1 Arbeitsauftrag Versionsverwaltung

Versionsverwaltung ist ein wesentlicher Bestandteil der Softwareentwicklung, der dazu dient, Änderungen in einem Projekt systematisch zu verfolgen, zu organisieren und zu dokumentieren. In der sich ständig weiterentwickelnden Welt der Software ist es unerlässlich, den Überblick über unterschiedliche Entwicklungsstände, Fehlerbehebungen und neue Funktionen zu behalten. Genau hier kommt die Versionsverwaltung ins Spiel.

Versionsverwaltungssysteme ermöglichen es Entwicklerteams, den Verlauf ihres Projekts effizient zu managen. Sie bieten Werkzeuge zur Verfolgung von Änderungen an Quellcode, Dokumentation und anderen Ressourcen, sodass Teammitglieder nahtlos zusammenarbeiten können, ohne dabei die Kontrolle über den Entwicklungsprozess zu verlieren.

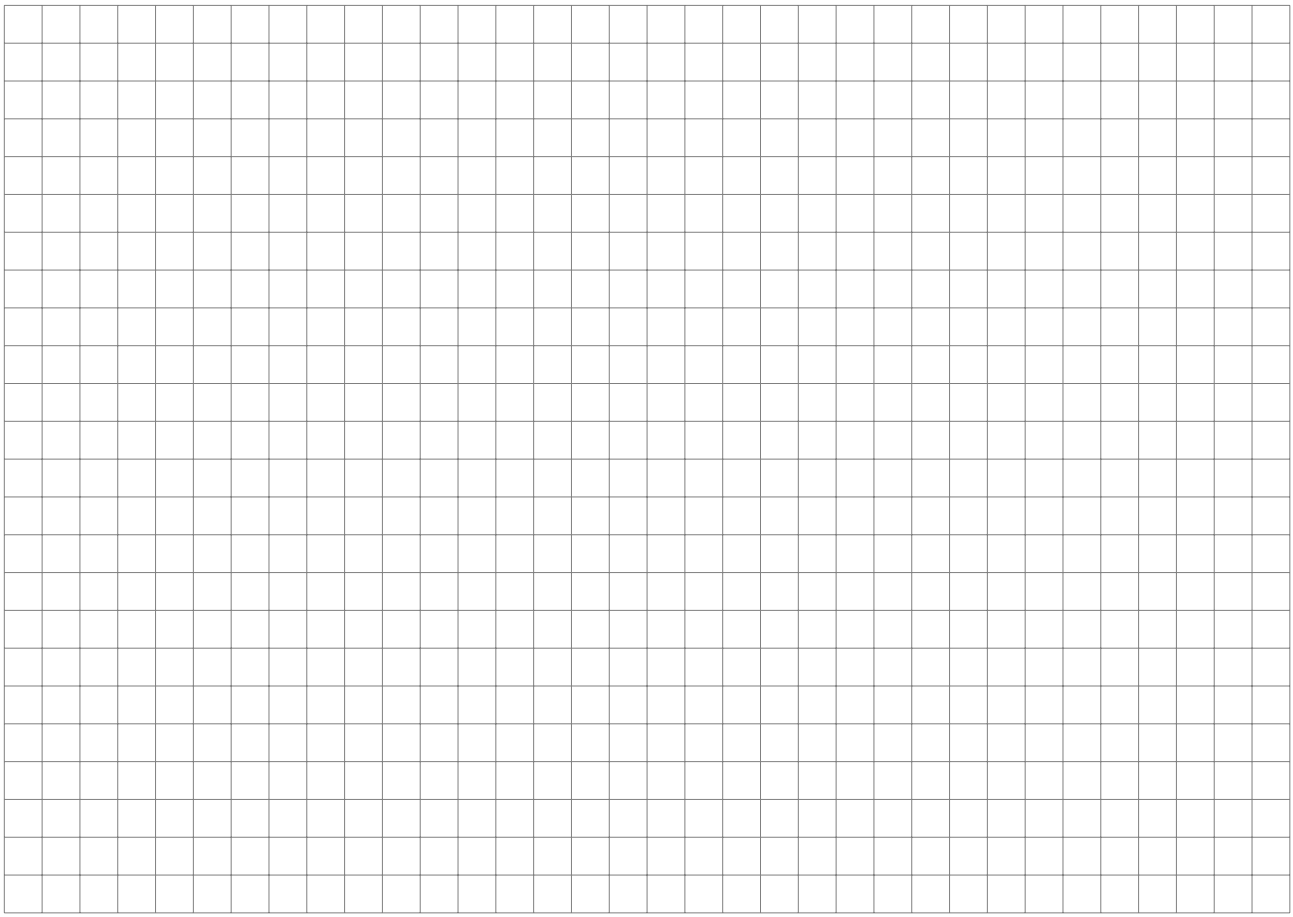
Die grundlegende Idee hinter der Versionsverwaltung besteht darin, jeden Schritt im Entwicklungszyklus festzuhalten, sei es das Hinzufügen neuer Funktionen, das Beheben von Fehlern oder das Anpassen von bestehendem Code. Dies ermöglicht nicht nur die Rückverfolgung von Änderungen, sondern auch das einfache Wiederherstellen früherer Zustände, falls Probleme auftreten oder bestimmte Versionen für spezifische Anforderungen benötigt werden.

> **Arten von Versionsverwaltungssystemen**
>
> Es gibt verschiedene Arten von Versionsverwaltungssystemen, darunter zentrale Systeme wie SVN (Subversion) und dezentrale Systeme wie Git. Beide Ansätze haben ihre Vor- und Nachteile, aber sie alle dienen dem gemeinsamen Ziel, die Entwicklung von Softwareprojekten transparenter, effizienter und fehlerfreier zu gestalten.

1.1 Aufgaben

Versionsverwaltung | Partnerarbeit | 20’

1. Notieren Sie sich unterschiedliche Funktionen einer Versionsverwaltung. Welche Hauptaufgaben übernimmt eine Versionsverwaltung? Beschreiben Sie jedes Stichwort kurz und versuchen Sie dazu Beispiele zu finden.
2. Welche Arten der Versionsverwaltung gibt es grundsätzlich?
3. Welche Versionsverwaltung nutzen Sie in Ihrem Betrieb?



1.2 Workshop Git

> Vertiefungsauftrag
>
> Falls Sie Git bereits kennen und vertraut mit der Arbeitsweise sind, arbeiten Sie den Workshop trotzdem durch. Versuchen Sie dazu ein Protokoll mit den entsprechenden Befehlen zu erstellen, die Sie anschliessend als Musterlösung Ihren Klassenkolleginnen und -kollegen abgeben können.

1.2.1 Arbeitsauftrag Git - Teil 1

Bearbeiten Sie diese Aufgabe alleine. Erstellen Sie die geforderten **Screenshots** und laden Sie diese am Ende in OneNote hoch. Hinweis: In OneNote finden Sie Unterlagen zu Git, die Ihnen bei der Bearbeitung weiterhelfen.

1.2.1.1 Aufgabe 1

a) Installieren Sie Git und überprüfen Sie mit `git --version`, ob Git korrekt installiert wurde.
b) Konfigurieren Sie Git und setzen Sie den Namen (inkl. E-Mail), unter der Sie zukünftig arbeiten werden.
c) Erstellen Sie einen leeren Ordner «Übung1» und wechseln Sie mit der Kommandozeile in diesen Ordner.
d) Rufen Sie `git init` auf, um den Ordner in ein Git-Repository umzuwandeln. Hinweis: Dabei wird ein versteckter Ordner namens `.git` erstellt. Darin werden die für Git relevanten Informationen gespeichert. Falls Sie den Ordner wieder aus dem Versionsverwaltungssystem Git entfernen möchten (oder zurücksetzen), können Sie den versteckten Ordner löschen.

1.2.1.2 Aufgabe 2

e) Erstellen Sie im Übungsordner eine Markdown-Datei «test.md» und fügen Sie darin etwas Inhalt (mit Markdown-Syntax) ein. Informationen zu Markdown finden Sie unter: https://de.wikipedia.org/wiki/Markdown
f) Überprüfen Sie den aktuellen Status Ihrer Arbeitskopie mit `git status`.
g) Fügen Sie die Datei «test.md» mit `git add` zum Index hinzu und überprüfen Sie den Status erneut. **[1. Screenshot]**
h) Erstellen Sie einen Snapshot, basierend auf dem aktuellen Index mittels `git commit` und überprüfen Sie den Status Ihrer Arbeitskopie.
i) Lassen Sie sich eine übersichtliche Darstellung der Versionshistorie darstellen. **[2. Screenshot]**

1.2.1.3 Aufgabe 3

j) Machen Sie eine Änderung an «test.md», überprüfen Sie den Status, fügen Sie die Änderung dem Index hinzu und erstellen Sie einen Commit. Tipp: Achten Sie immer darauf, dass die Commit-Nachricht Ihre Änderung kurz und treffend beschreibt.
k) Erstellen Sie einen Branch «experiment» und wechseln Sie zu diesem Branch, um darin an einer Idee zu arbeiten.
l) Erstellen Sie eine neue Datei «name.md» und fügen Sie darin Ihren Namen ein.
m) Erstellen Sie den Commit und speichern Sie den Zustand der neuen Datei in Ihrem Repository (experiment).
n) Lassen Sie sich eine übersichtliche Darstellung der Versionshistorie darstellen. **[3. Screenshot]**

1.2.1.4 Aufgabe 4

Sie möchten die Änderungen vom Branch «experiment» in den master-Branch übernehmen. **[4. Screenshot]**

1.2.1.5 Aufgabe 5

o) Machen Sie eine Änderung (Anpassung in der «test.md») im master Branch und committen Sie diese.
p) Da Sie nun etwas im Branch «experiment» ausprobieren möchten, wechseln Sie zum Branch «experiment» und mergen den master in experiment, um an der aktuellen Version weiterzuarbeiten.
q) Fügen Sie bei «name.md» Ihre Adresse hinzu, committen diese und mergen diese Änderung zurück in den master. **[5. Screenshot]**

1.2.1.6 Aufgabe 6

Finden Sie heraus was eine `.gitignore` Datei macht und wo sinnvolle Einsatzzwecke davon liegen.

1.2.2 Arbeitsauftrag Git - Teil 2

Bearbeiten Sie diese Aufgabe in Zweiergruppen. Erstellen Sie die geforderten Screenshots und laden Sie diese am Ende in OneNote hoch. Hinweis: In OneNote finden Sie Unterlagen zu Git, die Ihnen bei der Bearbeitung weiterhelfen.

> **Caution**
>
> Damit Sie in der Arbeit mit GitHub nicht ständig ein Passwort eingeben müssen, können Sie GitHub mit SSH-Keys bedienen. Dazu erstellen Sie sich ein Schlüsselpaar und kopieren den public key zu GitHub.
> Mehr Informationen dazu finden Sie unter: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

1.2.2.1 Aufgabe 7

Sie wollen im Team an Softwareprojekten arbeiten.

r) Jemand aus der Zweiergruppe soll in Github ein Übungsrepository anlegen und die andere Person mit Lese- und Schreibrechte ausstatten.
s) Klonen Sie das erstellte Repository, um an einer Kopie auf dem eigenen Rechner arbeiten zu können.
t) Überprüfen Sie mit `git remote -v`, welche Remote-Repositories mit Ihrem lokalen Repository verknüpft sind. **[6. Screenshot]**

1.2.2.2 Aufgabe 8

Bisher ist das Repository leer. Arbeiten Sie nun unabhängig voneinander und erstellen Dateien und Inhalte.

u) Committen Sie Ihre Änderungen.
v) Teilen Sie Ihre Änderung mit der anderen Person, indem Sie diese «pushen». (Falls der andere schneller war, resultiert darin ein Fehler) **[7. Screenshot]**
w) Ziehen Sie sich die neusten Änderunge mittels `git fetch`. Falls Sie noch lokale Committs haben (`git branch -vv` zeigt «ahead» an), dann mergen Sie die Änderungen Ihres Kollegen in Ihren master branch.

1.2.2.3 Aufgabe 9

Nach einiger Zeit ist Ihnen im Team aufgefallen, dass manchmal Code committed wird, der gar nicht funktioniert. Dies stört alle anderen bei ihrer Arbeit. Deshalb einigen Sie sich darauf, Code nur noch in master zu mergen, wenn dieser mindestens von einer anderen Person kontrolliert wurde.

x) Erstellen Sie einen neuen lokalen Branch und machen Sie darin min. 1 Änderung, die Sie committen **[8. Screenshot]**
y) Veröffentlichen Sie den Branch im Remote Repository
z) Verwenden Sie die Github Weboberfläche, um einen Pull Request zu erstellen. **[9. Screenshot]**. Laden Sie Ihren Teamkollegen durch Erwähnung seines Benutzernamens ein, Feedback zum Pull Request zu geben. Wenn der Kollege einverstanden ist: Mergen Sie den Branch und löschen ihn danach **[10. Screenshot]**

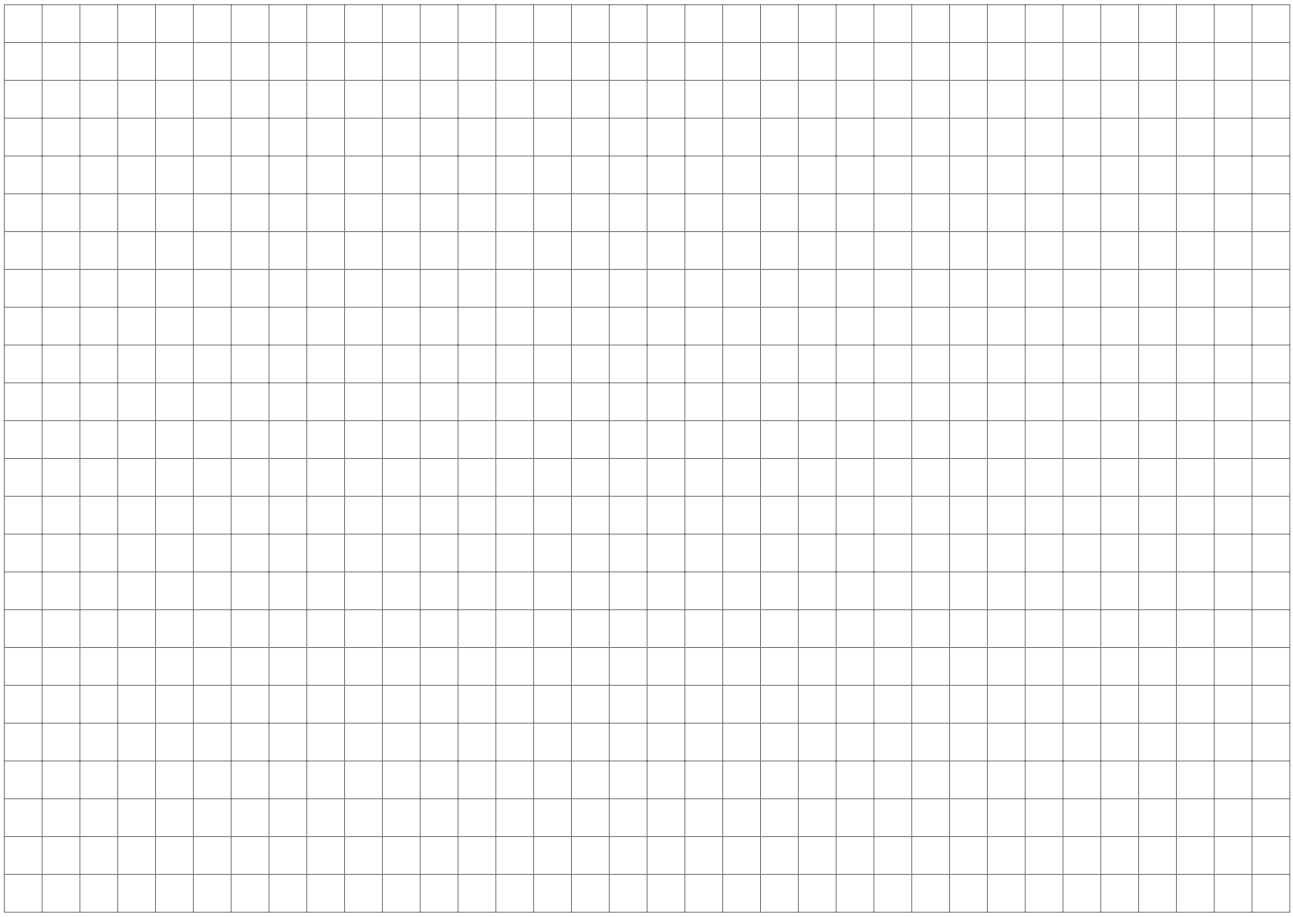
2 Tagging

Um vor lauter Commits die Übersicht nicht zu verlieren, macht es Sinn, gewisse Stände mit einem Tag zu versehen. Dieses Konzept der Tags kennen Sie auch aus anderen Anwendungen wie Blogs, aber auch Instagram etc. Dort wird häufig mit dem Zeichen # und eine Wort ein Tag gesetzt.

Dieses Konzept kann auch bei einer Versionsverwaltung angewendet werden. Dazu finden Sie unter https://git-scm.com/book/en/v2/Git-Basics-Tagging Informationen.

Tagging mit git | Einzelarbeit | 30'

1. Studieren Sie die offizielle Dokumentation über Tags: https://git-scm.com/book/en/v2/Git-Basics-Tagging
2. Erstellen Sie dazu für sich eine Zusammenfassung.



Portfolio

Erstellen Sie ein Repository, in dem Sie Ihre Applikation und deren Änderungen verwalten. Fügen Sie die entsprechenden Projektmitglieder hinzu und setzen Sie Ihre Arbeitsumgebung sinnvoll auf. Dokumentieren Sie Ihr Vorgehen/Resultate und begründen Sie, warum Sie diese Vorgehensweise gewählt haben.