Inhaltsverzeichnis

CampusSource git Plattform

CampusSource git dient der Verbreitung von hochschulbezogenen Softwareprojekten und Verbesserung von Kollaborationsmöglichkeiten. Die Nutzung ist nicht landesspezifisch eingeschränkt.

So geht es los

Nutzungsbedingungen

1. Geltungsbereich und Benutzerkreis

Diese Nutzungsvereinbarung gilt für die von CampusSource e.V. im Internet bereitgestellte Entwicklungsplattform („git.campussource.de“). Die Nutzung der Entwicklungsplattform ist durch die Mitglieder*innen nur zulässig, wenn diese die Nutzungsvereinbarungen zuvor akzeptieren.

Diese Entwicklungsplattform steht weltweit zur Verfügung, und fokussiert den Nutzungsbereich Hochschulen.

2. Registrierung und Zugangskennung

Mitgliedern aus dem unter 1 benannten Benutzerkreis werden in der Regel durch von CampusSource erstellte Zugangsdaten zur Verfügung gestellt. Darüber hinaus gibt es die Möglichkeit sich selbstständig zu registrieren. Dazu steht eine Formularbasierte Registrierung zur Verfügung. Bei einer Selbstregistrierung erfolgt keine automatische Zuweisung in bestehende Gruppen oder Kurse. Es kann daher notwendig sein Rücksprache mit CampusSource oder einem Organisationsverantwortlichem des Repositories zu halten um berechtigten Zugriff auf weitere Inhalte zu erlangen.

Vor- und Nachname sowie der Benutzername der Nutzer*innen werden den Organisationsverantwortlichen und den Mitteilnehmer*innen in den jeweils zugewiesenen Repositories angezeigt.

Durch Nutzung des CampusSource-Angebots unter den Bedingungen dieser Vereinbarung versichert der/die Nutzer/in CampusSource dass:

- die Information, die er/sie CampusSource übermittelt hat, um sich als Benutzer*in zu registrieren, vollständig, korrekt und aktuell gültig ist.
- dass er/sie CampusSource über alle Änderungen Ihrer Benutzerdaten informieren wird und/oder seine Benutzerdaten selbstständig aktuell hält.
- dass er/sie seine/ihre Nutzerkennung (Username) und sein/ihr Passwort niemandem mitteilen wird.

Falls irgendeine der Nutzerinformationen wissentlich falsch gegeben wurden oder bei Änderung der Nutzerdaten diese absichtlich nicht korrigiert werden, oder der/die Nutzer/in in irgendeiner Weise gegen diese Bestimmungen verstößt, hat CampusSource das Recht, seine/ihre Nutzungsberechtigung vorläufig aufzuheben oder zu beenden.

3. Leistungsvereinbarung und Nutzungsberechtigung

Die Nutzungsberechtigung der Nutzer*innen umfasst den Zugang zu den Inhalten der Entwicklungsplattform CampusSource. Im Einzelnen sind dies:

- Das via Internet Abrufen und Nutzen der Repositories auf ein dem/der Nutzer/in oder einem/r Dritten gehörenden Daten verarbeitenden Gerät zum Zwecke des Informierens, Entwickelns und Austausch.
- Das Bearbeiten der Repositories.
- Die Kommunikationsmöglichkeiten der Entwicklungsplattform bestehend aus Email, Wiki und Ticketsystem.
- Die Möglichkeit zum Bereitstellen von eigenen Inhalten in den dafür freigeschalteten Bereichen

CampusSource ist bemüht, den Dienst verfügbar zu halten. CampusSource übernimmt keine darüber hinausgehenden Leistungspflichten. Insbesondere besteht keine Garantie auf eine ständige Verfügbarkeit des Dienstes.

CampusSource übernimmt keine Gewähr für die Richtigkeit, Vollständigkeit, Verlässlichkeit, Aktualität und Brauchbarkeit der bereitgestellten Inhalte.

4. Pflichten der Nutzer*innen

Der/die Nutzer/in verpflichtet sich gegenüber CampusSource, keine Beiträge zu veröffentlichen, die gegen die guten Sitten oder geltendes Recht verstoßen oder rechtswidrige Inhalte aufweisen. Der Nutzer/die Nutzer/in verpflichtet sich insbesondere dazu, keine Beiträge zu veröffentlichen:

- deren Veröffentlichung einen Straftatbestand erfüllt oder eine Ordnungswidrigkeit darstellt,
- die gegen das Urheberrecht, Markenrecht oder Wettbewerbsrecht verstoßen,
- die gegen das Rechtsdienstleistungsgesetz verstoßen,
- die beleidigenden, rassistischen, diskriminierenden oder pornographischen Inhalt haben,
- die Werbung enthalten.

Bei einem Verstoß gegen diese Verpflichtung ist CampusSource berechtigt, die entsprechenden Beiträge abzuändern oder zu löschen und den Zugang des Nutzers/der Nutzerin zu sperren. Der/die Nutzer/in ist verantwortlich für jede Nutzung des CampusSource-Angebots, die mit Hilfe seiner/ihrer Nutzerkennung (Username) und des Passworts ausgeführt wird. Er/sie hat dafür Sorge zu tragen, dass seine/ihre Nutzerkennung (Username) und Passwort vor unautorisiertem Gebrauch geschützt sind. Falls der/die Nutzer/in eine missbräuchliche Verwendung seiner/ihrer Zugangsdaten bemerkt oder vermutet, verpflichtet er/sie sich, CampusSource unverzüglich zu benachrichtigen CampusSource hat das Recht, Beiträge und Inhalte zu löschen, wenn diese einen Rechtsverstoß enthalten könnten. Ebenfalls hat CampusSource das Recht die verursachenden Accounts zu sperren oder von der Entwicklungsplattform zu entfernen.

Um Mißbrauch zu vermeiden ist die Datenübertragung begrenzt auf 2 GB pro Kalenderwoche. Wenn ein Repository 2 Jahre nicht mehr genutzt wird (d.h. kein Commit mehr stattgefunden hat), wird die registrierte Email Adresse des Besitzers per Email informiert. Wenn nach Absendung der Mail nach 2 Monaten kein Commit mehr stattgefunden hat, wird das Repository archiviert und die Archivdatei der registrierten Email Adresse des Besitzers zur Verfügung gestellt. Danach wird das Repository gelöscht.

5. Datenschutz

Die Informationen zum Datenschutz entnehmen Sie bitte der Datenschutzerklärung der Entwicklungsplattform CampusSource unter:

https://www.campussource.de/datenschutz/

6. Urheberrecht

Die in Gestalt der Repositories der Entwicklungsplattform bereit gestellten Werke sind urheberrechtlich geschützt. Daher bedarf es für jede Nutzung, die über die im UrhG geregelten Fälle der erlaubnisfreien Nutzung hinausgeht, einer vorherigen Genehmigung. Wir möchten ausdrücklich darauf hinweisen, dass es sich bei den bereit gehaltenen Werken nicht um (freie) amtliche Werke im Sinne des § 5 UrhG handelt.

7. Haftungsausschluss

Als Diensteanbieter ist CampusSource gemäß §§ 7, Abs.1 und 2 TMG für eigene Informationen verantwortlich, die sie zur Nutzung bereithält. Nach §§ 8 TMG besteht keine Verantwortung für die von Nutzern eingestellten Informationen oder für die von anderen Anbietern bereit gestellten Inhalte, auf die mittels Hyperlinks verwiesen wird. Sollten mit unserem Angebot verlinkte Seiten aus fachlichen oder rechtlichen Gründen Anlass zur Beanstandung geben, bitten wir um eine entsprechende Mitteilung an die Redaktion der betreffenden Seite. CampusSource als Betreiber der Entwicklungsplattform haftet ausnahmslos nicht für Leistungsstörungen, die an oder aufgrund der von dem/der Nutzungsberechtigten bereitzustellenden Hard- und Software entstehen oder mitverursacht werden. Dies schließt jegliche Schäden ein, einschließlich entgangener Einsparungen oder anderer Schäden, die zufällig oder als Konsequenz aus dem Kopieren per Download, der Anwendung oder Unfähigkeit der Anwendung der Dokumente resultieren. Zudem wird ausdrücklich darauf hingewiesen, dass das Risiko, sich den Rechner durch Herunterladen von Dateien aus dem Internet mit einem Virus zu infizieren, jede/r Nutzer/in selbst trägt. Die Haftung für etwaige Schäden wird nicht übernommen.

8. Etikette

In den Foren und dem Austausch per Mail und/oder Tickets sollte ausdrücklich auf einen fairen, freundlichen und konstruktiven Umgangston geachtet werden. Beleidigungen bzw. einzelne oder die Gruppe/den Kurs schädigende Aussagen sind zu unterlassen. Bei Verstoß gegen diese Grundsätze behält sich CampusSource vor die verursachenden Accounts zu deaktivieren und/oder zu löschen.

9. Wiederruf und Löschung

Der/die Nutzer/in kann jederzeit die von ihm bereitgestellten Informationen, Dateien und Beiträge selbstständig löschen sowie die Einwilligung in die Nutzungsvereinbarung widerrufen. Der/die Nutzer/in kann jederzeit die Löschung seines Accounts beantragen und/oder der Nutzungsvereinbarung widersprechen. Dies kann selbstständig in den persönlichen Einstellungen der Entwicklungsplattform durchgeführt werden oder indem sich der/die Nutzer/in direkt an CampusSource wendet. Hierzu nutzt er die im System angegebene E-Mail-Adresse CampusSource (siehe Impressum). Eine durchgeführte Löschung eines Accounts kann nicht rückgängig gemacht werden. CampusSource weist darauf hin, dass bei einem reinen Widerspruch gegen die Nutzungsvereinbarung kein Zugriff mehr auf die Entwicklungsplattform CampusSource gewährt werden kann.

10. Schlussbestimmung

Auf die vertraglichen Beziehungen zwischen CampusSource und dem Nutzenden findet das Recht der Bundesrepublik Deutschland Anwendung. Gerichtsstand für sämtliche Streitigkeiten ist Hagen, Deutschland.

Sollten einzelne Regelungen dieser Nutzungsbedingungen unwirksam sein oder werden, wird dadurch die Wirksamkeit der übrigen Regelungen nicht berührt. Die Vertragspartner verpflichten sich, eine unwirksame Regelung durch eine solche wirksame Regelung zu ersetzen, die in ihrem Regelungsgehalt dem gewollten Sinn und Zweck der unwirksamen Regelung möglichst nahe kommt. Das gilt entsprechend bei Vertragslücken.

Registrieren

imgsrccsgit_reg1.pngwidth400pxcaptionRegistrierung

Um an unseren und hochschulinternen Projekten teilzunehmen muss man sich lediglich in unserem GIT registrieren.

Dazu sind folgende Schritte notwendig:

1. Auf Seite gehen: https://git.campussource.de/git/
2. Oben rechts auf "Registrieren" klicken
3. Benutzername und Passwort wählen und E-Mail Adresse angeben
4. Sie erhalten einen Verifierungslink per EMail, wenn Sie den Nutzungsbedingungen zustimmen klicken Sie auf den Link und bestätigen die Anmeldung mit dem Passwort. Fertig

imgsrcgit_reg2.jpgwidth400pxcaptionRegistrierung

Danach ist man sofort angemeldet.

imgsrcgit_ang.jpgwidth400pxcaptionAngemeldet

Nutzung der Anwendung

Repository einrichten

Ein Repository ist verknüpft mit der Person, die es anlegt. Damit werden auch die Schreibrechte nur an Sie vergeben. Wenn Sie im Team arbeiten wollen: Bitte teilen Sie uns nach der Registrierung die jew. Kennungen mit und wir richten für Sie dann die Organisation ein.

Repository verwenden

Das Repository kann man im Browser, lokal auf dem PC oder auch direkt auf dem Server verwenden.

Daten bearbeiten

Um Daten zu bearbeiten oder erst mal einen bestimmten Stand in das Git zu bekommen kann der Browser oder auch Eclipse oder ähnliches verwendet werden. Die Ordnerstruktur sollte der auf dem Server ähneln, um später diese auch direkt auf dem Server Synchronisieren zu können.

Daten im Browser bearbeiten

Um die Daten im Browser zu bearbeiten, geht man zunächst in das gewünschte Repository. Dazu klickt man in der Übersicht unter Repositories auf das gewünschte. Hier im Beispiel steht nur eins zur Verfügung.

imgsrcgit_edit1.jpgwidth400pxcaptionBearbeiten

Man landet im root Verzeichnis des Repositories. Mit Klick auf eine Datei kann diese nun bearbeitet werden oder mit klick auf einen Ordner, gelangt man in diesen. Wenn es mehrere Unterordner gibt und dazwischen keine weiteren Verzweigungen oder Dateien, werden diese gebündelt dargestellt. D.h. bei klick auf diesen Link gelangt man direkt zu dem Punkt an dem es Dateien oder weitere Verzweigungen gibt.

imgsrcgit_edit2.jpgwidth400pxcaptionBearbeiten

Möchte man eine neue Datei irgendwo dazwischen anlegen klickt man zuerst auf den gebündelten Ordner und dann oben drüber in der Pfadangabe auf den Ordner wo man hin möchte.

imgsrcgit_edit3.jpgwidth400pxcaptionBearbeiten

Um eine Datei nun zu bearbeiten geht man in den entsprechenden Ordner und klickt diese an. Dadurch kann man diese erst mal betrachten.

imgsrcgit_edit4.jpgwidth400pxcaptionBearbeiten

Wenn man nun auf den Stift klickt, kann man diese auch bearbeiten.

imgsrcgit_edit5.jpgwidth400pxcaptionBearbeiten

Wenn die Arbeiten abgeschlossen sind, scrollt man nach unten. Dort gibt es 2 Textfelder unterhalb von "Anderungen committen". In dem ersten gibt man eine kurze Beschreibung ein, was geändert wurde und in dem unteren Textfeld eine lange Beschreibung.

imgsrcgit_edit6.jpgwidth400pxcaptionBearbeiten

Die kurze Beschreibung erscheint immer direkt neben den Dateien in der Übersicht. Hier am besten nur ganz wenige Stichworte verwenden.

imgsrcgit_edit7.jpgwidth400pxcaptionBearbeiten

Die lange Beschreibung kann man sich mit klick auf die Kurzbeschreibung anzeigen lassen und dazu auch direkt die Änderungen. Somit sieht man die Änderungen und hat auch direkt die lange Erklärung dazu.

imgsrcgit_edit8.jpgwidth400pxcaptionBearbeiten

Klonen und Synchronisieren des Repository

Klonen des Repository

Sie können ein git Repository auf Ihr lokales System klonen, wenn Sie

- unter Linux: die Anwendung **git** installieren
- unter Windows: Spezielle Client Programme wie Tortoise-git oder Eclipse (s.u.) nutzen.

Die URL zum Klonen lautet z.B.

```
git clone https://git@git.campussource.de/git/Organisation/Repository.git
```

Danach können Sie mit den üblichen Kommandozeilen-Befehlen (git pull etc.) synchronisieren.

Zugang mit SSH

Auf Anfrage öffnen wir für einzelne IP-Nummernkreise auch den SSH-Port vom Git-Server. Damit ist die Git Anbindung etwas stabiler, außerdem kann man die Passworteingabe leicht ausschalten.
Den SSH Zugriff bieten wir aus Sicherheitsgründen nicht weltweit, sondern nur für IP-Nummernkreise. Senden Sie uns eine Nachricht, wenn Sie Interesse haben

Dazu muss der SSH Schlüssel des Servers im GIT Profil hinterlegt werden. Loggen Sie sich dazu auf https://git.campussource.de/git mit der Kennung, die auf dem Server verwendet werden soll ein und wechseln oben rechts bei dem Profil auf Einstellungen.

imgsrcgit_ssh_key.jpgwidth400pxcaptionSSH-Key

Hier sind folgende Schritte nötig:

1. Klick auf auf "SSH- / GPG-Schlüssel".
2. Dann bei "SSH-Schlüssel verwalten" auf "Schlüssel hinzufügen" klicken.
3. Für den SSH Schlüssel einen Namen wählen. Der Name dient nur zur Identifizierung für einen selbst. Kann also frei gewählt werden.
4. Bei Inhalt kommt dann der Schlüssel aus "~/.ssh/id_rsa.pub" aus dem Home Verzeichnis des Servers rein. (Falls es die Datei auf dem Server noch nicht gibt kann diese mit "ssh-keygen" in der Shell angelegt werden)
5. Zum Abschluss noch auf "Schlüssel hinzufügen" klicken. Fertig.

Wenn das eingerichtet ist sollte mit folgendem Befehl das Repository verwendet werden können.

```
git clone ssh://git@git.campussource.de:22222/Organisation/Repository.git
```

Sie können auch ein bereits per HTTP geklontes Repo umstellen, indem Sie in der Datei ./git/config die Variable **url** auf folgenden Wert ändern:

```
url = ssh://git@git.campussource.de:22222/Organisation/Repository.git
```

Migration eines Repository

Sie können ein Repository auch von einer anderen GIT-Anwendung importieren, z.B.**github** oder gitea selbst. Gehen Sie dazu in der Zielanwendung auf das "+"-Icon rechts oben und wählen Sie "Neue Migration". Danach können Sie die Quelle angeben, z.B. gitea. Dort geben sie dann an

- Die Migrations- / Klon-URL des Quell-Repo
- Den Besitzer
- Ggf. ein Zugangs-Token. Bei gitea muss man zur Migration ein Token über das API anlegen.
  - Wenn die Gitea-Anwendung z.B. hier liegt: https://superx-rocks.de/git/
  - Dann rufen Sie in der Shell z.B. auf:

```
curl -XPOST -H "Content-Type: application/json"  -k -d '{"name":"test"}' -u username:password https://superx-rocks.de/git/api/v1/users/username/tokens
```

Sie erhalten z.B.

```
{"id":1,"name":"test","sha1":"9fcb1158165773dd010fca5f0cf7174316c3e37d","token_last_eight":"16c3e37d"}
```

Die Zeichenkette hinter "sha1:" geben Sie in gitea im Feld "Zugangs-Token" ein. Danach können Sie die Migration starten.

Die wichtigsten git Funktionen

Woher kommt git

Linus Torvalds, der "Vater" von Linux, nutzte anfangs eine kommerzielle Versionierungsssoftware. Als die Lizenz restriktiver wurde, suchte er nach Alternativen. Die damaligen "Platzhirsche" CVS und SVN gefielen ihm nicht, u.a. weil sie (zum Committen) eine permanente Verbindung zum Server voraussetzte. Daher entwickelte er kurzerhand eine neue Software, die alle Features bot, die er brauchte. Da er wegen "Linux" im Ruf stand, seine Softwareprodukte mit seinem eigenen Namen zu verbinden, taufte er seine Software "git" (zu deutsch "Trottel") - ein ironischer Seitenhieb.

Nützliche Werkeuge

Wir haben oben beschrieben wie Sie mit git im Browser arbeiten können. Es gibt aber mit der Kommandozeile noch wesentlich mächtigere Werkzeuge.

- erstes Tool der Wahl ist natürlich das Kommandozeilen-Werkzeugt "git". Details dazu siehe unten.

git gui

- git gui ist ein (etwas altbacken wirkendes) graphisches Frontend

imgsrcgitgui.pngwidth800pxcaptiongit gui

Vorsicht: Sie sollten das tool nur in der englischen Lokalisierung nutzen, bei der deutschen stiften Sie nur Verwirrung. Auch bei Konflikten arbeitet es nicht sehr gut.

gitk

Auch das Tool "gitk" wirkt zwar auf den ersten Blick recht "altbacken", aber es eignet sich sehr gut zum Durchsuchen einer Commit Historie bzw. zum Prüfen von Änderungen.

imgsrcgitk.pngwidth800pxcaptiongitk Beispiel

Eclipse Plugin

- Auch in Eclipse gibt es ein git-Plugin, das bei der Java oder J2EE-Edition nicht einmal mehr nachinstalliert werden muss. Dies fügt sich nahtlos in die Oberfläche ein:

imgsrcgit_eclipse.pngwidth800pxcaptionEclipse und git

Sie finden die git-bezogenen Menüs mit der rechten Maustaste unter "Team".

Nützliche Kommandozeilen-Befehle

Umgebung

Die Kommandozeile funktioniert, wenn man sich in einem Verzeichnis befindet, unterhalb dessen das Repository "geklont" wurde. Mit

```
git status
```

zeigt man den aktuellen Status an.
Unterhalb es Stamm-Verzeichnisses gibt es lokal immer einen Ordner ".git", der unsichtbar ist. Und dort gibt es eine Textdatei "config", die wichtige Konfigurationen enthält.

Das lokale Verzeichnis ist ein Klon des Remote Repository. Mit

```
git pull
```

bekommen Sie immer den aktuellen Stand. Dies sollten Sie regelmäßig machen, damit Ihre lokale Installation nicht zu sehr divergiert.

Arbeit mit Branches

Lokale Branches

Bei git entwickelt man typischerweise in sog. "Branches", d.h. man zweigt vom ausgelieferten Code (der "Hauptbranch", früher hieß der "master", neuerdings nennt man es "Default-Branch") ab, entwickelt dort in Ruhe, und wenn alles funktioniert "merged" man seinen Branch in den Hauptbranch.

In welchem Branch man sich befindet, zeigt folgendes Kommando an:

```
git branch
```

Ergebnis z.B.:

```
* master
```

Dabei wird der Branch, in dem man sich befindet mit einem "*" gekennzeichnet. Einen neuen lokalen Branch erstellt man einfach mit

```
git checkout -b meine_coole_entwicklung
git branch
master
* meine_coole_entwicklung
```

Wenn ein erstellter lokaler Branch nicht mehr benötigt wird, kann dieser mit

```
git branch -D meine_coole_entwicklung
```

gelöscht werden.

Remote Branches

Der neue Branch ist zunächst nur lokal verfügbar. Dies kann man so beibehalten, falls man nur alleine in diesem arbeitet. Er lässt sich jedoch auch mit

```
git push  --set-upstream origin meine_coole_entwicklung
```

auf ein entferntes Repository veröffentlichen um mit anderen zusammen zu arbeiten.

Wenn man fertig ist und einen stabilen Stand hat, kann man diesen in den "master" mergen:

```
git checkout master
git pull
git merge meine_coole_entwicklung
git push
```

Alle verfügbaren romote-Branches kann man sich anzeigen mit

```
git branch -r
```

die entfernten Branches anzeigen lassen (-a würde lokale und entfernte anzeigen) und den entsprechenden Branch mit

```
git checkout -b jemandes_coole_entwicklung origin/jemandes_coole_entwicklung
```

lokal verfügbar machen.

Auch den Remote Branch können Sie löschen mit

```
git push origin :meine_coole_entwicklung
```

Wenn Branches zu grob sind

Manchmal will man in der Commit Historie zu einem speziellen Commit zurückkehren (z.B. um zu prüfen ob damals noch alles funktionierte). Dafür sind Branches nicht geeignet, weil man die im Nachhinein nicht anlegen kann. Sie können über die Commit Historie den Hash ausfindig machen und so ganz leicht zum damaligen Stand zurückkehren, z.B.:

```
git checkout 4735d97b8f44e6d809783019f6ce0fe515d621c2
```

Hier sollten Sie nicht committen oder branchen, es dient nur der Diagnose.

Committen

Neue Commits erstellen

Bei git committen Sie in zwei (oder drei) Schritten:

1. Sie ändern den Code, und fügen dann die Änderung mit

```
git add Dateipfad
```

auf eine Art "Bühne", also Dateien, die für einen Commit vorgemerkt sind. Dies wiederholen Sie für beliebig viele Dateien, die Sie in einem Commit bündeln wollen, oder Sie geben direkt ein

```
git add -A -v
```

Damit werden alle geänderten Dateien auf die Bühne geschoben.
Vorher sollten Sie mit *git status* anzeigen, was alles "geadded" wird.

2. Sie committen mit einer sprechenden Nachricht:

```
git commit -m "Bugfix meiner coolen Entwicklung #Ticketnr"
```

3. Wenn Sie mit remote branches arbeiten publizieren Sie die Änderung mit

```
git push
```

Commit-Hashes

Git speichert jeden einzelnen Commit mit einem sog. "Hash"-Wert, also einer SHA-1-Zufalls-Zeichenkette, die eindeutig ist und im Sinne einer digitalen Signatur auch nicht änderbar ist.

Die Hashes können Sie z.B. nutzen, um Zustände einer Datei

- aus der Vergangenheit wieder herzustellen (checkout)
- Einzelne Commits zurückzunehmen (revert)
- Differenzen zwischen Commits anzuzeigen (diff

Eine Konvention bei git ist es, dass man die Hashes abkürzen kann, also statt des gesamten Hashes z.B.

```
4735d97b8f44e6d809783019f6ce0fe515d621c2
```

kann man auch nur die ersten 8 Stellen nehmen:

```
4735d97b
```

Commit-Historie einsehen

Mit dem Kommando

```
git log --pretty=format:"%h - %an, %ar : %s" Dateipfad
```

kann man sich die Historie von Dateien ansehen. Noch komfortabler geht das mitgitk

imgsrcgitk.pngwidth800pxcaptiongitk

Den Hash kann man sich im Feld "SHA1-ID" mit STRG-C rauskopieren.

Wenn man z.B. einen Patch erstellen will, kann man die Dateien erstmal auflisten:

```
git log --name-only Dateipfad
```

Diese Dateiliste kann man in eine Textdatei kopieren, und mit folgenden Kommando sortieren und alle Duplikate entfernen, z.B. die Datei files.txt:

```
cat files.txt | sort -u >files_sorted.txt
```

Daraus kann man dann eine zip-Datei erzeugen, mit allen Dateien in files_sorted.txt:

```
zip mein_patch.zip -@ < files_sorted.txt
```

Wenn Sie in der Shell Differenzen einer Datei zwischen zwei Commits einsehen wollen, können Sie die SHA-Ids aus obigem git log Kommando direkt vergleichen, mit

```
git diff <
```

also z.B.

```
git diff 3e9919e7c 231b35927 -- superx/WEB-INF/conf/edustore/db/module/sos/hilfstabellen/sos_stg_aggr_fuellen.sql
```

Wenn Sie einen Commit nach dem Hash z.B. 3e9919e7c suchen, schreiben Sie

```
git show 3e9919e7c
```

Konflikte

Wenn es Konflikte gibt, z.B. weil die gleiche Datei von verschiedenen Parteien geändert und "gepushed" wurde, können Sie ein sog. "mergetool" nutzen, um die Änderungen einzeln zu prüfen:

```
git mergetool --tool=meld
```

Resetten

Falls Sie Ihre Änderungen verwerfen möchten, können Sie wie folgt vorgehen:

Wenn einzelne Dateien/Verzeichnisse zurückgesetzt werden sollen, und noch nicht geadded/commited wurde:

```
git checkout -- Dateipfad
```

Bei Angabe eines Verzeichnisses werden rekursiv alle Dateien in allen Unterverzeichnissen zurückgesetzt.

Wenn man z. B. alle Änderungen im aktuellen Verzeichnis und dessen Unterverzeichnissen verwerfen will, gibt man ein:

```
git checkout -- .
```

(wichtig ist der Punkt am Ende)

Wenn Dateien schon committed wurden, kann man die remote-Datei erzwingen mit

```
git checkout --theirs -- Dateipfad
```

Wenn Commits schon gepusht wurden, holt man sich aus der Commit Historie den Hash des Commits, um dann z.B. den Commit rückgängig zu machen:

```
git revert 4735d97b8f44e6d809783019f6ce0fe515d621c2
git push
```

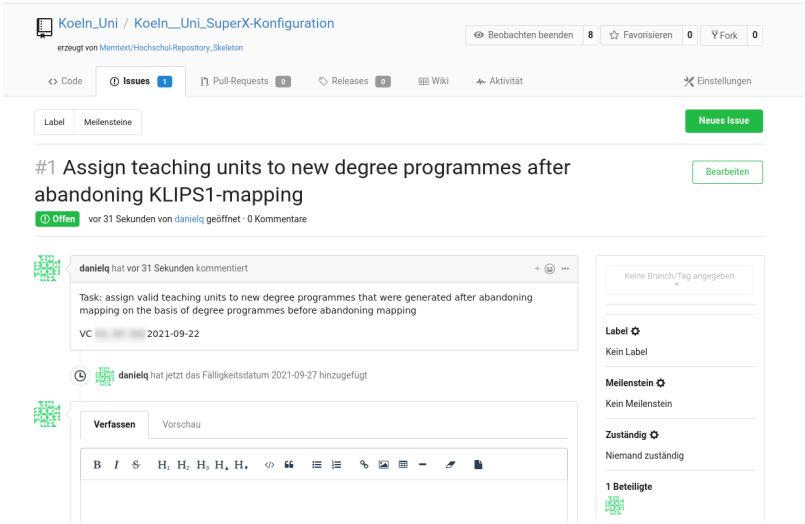
Dokumentation

Zur Dokumentation können Sie das interne Wiki und Ticket-System nutzen.

Ticket System

Tickets anlegen und bearbeiten

Im Ticket System können Sie ein Ticket anlegen, im Menü "Issue":



Hier können Sie ein Thema, eine Beschreibung, und Ziel-Meilensteine bzw. Fälligkeiten definieren. In der rechten Seitenleiste können Sie auch steuern, ob Sie bei Änderungen Emails "abonnieren" o

Es gibt in der Bearbeitung ein paar Unterschiede zu HISZILLA:

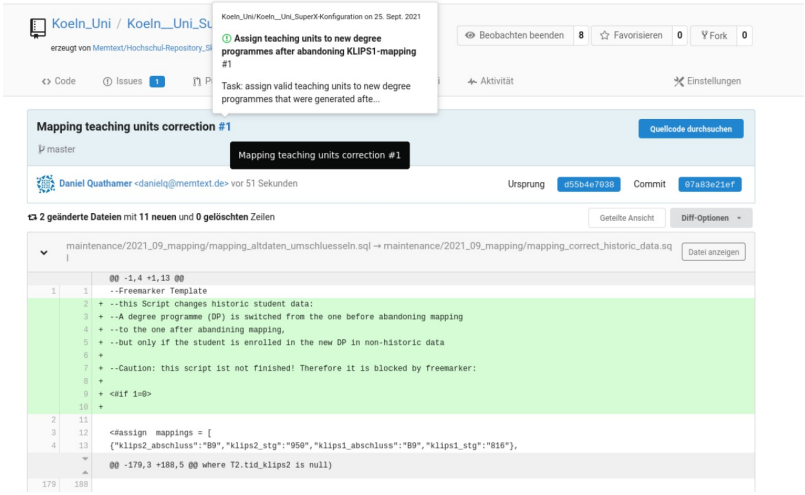
- Ticket Kommentare lassen sich bearbeiten / löschen
- Sie können nur bestimmte Dateitypen anhängen (Text). Binärdateien oder große Dateien bitte über git selbst hochladen.
- Zur Fomatierung wird Markdown genutzt
- Man kann in seinem Profil einen Avatar wählen oder anlegen

Kombination mit Commits

Jedes Ticket bekommt eine fortlaufende Nummer (hier im Beispiel die #1), auf die Sie in Commits referenzieren können. Wenn Sie z.B. einen Commit mit der Message

```
git commit -m "Mapping teaching units correction #1"
```

anlegen, wird dieser später in der Browser Oberfläche automatisch mit den Ticket verlinkt:



Sie können auch auf Issues in andere Repositories verlinken, nach dem Schema

owner/repository#1234