Anwendungsszenarien Unterricht

Inhalt

1. SuS versionieren lokal

Einsatzbereich: IMP ab Klasse 8

Voraussetzungen

- kein Account bei einem GIT Hoster notwendig
- Stage, Commit, Anlegen eines Repositories, Check out

Oft haben SuS Probleme, ihre Dateien ordentlich zu verwalten. Oft existieren mehrere Versionen in verschiedenen Ordnern, die Arbeit eines Tages wird gar nicht gespeichert oder durch ungwollte Änderungen wird der Code nicht mehr ausführbar. Den SuS fällt es schwer, diesen Fehler zu korrigieren, da oft unbeabsichtig notwendiger Code gelöscht wurde, den die SuS nicht wiederherstellen können.

Durch die Verwendung von GIT von Beginn des Programmierunterrichts an, werden diese Probleme minimiert. Es muss zur Routine werden, am Ende jeder Arbeitsphase / Unterrichtsstunde einen Commit zu machen. Dabei wird die Arbeit des Tages rekapituliert und der aktuelle Arbeitsstand im Commit festgehalten.

Bei ungewollte Änderungen kann jederzeit zum Arbeitsstand des Vortages zurückgekehrt werden oder die Unterschiede zum Vortag angezeigt werden.

Die Sus initialisieren dazu einen Ordner als GIT-Repository. In diesem bearbeiten sie ihr Projekt und speichern die aktuelle Version am Ende jeder Stunde. Dazu werden alle Änderungen der Stunde mit einem entsprechenden Kommentar commited.

Solange die SuS nur in der Schule am Projekt arbeiten, ist es nicht erforderlich das Repository auf Git Camp zu pushen. Dadurch ist es unmöglich Versionskonflikte zu bekommen. Die SuS werden auf sehr einfache Art und Weise an GIT herangeführt.

2. Lehrperson stellt Material per GIT bereit, SuS versionieren lokal

Einsatzbereich: IMP ab Klasse 8

Voraussetzungen

- öffentlich verfügbares Repository oder schuleigene Git Camp Instanz mit Zugängen für die Schüler
- Clonen, Pull, Stage, Commit, Check out, (Rebase, Fetch)

Die Lehrperson stellt ein Repository online für Schüler zur Verfügung. Die Schüler clonen dieses Repository lokal in der Schule und arbeiten lokal. SuS machen am Ende der Stunde einen Sicherungsschritt (commit).

Dabei können die Unterschiede zur Vorversion sichtbar gemacht werden. Außerdem ist es möglich, zu einer Vorversion zurückzukehren. Dies entspricht den Möglichkeiten des ersten Anwendungsszenarios. Hier kommt im ersten Schritt ein Clonen eines vorhanden Repositorys hinzu.

Variante Update bei Fehler:

Gegebenenfalls kann die Lehrperson ein Update des Repository zur Verfügung stellen. Die Sus fetch/pullen die Neuerungen und führen diese mit ihrer Version so zusammen, dass das Update die Originalvorlage ersetzt und nachträglich die Änderungen der SuS angewendet werden. Dies bezeichnet man als **Rebase**. Dadurch vermeidet man den Versionskonflikt, der entstehen kann, wenn beide Änderungen (Update der Lehrperson und eigene Arbeit der SuS) gleichberechtigt zusammengeführt werden würden (merge).

In der Regel betreffen die Änderung an der Vorlage durch die Lehrkraft aber ohnehin Bereiche, in denen die SuS nicht selbst programmiert haben. Daher sollten selbst bei Merge nur selten Versionskonflikte auftreten.

Variante Lösungen zur Verfügung stellen:

Die Lehrperson kann die Lösungen der Aufgaben im Repository als Commit bereitstellen. Die SuS pullen die neue Variante und mergen mit ihrer Lösung. Die SuS können dabei ihre Version mit der Lehrerversion vergleichen und ggf. durch Musterlösung ersetzen. Dafür ist aber eine Konfliktbehandlung beim Mergen erforderlich.

3. SuS versionieren online (SuS arbeiten zu Hause und in der Schule)

Einsatzbereich: IMP ab Klasse 10

Voraussetzungen

- schuleigene GIT-Camp Instanz mit Zugängen für SuS
- Fork, Clone, Stage, Commit, Pull, Push

Die Lehrperson legt Accounts für die SuS auf der schuleigenen GIT Camp Instanz an und stellt dort Material in einem Repository. Die SuS brauchen einen lesenden Zugriff auf das Repository.

Die SuS forken dieses Repository direkt in der Web-Oberfläche von GIT Camp. Dadurch haben sie einen eigenen Ableger der Projekts, bei dem sie volle Zugriffsrechte haben. Sie sollten auch der Lehrperson volle Zugriffsrechte (zumindest Leserechte) einräumen.

Es ist alternativ auch möglich, das Repository der Lehrkraft zu clonen. Grundsätzlich verwendet man einen Fork, wenn man beabsichtigt ein Projekt eigenständig weiterzuentwickeln, welches unabhängig vom Original existiert. Clone wird verwendet, wenn man in einem Team gemeinsam an einem Repository arbeitet und einzelne Teile des Projekts bearbeitet. Daher ist in diesem Szenario ein Fork sinnvoller, auch wenn ein Clone genauso benutzt werden könnte.

Das eigene Repository clonen die SuS sowohl in der Schule als auch zu Hause und führen am Ende jeder Stunde ein Commit mit anschließendem Push durch, so dass sie zu Hause mit einem Pull auf die aktuelle Version zugreifen können.

Machen Sie den SuS deutlich, dass es sehr wichtig ist, immer einen Pull durchzuführen, bevor man anfängt zu arbeiten. Vergessen die SuS dies, kann es Änderungen in der Schule und zu Hause geben, die im Konflikt zueinander stehen. Die SuS merken dies daran, dass sie die zweite Änderung nicht mehr pushen können. GIT meldet die Konflikt und verlangt zunächst ein Pull/Merge bevor die nächste Push Aktion möglich ist. Es muss also aufwendig ermittelt werden, welche Version die richtige ist (Conflict Solver), was während einer Unterrichtsstunde nicht immer so einfach ist.

Die Lehrperson hat über die GitCamp-Webseite jederzeit Einblick in den Stand der Schülerrepositorys. Sie sieht alle Forks im eigenen Projekt.

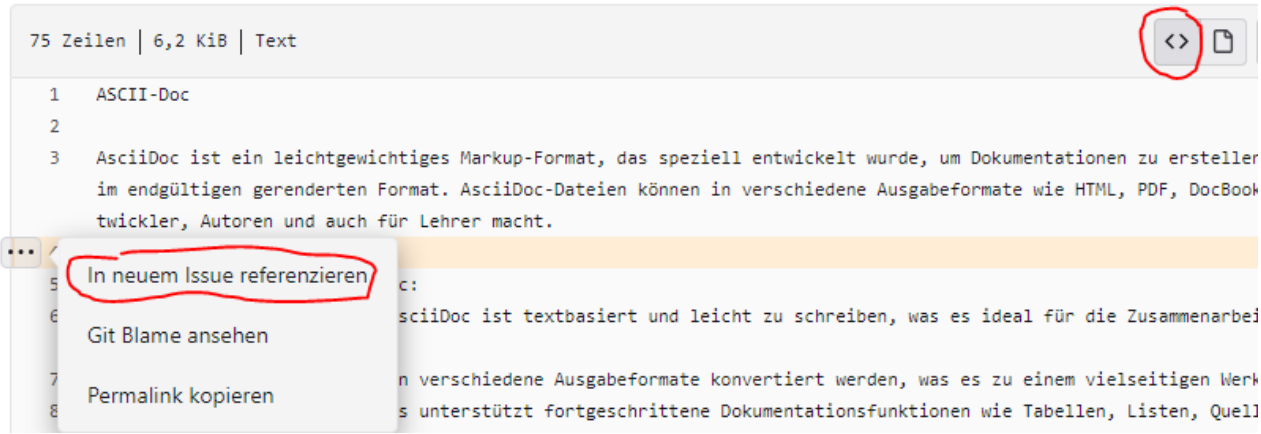
Die Lehrperson kann im eigenen Projekt lokal weitere Branches für jeden SuS anlegen und den tracking branch auf die Repositorys der SuS legen. Dann kann die Lehrperson den aktuellen Stand der Schülerprojekte jederzeit lokal anschauen und überprüfen.

Bei Fehlern hat die Lehrperson die Möglichkeit die Änderungen direkt am Code durchzuführen und Kommentare dazu beim Commit unterzubringen. Anschließend push die Lehrperson die Änderungen. Die Sus sehen die Änderungen, die von der Lehrperson vorgenommen wurden und lesen die Kommentare. Auch hier ist es dringend geboten, dass die SuS zunächst ein Pull durchführen, bevor sie selbst weiterarbeiten. Ansonsten können auch hier Konflikte auftreten. Da die Lehrkraft die eigenen Änderungen aber kennt, sind sie meist leichter zu lösen.

Variante: Kommentieren von Schülerprojekten: In GitCamp hat eine Lehrerin automatisch Leserechte auf den Repos der Schülerinenn. Daher kann eine Lehrerin jederzeit den Arbeitsstand der Schülerinnen ansehen. Dies kann sie entweder online oder indem sie einen lokalen Clone anlegt. Dazu müssen die Repos der Schülerinnen nicht als Fork des Lehrerprojekts

entstanden sein.

Statt die Schülerinnenprojekte nun zu ändern und zu pushen, kann die Lehrerin den Schülerinnen auch einen Issue im online-Repository eintragen. Dieser Issue kann an eine bestimmte Zeile im Code gebunden werden. Dazu lässt sich die Lehrerin im Gitcamp den Quellcode einer Datei anzeigen, so dass links Zeilennummern zu sehen sind (Quellcode muss explizit rechts oben gewählt werden!). Dann kann in der fehlerhaften Zeile ganz links geklickt und ein Issue zu dieser Zeile erstellt werden.



4. Zusammenarbeit in Gruppenprojekten

Einsatzbereich: BF in der Kursstufe

Voraussetzungen

- schuleigene GIT-Camp Instanz mit Zugängen für SuS
- Clone, Stage, Commit, Push, Pull, Merge

Eine einfache Version der Zusammenarbeit von zwei (mehreren) Schülerinnen kann durch die Verwendung eines gemeinsamen online Repositorys erreicht werden. Dabei legt eine Schülerin das Online-Repo in Gitcamp an und räumt den anderen Teammitgliederinnen Lese- und Schreibrechte darauf ein.

Das ganze Team startet mit einer gemeinsamen Vorlage (ggf. ein Fork oder Clone eines Lehrerinnen-Repositorys). Diese Vorlage wird allen Teammitgliederinnen gecloned, damit es lokal auf ihrem Rechner vorliegt. Gleichzeitig arbeiten sie an verschiedenen Aspekten des Projekts. Die geringsten Konflikte erzielt man, wenn jedes Teammitglied an einer eigenen Datei (Klasse) arbeitet.

Nach Abschluss eines Arbeitsschrittes wird die eigene Arbeit durch einen Commit lokal gesichert. Bevor der Arbeitsstand gepusht werden kann, muss vor ein Pull durchgeführt werden, da die anderen Teammitgliederinnen auch Änderungen vorgenommen haben könnten. Aufgrund des gleichzeitigen Arbeitens wird ein Merge der beiden Änderungen notwendig sein. Es sollte stets die Option "Merge to working tree" gewählt werden. Nachdem alle Merge-Konflikte beseitigt sind, das Projekt erneut getestet wurde, wird das zusammengeführte Projekt durch einen weitere Commit gesichert und dann gepusht. Der Push ist nun möglich, da das

zusammengeführte Projekt ein Nachfolger der zuerst gepushten Änderungen ist.

Einsatzbereich: LF in der Kursstufe

Voraussetzungen

- schuleigene GIT-Camp Instanz mit Zugängen für SuS
- Fork, Clone, Stage, Commit, Push, Pull, Merge

Die Rohversion eines Projektes wird ggf. immer noch von der Lehrperson per Repository bereit gestellt. Ein Mitglied wird zum "Sprecher" der Gruppe. Dieser forked das Lehrerprojekt oder legt ein eigenes Repository an, wenn es keine Vorlage gibt. Er nimmt alle anderen Mitglieder der Gruppe als "Developer" in das Repository auf. Alle Gruppenmitglieder clonen diesen Fork und haben damit ihr lokalen Versionen.

Die Gruppenmitglieder arbeiten lokal, committen ihr Änderungen und pushen sie danach. Ab dem Push des zweiten Gruppenmitglieds ist ein Zusammenführen der verschiedenen Änderungen erforderlich. Arbeiten die SuS an verschiedenen Dateien ist dies problemlos möglich. Bei Änderungen an gleichen Dateien müssen die Änderungen gemerged werden und ggf. Konfiktbehandlung durchgeführt werden.

Die SuS könnten auch in verschiedenen Branches arbeiten und erst am Ende ihre Branches zu einem gemeinsamen Projekt zusammenführen. Das Arbeiten mit mehreren Branches setzt aber ein sehr diszipliniertes Arbeiten voraus. Sobald einmal ein falscher Branch aktiv ist, wird es schwierig die Fehler wieder zu korrigieren. Daher ist nur bei sehr fitten SuS zu empfehlen, dieses Konzept einzuführen.