Carl-Zeiss-Gymnasium Jena

Klassenstufe 10

Git und GitHub

Koch, R.

Schuljahr 2025/2026

Fachschaft Informatik

1 Git und GitHub

1.1 Definitionen und Funktionen

Definition: Git

Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien.

Diese Dateien liegen in sogenannten Repositories. Jeder Benutzer besitzt eine lokale Kopie des gesamten Repositorys inklusive der Versionsgeschichte (history). So können die meisten Aktionen lokal und ohne Netzwerkzugriff ausgeführt werden.

Definition: Repository

Ein **Repository** ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung digitaler Objekte (Programme, Publikationen, Datenmodelle, ...).

Daten können neben dem Übertragen auf das eigene System mit unterschiedlichen Netzwerkprotokollen zwischen Repositories ausgetauscht werden.

Für die Verwaltung stehen unter anderem die folgenden Funktionen zur Verfügung:

branching – Erstellen neuer Entwicklungszweige zum Testen und Weiterentwickeln,

merging – Verschmelzen zweier oder mehrerer Zweige,

committing – Freischaltung einer oder mehrerer Änderungen.

Für den Austausch stehen unter anderem die folgenden Funktionen zur Verfügung:

forking – Kopieren eines Repository,

push – Hochladen eines Commits in ein Repository,

pull – Herunterladen eines Repositorys und updaten der lokalen Dateien.

1.2 Github

Um den Umgang mit Git intuitiver zu gestalten, wurden verschiedene Plattformen entwickelt. Eine der größten Open-Source-Software war **Github**, dass 2018 von Microsoft übernommen wurde. Ein weiterer Vorteil liegt in der Organisation von kollaborativen Projekten.

1.2.1 forking – eine eigene Kopie anlegen

Logge dich in deinen GitHub-Account ein und navigiere zu dem Repository, das du kopieren möchtest (z. B. `https://github.com/CZG-KoR/10C1_25`). Drücke dort auf den Forkbutton.

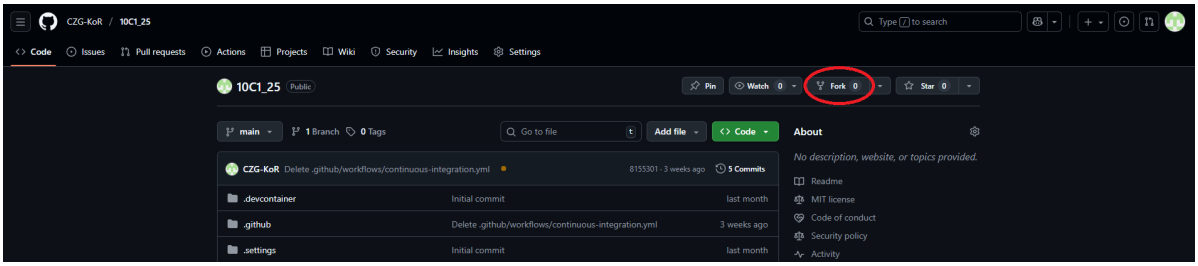

Abbildung 1.1: Fork erstellen 1

Wähle deinen Account als Owner, einen Repository name (der kann auch beibehalten werden) und füge ggf. eine Description hinzu. Der Vorgang wird mit dem markierten Button abgeschlossen.

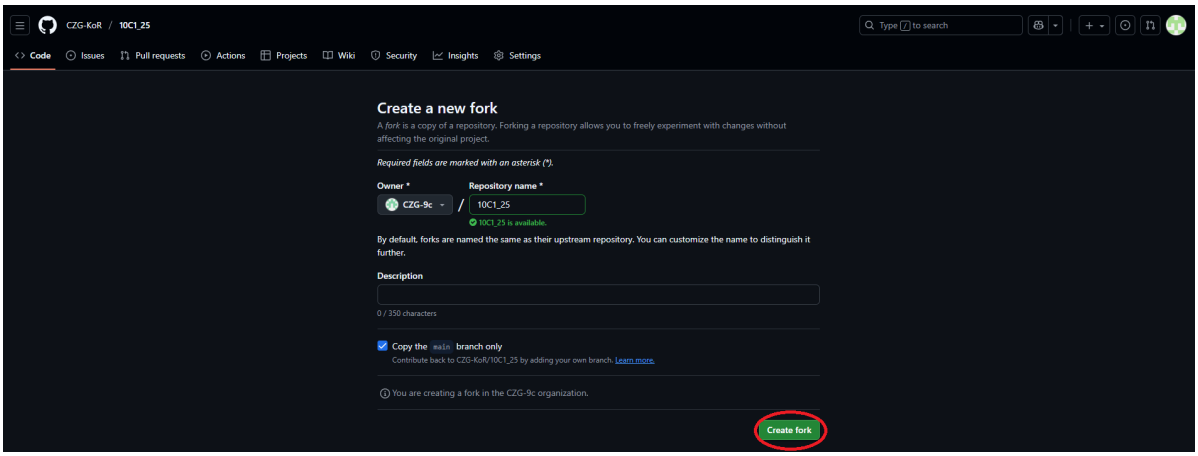

Abbildung 1.2: Fork erstellen 2

1.2.2 Verwaltung des eigenen Repository

Im eigenen Verzeichnis, im Reiter Repositories, befindet sich nun eine Kopie des originalen Repository. Wählt man dieses an, können die Dateien online verwaltet werden.

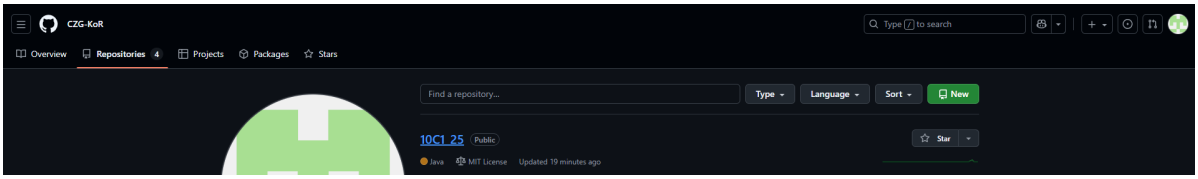

Abbildung 1.3: Repositoryübersicht

Über die markierten Button können Daten hochgeladen oder das gesamte Repository heruntergeladen werden.

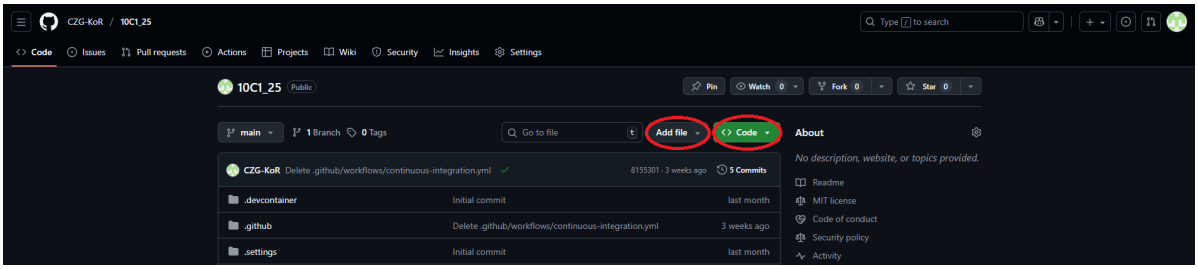

Abbildung 1.4: Hoch- und Runterladen

Navigiert man zu einer Datei kann diese im integrierten Codespace editiert oder gelöscht werden. Alle Änderungen werden durch einen Commit versioniert und den entsprechenden Branch kopiert. Es ist auch möglich einen neuen Branch zu erstellen. Dies sollte immer dann getan werden, wenn man sich nicht 100% sicher ist, ob die vorgenommene Änderung funktional ist.

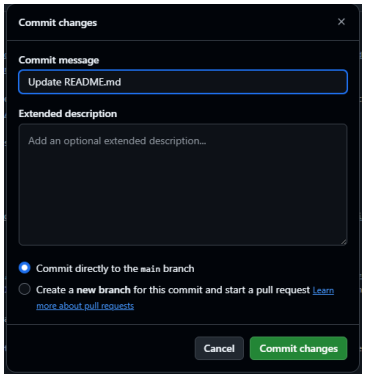

Abbildung 1.5: Commit

1.2.3 Eigenes Repository mit dem Original synchronisieren

Wenn alle Commits lokal stabil laufen, können sie in das Ursprungsrepository übergeben werden. Dafür muss zuerst der Fork synchronisiert werden.

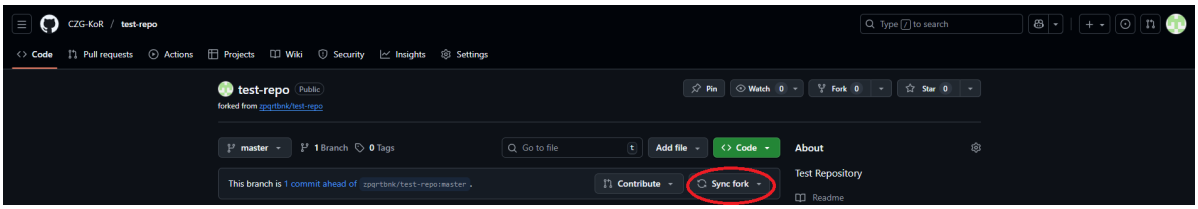

Abbildung 1.6: Synchronisierung

Sollte sich das Ursprungsrepository durch Commits anderer Mitarbeiter geändert haben, kann es zu Konflikten kommen. Es können sich z. B. Variablennamen oder Funktionsaufrufe zwischenzeitlich geändert haben. Bevor der eigene Code hochgeladen werden kann, müssen diese Konflikte gelöst werden. Diese sind im Codespace hervorgehoben.
Nach der Lösung kann eine Anfrage an den Besitzer des Ursprungsrepositorys gestellt werden, die Commits zu übernehmen.

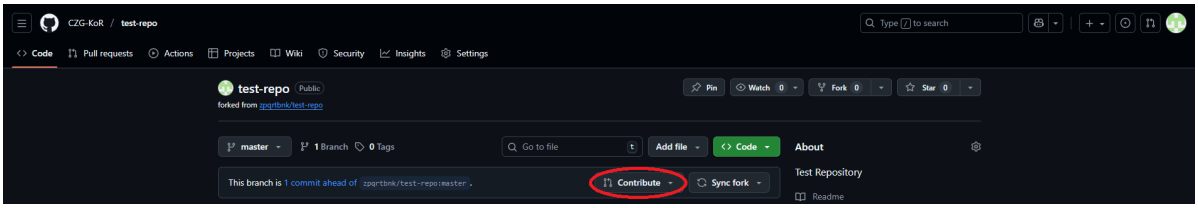

Abbildung 1.7: pull request 1

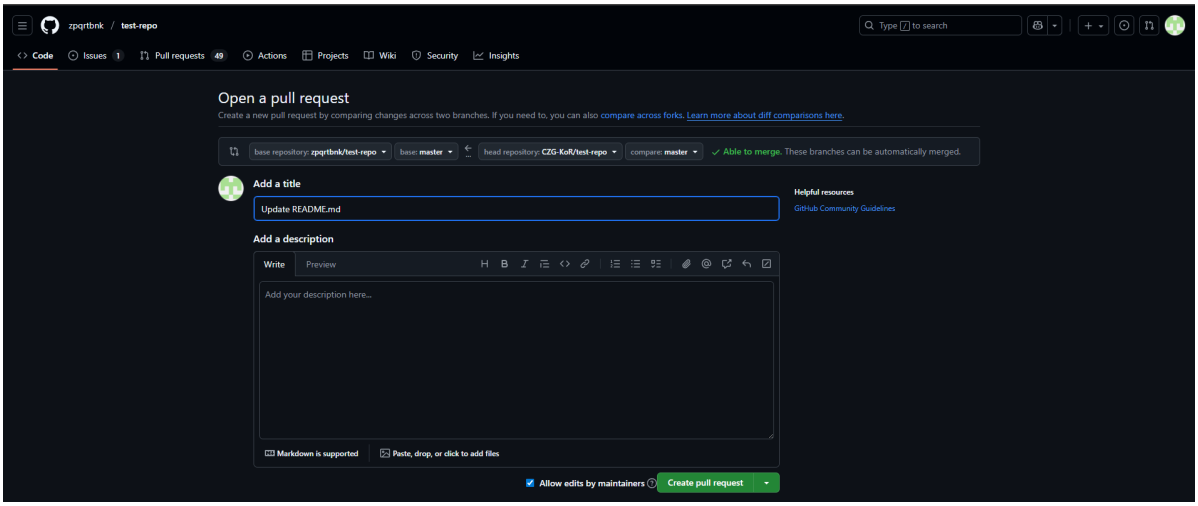

Abbildung 1.8: pull request 2

Beim pull request ist darauf zu achten, dass das richtige Repository und Branch ausgewählt ist. Es sollte eine ausführliche Beschreibung aller vorgenommenen Änderungen hinzugefügt werden. Mit dem Stellen des pull requests kann der Besitzer des Orignalrepository die beiden Datenbestände verschmelzen (merging).

2 NetBeans-Integration

Größere Projekte lassen sich im Codespace von Github nicht komfortabel programmieren. Deshalb ist es üblich sein Repository mit Entwicklungsumgebungen wie NetBeans zu verknüpfen.

2.1 Erzeugung eines Security-Tokens

Um NetBeans mit Git zu verbinden, benötigt es den Speicherort des Repositorys und ein Token.

Definition: Token

Ein Security-Token (einfach: Token) ist ein elektronischer Schlüssel zur Identifizierung und Authentifizierung von Benutzern.

Um ein Token zu erstellen, geht man auf der rechten Seite bei GitHub auf sein Icon und wählt Settings aus.

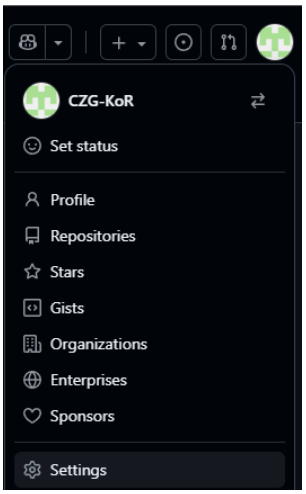

Abbildung 2.1: Token erstellen 1

Anschließend muss ganz nach unten gescrollt werden um zu den Developer Settings zu kommen. Dort wählt man Generate new token und wählt classic.

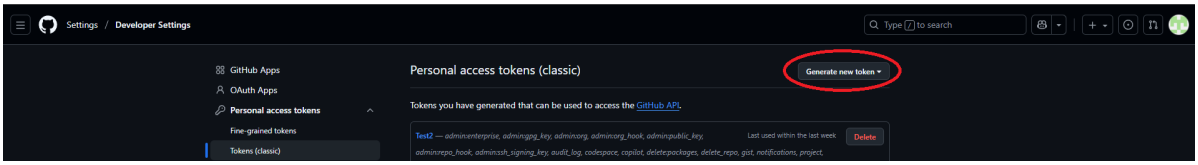

Abbildung 2.2: Token erstellen 2

Es kann eine Lebenszeit für den Token ausgesucht werden sowie seine Mächtigkeit. Im Regelfall reichen die repo-Optionen aus. Nach dem Drücken von Generate token könnt ihr den Schlüssel **einmalig** sehen und speichern.
Wichtig! Der Token ist wie ein Passwort zu behandeln.

2.2 Projekt in NetBeans clonen

Um das Projekt aus einem Repository zu clonen muss man NetBeans öffnen und in der Menüleiste Team auswählen, zu Git navigieren und Clone... anklicken.

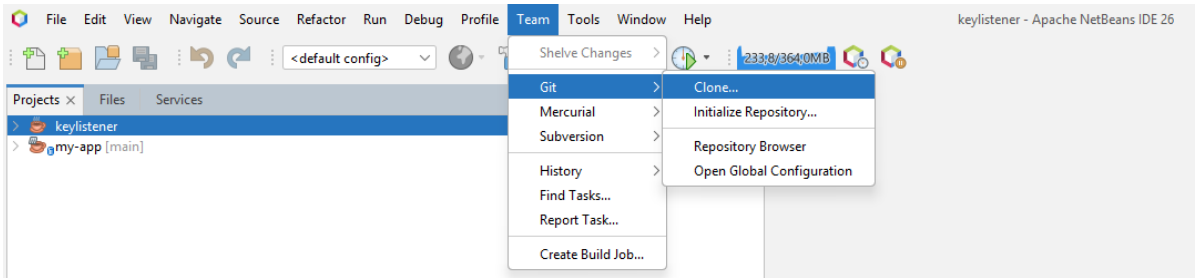

Abbildung 2.3: Projekt clonen 1

Es öffnet sich das zugehörige Fenster. Es müssen die zu kopierende Repository-URL angegeben werden, der GitHub-Nutzername und als Passwort der erstellte Token. Ein lokaler Ort für das Projekt kann darunter festgelegt werden.

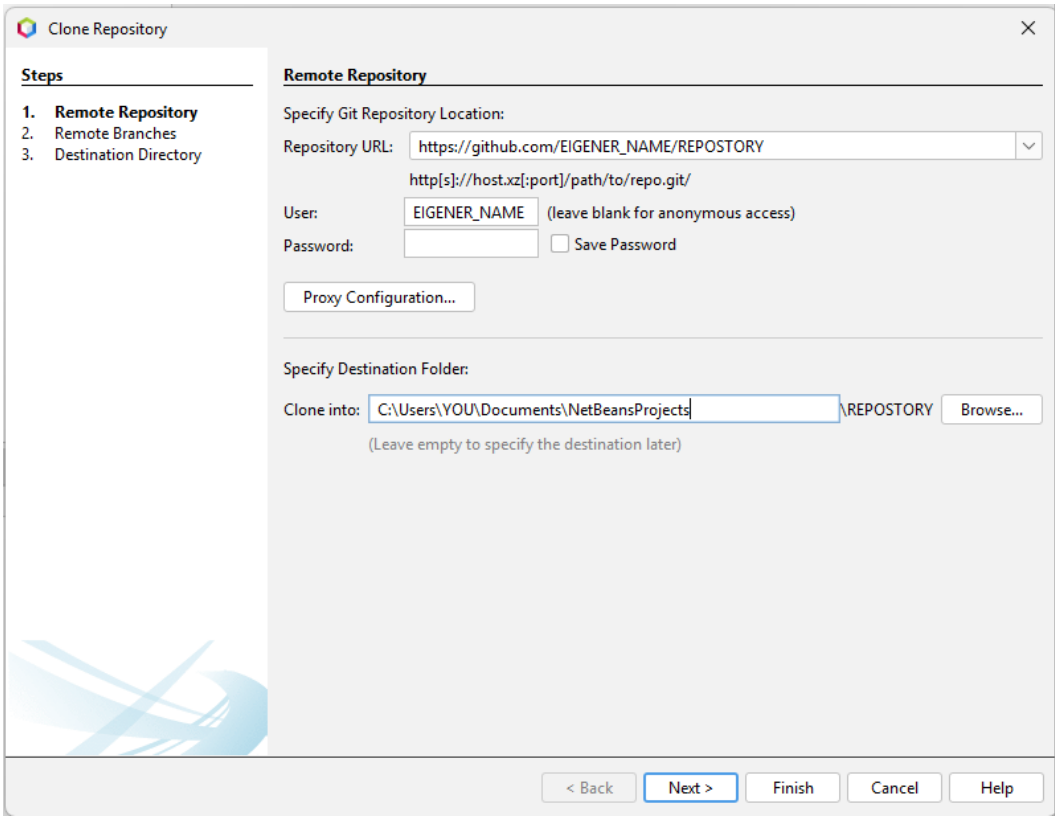

Abbildung 2.4: Projekt clonen 2

Nach Abschluss des Downloads öffnet sich das kopierte Projekt in NetBeans und kann bearbeitet werden.

2.3 Repository in NetBeans verwalten

Dateien und Verzeichnisse in denen Änderungen vorgenommen wurden werden farblich markiert. Wenn eine Änderung in einen Branch gemergt werden soll, muss zuerst der Commit vorgenommen werden. Dafür macht meinen einen Rechtsklick auf das Projekt, navigiert zu Git und wählt Commit... . Dort können die Beschreibung der Änderung und alle zu aktualisierenden Dateien ausgewählt werden und den Commit bestätigen.

Solange man allein in einem Repository arbeitet kann das Projekt direkt gepusht werden.

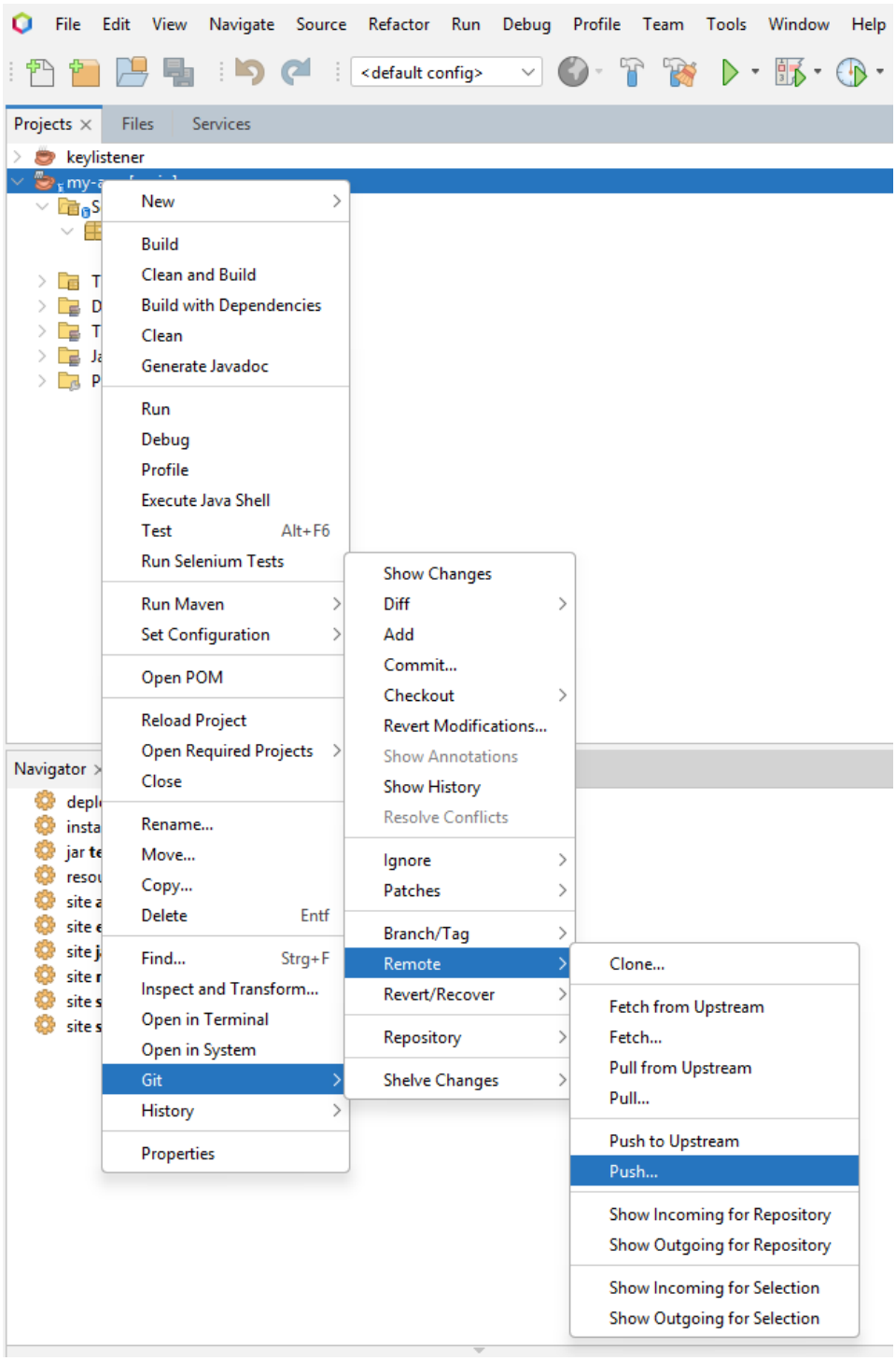

Abbildung 2.5: Projektverwaltung

Im sich öffnenden Fenster sollten die Daten vom Clone... übernommen worden sein. Wenn nicht, müssen diese erneut angegeben werden. Anschließend kann der Branch der gemergt werden soll ausgewählt werden und mit Finish bestätigt.

Arbeiten in einem Repository mehrere Personen sollte vor dem Push noch ein Pull durchgeführt werden, um Konfliktprobleme bei unterschiedlichen Versionsständen des Branches zuerst zu lösen.

Verwechslungsgefahr besteht durch Push to bzw. Pull from Upstream. Das würde versuchen die entsprechende Operation auf dem Ursprungsrepository durchführen.

NetBeans unterstützt noch eine große Menge weiterer Git-Operationen. Diese sollten ggf. erst an einem Testrepository ausprobiert werden.