GIT/GITLAB

1. Was genau sind die Gründe um Gitlab zu verwenden?
Um beispielsweise gemeinsam an (Software) Projekten zu arbeiten, Zwischenschritte speichern und die finale Version davon ggf. dann zur Verfügung stellen. Änderungen von anderen Entwicklern können genau eingesehen werden und dadurch wird es einfacher, Bugs zu finden oder Änderungen nachzuvollziehen. Für Open-Source Projekte ist es auch gut geeignet, da die Nutzer den Code der Software und die jeweiligen Änderungen mitverfolgen können und gegebenenfalls auch ihr Wissen mit einbeziehen können um die Software zu verbessern.

2. Welche Daten gehören (nicht) ins Repo?

- .java Files
- .xml Files
- .Jason Files
- ~~Bilddateien für Gameprojekte~~
- ~~Musikdateien für Projekte~~
- .project Files (Endung je nach Tool)
- ~~UML Modelle~~
- ~~Zeichnungen~~
- ~~Notizen~~
- Dokumentation
- Configurationsfiles
- ~~Kapitel eines Buches~~
- ~~Eine Bachelorarbeit~~
- ~~Passwörter für Cloud-Services~~
- ~~Passwörter für lokale Services (Self-hosted)~~
- ~~logfiles~~
- ~~Messdaten vom Profiling~~

3. Was soll der Mist mit den Stages (dass add/commit nur lokal wirken)?
Dient zur eigenen Sicherheit, um Fehlern aus den Weg zu gehen, eine Art Schutzmechanismus.

4. Würden Sie in einer Firma Gitlab selber hosten oder GIT als Service im Netz? Begründung.
Hier kommt es sehr auf den Umfang des Projekts an. Für einmalige Versuche oder Entwicklungen im Rahmen eines Studiums reicht der Service im Internet völlig aus. Arbeitet man aber vermehrt an solchen Projekten und bezieht zudem noch Daten von Kunden, sollte man sich überlegen GIT selbst zu hosten. Vorzeige Beispiel ist hier die HDM Stuttgart.
Ansonsten empfehlen wir unter den obenen genannten Bedingungen selbst zu hosten, damit dann ggf. Daten von Kunden nicht an Dritte gelangen, da ist das Risiko zu hoch. Zudem hat man immer die volle Kontrolle über die gesamten Daten. Was allerdings gegen ein eigenständiges Hosten spricht, sind die Normen der Ausfallsicherheit. Benötigt mehrere Sicherungssysteme, ebenso hat man keinen Support wenn mal was ausfallen sollte.

5. Verwenden Sie Branches im Projekt oder arbeiten alle Teammitglieder auf dem Master Branch? Zeigen Sie Vor- und Nachteile der Verfahren

Alle Teammitglieder arbeiten auf der Master Branch.
Nur Master Branch:

Vorteile	Nachteile
- alles auf einen Blick erkennen	- kann bei komplexen/großen Projekten genau das Gegenteil bewirken

Verschiedene Branches:

Vorteile	Nachteile
- Bringt Struktur in das Projekt - Fehler können besser zurückverfolgt werden	- Evtl. sind einzelne Teams schneller/ in der Produktion, was ein Migrieren am Ende erschwert - Mehr Kommunikation ist erforderlich um ein Migrieren zu ermöglichen

6. Wie veröffentlichen Sie Ihre Änderungen auf dem globalen Repository? Wie oft checken Sie Ihre Änderungen im globalen Repo ein? Was ist besser: Nach jeder Änderung, nach einigen Änderungen, wenn Sie ein zusammenhängendes Stück fertig haben, wenn Sie eine Änderung machen die viele Kollegen betrifft, einmal am Tag, einmal die Woche. Wieso?
Zuerst wird committed, am besten mit Beschreibung, und dann wird gepusht.
Am besten pusht man den Code, wenn dieser auch funktioniert (ohne Syntaxfehler), also quasi ein funktionsabhängiger Push(ggf. “Auskommentier-Funktion” nutzen), sonst summiert sich die Anzahl der Fehler. Zudem sollte funktionierender Code nicht zu lange zurückgehalten werden, denn je mehr Zeit verstreicht, desto mehr Schwierigkeiten können bei einem verzögerten Push auftauchen.
Also kann man sagen das die Funktion an erster Stelle steht, jedoch die Beachtung der Zeit auch wichtig ist.

7. Eine Version eines Files im Repo sieht so aus: „There are two versions of GitLab: Community Edition (CE) and Enterprise Edition (EE)“ Ihre Version die Sie hochladen hat an dieser Stelle:
„There are two versions of GitLab: CE and EE.“
Wie löst Gitlab diesen Konflikt?
Es kommt zu einem Merge-Konflikt, den GitLab als Fehlermeldung anzeigen wird und außerdem die genauen Stellen, die sich überlappen. Es ist nun unsere Aufgabe als Developer, unseren Teil so zu bearbeiten, sodass er nicht mehr mit der bereits existierenden Version “crasht” und wir erneut versuchen können hochzuladen.

8. Hausaufgabe für jedes Team: Es muss eine chain-story (Kettengeschichte) erzählt werden bei der ALLE Teammitglieder jeweils immer einen Satz erfinden und dann einchecken. Ich will von ALLEN Teammitgliedern Sätze im Repo finden! Danach sagen Sie uns wie es ging. Haben Sie über externe Kanäle kommuniziert (Slack?). Anderweitig abgesprochen?

pk090 hat den Beginn gemacht
vv014 den ersten Satz der Kurzgeschichte
ss534 mit dem zweiten spannenden Satz erweitert
ss532 ergänzt die brisante Geschichte um eine weitere Zeile

Absprache über Whatsapp & Discord - zudem geupdatet vor dem eigenen pushen