**Workflow-Dokument: Gute GitBash-Routine beim Start der Arbeit**

**O -> wichtiger Schritt bzw. Schritt der meistens verwendet wird**

**O -> unwichtig für uns ZURZEIT**

**Vorbereitungen: Initiale Einrichtung**

1. **Git-Repository klonen (falls nicht bereits vorhanden):**

bash

git clone <REPOSITORY\_URL>

cd <PROJEKTORDNER>

1. **Überprüfen der aktuellen Branches:**

bash

git branch -a

Stelle sicher, dass du dich auf dem richtigen Branch befindest, z. B. main oder develop.

**Tägliche Routine beim Start der Arbeit**

**1. Repository-Status prüfen**

Bevor du mit der Arbeit beginnst, überprüfe den Zustand deines lokalen Repositorys:

bash

git status

* **Untracked Files:** Zeigt Dateien, die nicht im Repository sind.
* **Staged Changes:** Änderungen, die für den nächsten Commit bereit sind.
* **Unstaged Changes:** Änderungen, die noch nicht zum Commit hinzugefügt wurden.

**2. Sicherstellen, dass der Master-Branch aktuell ist**

Um sicherzustellen, dass dein lokaler Master-Branch (oder main) auf dem neuesten Stand ist:

bash

git checkout main

git pull origin main

* **Hinweis:** Falls euer Team einen develop-Branch verwendet, passe den Branch-Namen entsprechend an:

bash

git checkout develop

git pull origin develop

**3. Erstellen eines neuen Feature-Branches**

Für jede Aufgabe oder Änderung, erstelle einen separaten Branch:

bash

git checkout -b <feature-branch-name>

Beispiele:

* Für eine neue Funktion: feature/add-login-system
* Für einen Bugfix: bugfix/fix-authentication-error

**4. Arbeiten an deinem Feature**

Während der Arbeit:

* **Änderungen überprüfen:**

bash

git status

* **Änderungen zum Staging-Bereich hinzufügen:**

bash

git add <DATEI> # Für eine spezifische Datei

*oder:*

git add . # Für alle geänderten Dateien

* **Commit erstellen:**

bash

git commit -m "Beschreibung der Änderungen"

**5. Branch regelmäßig synchronisieren**

Während der Arbeit solltest du deinen Feature-Branch regelmäßig mit dem Master-Branch (oder develop) synchronisieren:

1. **Wechsle zum Master-Branch und aktualisiere ihn:**

bash

git checkout main

git pull origin main

1. **Wechsle zurück zu deinem Feature-Branch:**

bash

git checkout <feature-branch-name>

1. **Rebase deinen Feature-Branch auf den Master-Branch:**

bash

git rebase main

* + **Konflikte lösen (falls vorhanden):**
    - Öffne die betroffenen Dateien, löse die Konflikte und füge sie wieder hinzu:

bash

git add <DATEI>

* + - Setze das Rebase fort:

bash

git rebase --continue

**Zum Ende der Arbeit**

**1. Änderungen pushen**

Wenn du deine Arbeit abgeschlossen hast, pushe deinen Feature-Branch ins Remote-Repository:

bash

git push origin <feature-branch-name>

**2. Pull Request (PR) erstellen**

* Navigiere zur Plattform (z. B. GitHub, GitLab) und erstelle einen Pull Request, um deinen Branch in den Hauptbranch (z. B. main oder develop) zu mergen.
* Beschreibe die Änderungen klar und weise ggf. Reviewer zu.

**Zusammenfassung: Die tägliche Routine**

1. **Repository aktualisieren:**

bash

git checkout main

git pull origin main

1. **Feature-Branch erstellen:**

bash

git checkout -b <feature-branch-name>

1. **Änderungen machen, committen und regelmäßig synchronisieren:**
   * Arbeiten speichern:

bash

git add .

git commit -m "Beschreibung der Änderungen"

* + Branch aktuell halten:

bash

git rebase main

1. **Feature-Branch pushen und PR erstellen:**

bash

git push origin <feature-branch-name>