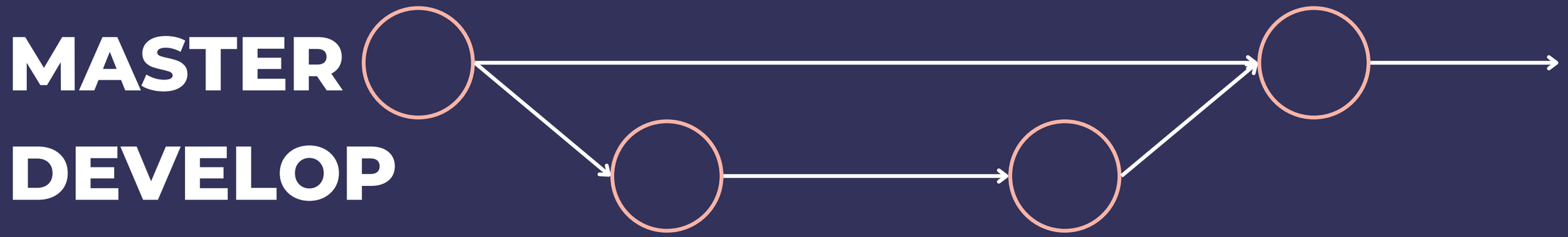
Bacaro
Tech

CODE AND FUN

IDEA “DAL BASSO”
MIGLIORARE I BRANCH

GIT FLOW
CHE COS'È?

Ideato da Vincent Driessen, Git Flow è un **modello di branching e workflow per il sistema di controllo della versione Git**



Bacaro
Tech

CODE AND FUN

GIT FLOW

QUALI SONO I BRANCH?

Branch principale (master): Il branch principale, spesso chiamato "master," rappresenta il codice in produzione. Questo è il ramo stabile

Branch di sviluppo (develop): Il branch di sviluppo è utilizzato per l'integrazione continua delle funzionalità.

Bacaro
Tech

CODE AND FUN

GIT FLOW
QUALI SONO I BRANCH?

Feature branches: Per sviluppare nuove funzionalità o correzioni di bug, vengono creati branch specifici chiamati “feature/” o “fix/”

Release branches: Quando è giunto il momento di preparare una nuova versione del software, viene creato un branch di rilascio.

Bacaro
Tech

CODE AND FUN

GIT FLOW
QUALI SONO I BRANCH?

Hotfix branches: Questi branch vengono utilizzati per risolvere i problemi di produzione in modo rapido e immediato, e le correzioni vengono quindi integrate sia nel branch principale che nel branch di sviluppo.

Bacaro
Tech

CODE AND FUN

GIT FLOW
WORKFLOW DEVELOP

Il workflow legato allo sviluppo di codice prevede:

1. **Stacco un branch master** che verrà battezzato **develop**.
2. **Stacco un branch da develop**, con la naming convention **“feature/...”**, per lo sviluppo di una nuova feature.
3. **Faccio il merge della feature in develop** dopo averla accuratamente testata.
4. Eseguo punto 2 e 3 per eventuali fix, ma la naming convention questa volta è **“fix/...”.**

Bacaro
Tech

CODE AND FUN

GIT FLOW
WORKFLOW RELESE

Il workflow legato ai rilasci prevede:

1. **Da develop viene staccato un branch relese**
2. Viene **stabilizzato il codice** che si trova su branch relese e in caso vengano scovati dei bug, questo branch oltre a essere mergiato su master che in develop
3. Viene **rilasciata l'applicazione nell'ambiente di produzione**
4. In caso ci fossero dei bug bloccanti in prod, **allora si aprira' un branch hotfix** per sistemare questo problema.

Bacaro
Tech

CODE AND FUN

GIT
LISTA DI COMANDI?

GIT - LA GUIDA TASCABILE

GIT FLOW
ESEMPIO

HOTFIX

RELESE

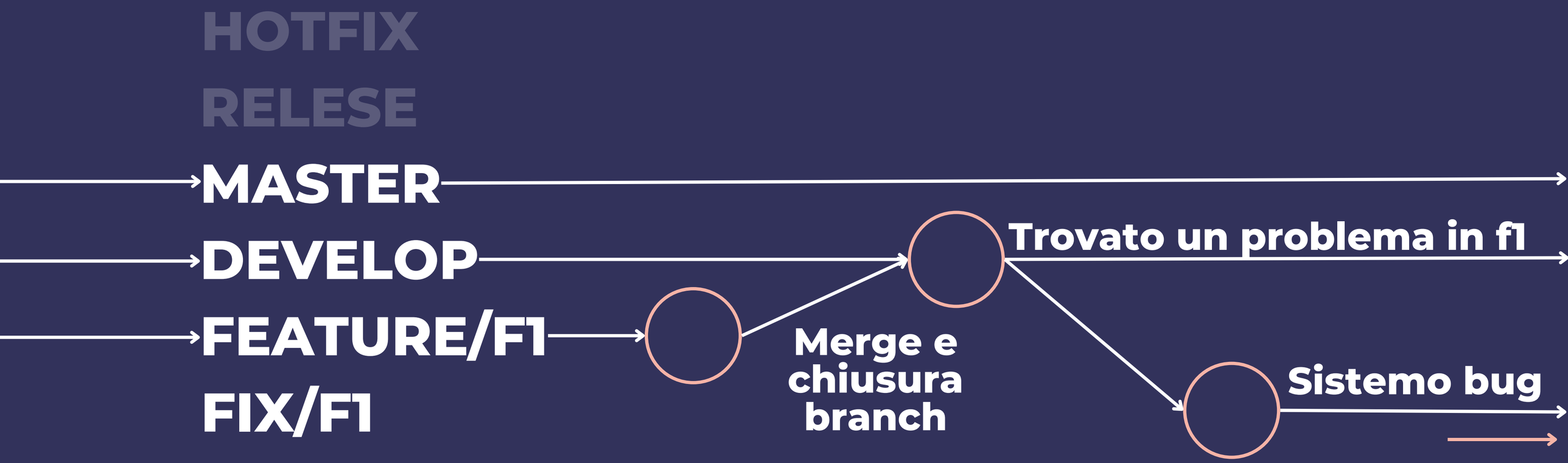
MASTER

DEVELOP

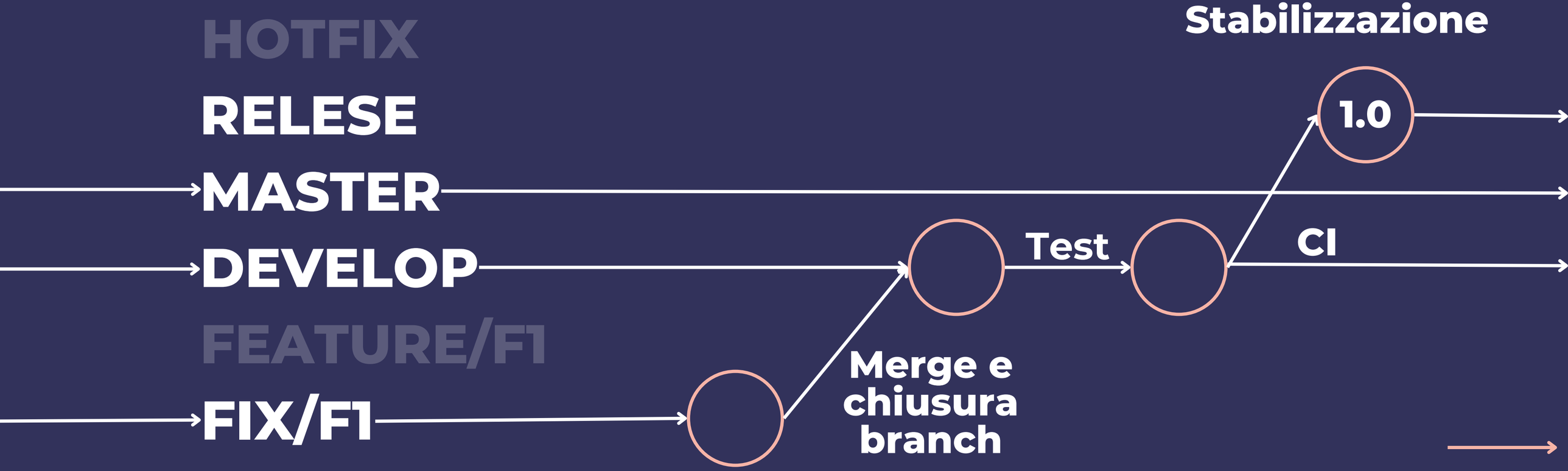
FEATURE/F1

FIX/F1

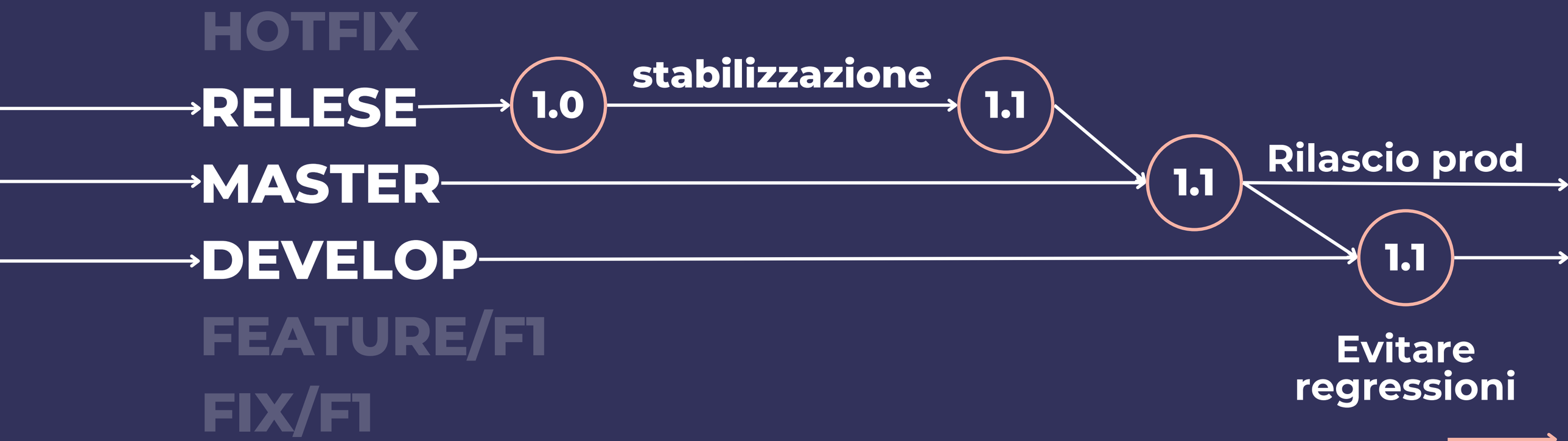
GIT FLOW
ESEMPIO



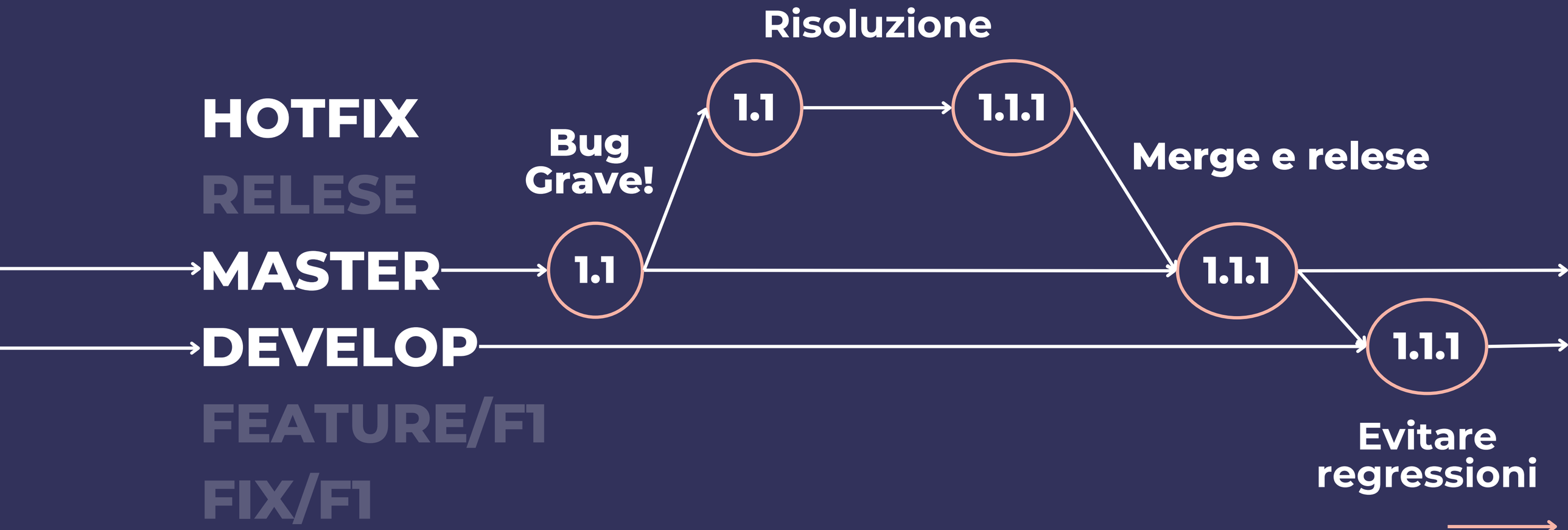
GIT FLOW
ESEMPIO



GIT FLOW
ESEMPIO

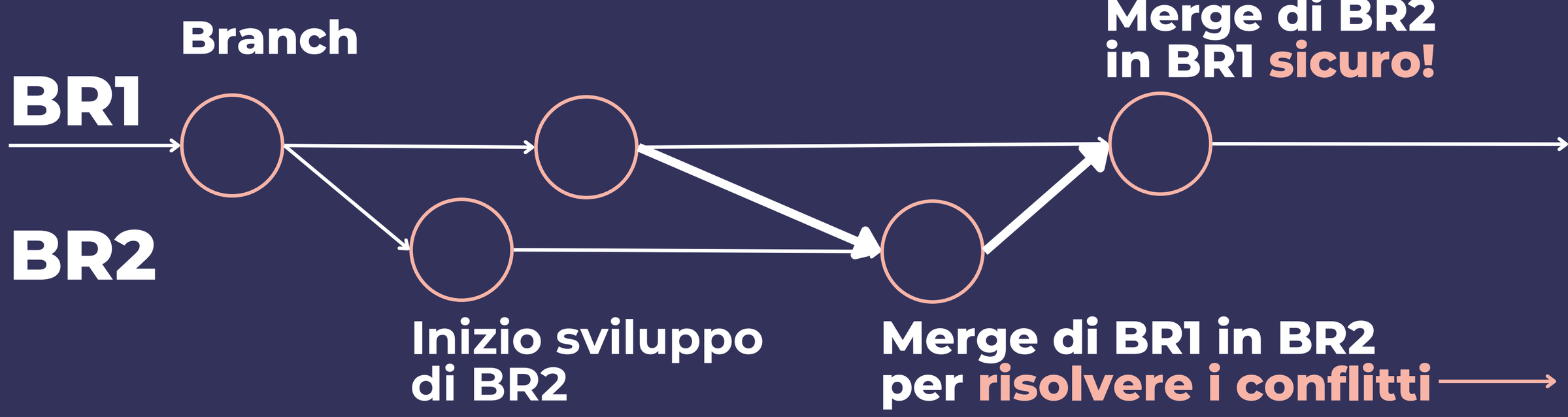


GIT FLOW
ESEMPIO



GIT FLOW
GESTIONE DEI MERGE

Una cosa importante da sottolineare è l'importanza di risolvere i conflitti nel proprio branch feature, e non nel branch che dovrà mergiare il nostro branch



SFIDE SOSTENUTE DA NOI DEVELOPER

- Curva di apprendimento maggiore
- Resistenza del managment
- Gestione del debito tecnico
- Documentazione e comunicazione
- Compromessi nella progettazione
- Pressione sulle scadenze

RISULTATI OTTENUTI
AL NETTO DELLA MIGRAZIONE

- Migliore leggibilità
- Maggiore produttività
- Riduzione delle future regressioni
- Riduzione del tempo di debug
- Aumento generale della qualità del codice
- Aumento generale del devops

Bacaro
Tech

CODE AND FUN

VI RINGRAZIA TUTTI PER AVER PARTECIPATO!

ASPETTIAMO VOSTRE DOMANDE

Bacaro
Tech

CODE AND FUN

BACARO TECH
SEGUITECI!

PAGINA INSTAGRAM