Nachdenkzettel Git

Aufgabe 1:

Ein Grund um ein VCS zu verwenden ist zum Beispiel die Versionierung. Man kann den Zustand des Codes zu bestimmten Zeitpunkten speichern und notfalls darauf zurückgreifen, sollten Fehler passieren.
Außerdem sind VCS nützlich im Bezug auf Änderungen. Man kann überwachen, wann und warum bestimmte Änderungen vorgenommen wurden oder sie Rückgängig machen.
VCS ermöglichen auch das Erstellen von Branches und das Merging dieser.
Mithilfe von VCS können außerdem mehrere Entwickler gleichzeitig an einem Projekt arbeiten und ihre jeweiligen Änderungen in den gemeinsamen Code einfügen.

Aufgabe 2:

Im Allgemeinen sollten in einem Software-Repository (.java, .xml, .json und Konfigurations-) Dateien enthalten sein, die für die Entwicklung und den Betrieb der Anwendung erforderlich sind. Bilddateien und Musikdateien, die im Projekt benötigt werden, sollten normalerweise vermieden werden, um die Repository-Größe nicht unnötig zu erhöhen.
UML-Modelle, Notizen und Dokumentationen sind wichtig für die Kommunikation im Team und sollten im Repository enthalten sein.
Vertrauliche Daten wie Passwörter sollten an sicheren Orten gespeichert werden.
Auch Abschlussarbeiten sollten nicht im repo enthalten sein, da sie für das Projekt irrelevant sind.
Log-Files gehören normalerweise nicht in ein Repository, da sie oft generiert werden.

Aufgabe 3:

Der Ablauf zur Arbeit mit einem Git-Repository besteht aus mehren Schritten.

1. Hinzufügen zur Staging-Area:
   Nachdem Änderungen an Dateien gemacht wurden, verwendet man den Befehl 'git add' , um diese Änderungen zur sogenannten Staging-Area hinzuzufügen.
2. Commit
   Nachdem Änderungen zur Staging-Area hinzugefügt wurden, erfolgt das Übernehmen in das lokale Repository mit dem Befehl 'git comit'. Jeder Commit sollte eine bestimmte Änderung darstellen und eine Commit-Nachricht haben.
3. Push
4. Nach lokalen Commits können die Änderungen auf ein remote Repository gepusht werden. Dies ist besonders wichtig, wenn man mit anderen Entwicklern zusammenarbeitet.
5. Pull
   Wenn andere Entwickler Änderungen am Remote-Repository vorgenommen haben, können diese mit 'git pull' heruntergeladen und in das lokale Repository integriert werden.

Aufgabe 4:

Bei einem merge-conflict konnten 2 Branches nicht automatosch zusammengeführt werden. Um sie zu beheben muss man sie manuell auflösen.
Die Konflikte werden markiert und es muss entschieden werden, welche Änderung behalten werden soll. Nachdem man den Konflikt als behoben markiert hat, kann man ihn commiten.

Aufgabe 5:

Nach jeder Änderung wird diese committed und gepusht. Dabei werden die Änderungen kurz aber aussagekräftig in den commit-Messages beschrieben.