

Anleitung zur Installation von Git, Installation von Java, Anmeldung zu GitLab der Hochschule, Verwaltung eines Kleinprojektes, Nutzung von Microsoft-Teams der Hochschule

Hochschule Reutlingen Fakultät Informatik

Martin Lauterbach

Die folgenden Seiten enthalten Informationen, die von Studierenden genutzt werden können, wenn Sie noch keine oder wenige Berührungspunkte mit Git und/oder Teams hatten, und Sie sich einem alleinigen Einstieg durch Online-Quellen noch nicht gewappnet fühlen; oder diese nicht schlüssig waren.

<https://youtube.com/shorts/rLxjsk3NAKM>

Die einzelnen Informationen sind jeweils mit Quellen (URLs) hinterlegt. Bitte beachten Sie, dass die hier dargelegten Links zur Zeit des Schreibens aktiv waren, diese aber jederzeit unbrauchbar werden können.

Dieses Dokument ist für den internen Gebrauch der Hochschule und der privaten Nutzung des Studierenden gemacht.

Dem Werk ist eine Anleitung für die Installation von Java beigelegt.

Stand: 23. Juni 2025

Inhaltsverzeichnis

1	Über Git	1
2	Zur Installation	2
2.1	Windows	3
2.2	MacOS	3
2.3	Linux	3
3	Zusatz: Java-Installation	4
3.1	Windows	4
3.2	MacOS	4
3.3	Linux	4
4	Zur Nutzung des Hochschul-GitLabs	5
4.1	Registrierung	5
4.2	Team-Erstellung	5
4.3	Repository-Erstellung	6
4.4	Mitgliedereinladung und Rechte	7
5	Nutzung eines Repositories als Kleinteam	8
5.1	Git-Grundfunktion: Feature-Branch-Workflow	8
5.2	Cloning: Die Repository vom Host herunterladen	9
5.2.1	Wichtiger Hinweis: Anmeldung mit git credentials	9
5.2.2	Manuell	10
5.2.3	IntelliJ Idea	11
5.2.4	Visual Studio Code	12
5.3	Git-Ignore	13
5.4	Branches, Commits und Pushes	14
5.4.1	IntelliJ Idea	14
5.4.2	Visual Studio Code	15
5.5	Branch mit Main updaten	16
5.5.1	IntelliJ Idea	16
5.5.2	Visual Studio Code	16
5.6	Merge-Request	17
5.6.1	IntelliJ Idea	17
5.6.2	Visual Studio Code	17
5.6.3	GitLab via Webbrowser	18
6	Anmeldung und Nutzung von Microsoft Teams	19
6.1	Beitritt zu einem privaten Team via Code	19
6.2	Beitritt zu einem intern-öffentlichen Team	19

1 Über Git

Aus Erfahrung heraus kann es vorkommen, dass Studierende bei Gruppenarbeiten kein gemeinsames Versionskontrollenprogramm (wie Git) nutzen, wenn dies nicht explizit vom Prüfer vorgeschrieben ist. Bei solchen Gruppenarbeiten kommt es dann in der Regel dazu, dass ein einzelnes Gruppenmitglied Code per Text-Messenger zugesendet bekommt, und diese Code-Fetzen in ein einreichbares Produkt zusammenflicken muss. Dies ist nicht nur ineffizient, sondern auch fehleranfällig.

Die Alternative zu diesem unstrukturierten Ansatz ist die Nutzung von Git. Git ermöglicht die gemeinsame Bearbeitung eines Projekts und bietet zahlreiche Vorteile:

1. Schnellerer Zugriff auf den Projektcode: Alle Teammitglieder können jederzeit auf die aktuellste Version des Codes zugreifen.
2. Kontrollierte Konfliktlösung: Überlappende Änderungen können effizient erkannt und gelöst werden.
3. Versionshistorie: Es ist möglich, zu früheren Versionen des Codes zurückzukehren, falls etwas schiefgeht.
4. Verantwortungs- und Rechteverwaltung: Aufgaben und Berechtigungen können gezielt verwaltet werden.

Moderne Repository-Hoster wie GitHub oder GitLab bieten darüber hinaus zusätzliche Funktionen wie Production-Pipelines, To-Do-Verwaltung und mehr, die den Entwicklungsprozess weiter optimieren können.

Dokumentation von IntelliJ IDEA bezüglich ihrer GIT-Integration:

<https://www.jetbrains.com/help/idea/using-git-integration.html> (29.09.24)

Dokumentation von Visual Studio Code bezüglich ihrer GIT-Integration:

<https://code.visualstudio.com/docs/editor/versioncontrol> (29.09.24)

Anhand einer Baum-Metapher soll hier kurz das Grundlegende Prinzip von Git erläutert werden.

Git als Baum hat die Fähigkeit, sich selbst erneut zu verbinden und gleichzeitig an mehreren Stellen zu wachsen. Der Hauptstamm dieses Baumes repräsentiert den Hauptbranch (oft als “main” oder “master” bezeichnet), welcher im Besten Fall die stabile und konsolidierte Version des Projekts darstellen soll.

Von diesem Hauptstamm ausgehend, entstehen verschiedene Äste (Branches), die unterschiedliche Entwicklungszweige oder Funktionalitäten des Projekts repräsentieren. Diese Äste können unabhängig voneinander wachsen und sich weiter verzweigen, was die parallele Entwicklung mehrerer Funktionen oder Features ermöglicht. Jeder dieser Äste kann als eine isolierte Entwicklungsumgebung betrachtet werden, in der Änderungen vorgenommen werden, ohne die Stabilität des Hauptstamms zu beeinträchtigen.

Ein wesentliches Merkmal dieses Baumes ist die Fähigkeit zur Rekonnektion. Sobald ein Entwicklungszweig (Branch) eine stabile und getestete Funktionalität erreicht hat, kann er wieder mit dem Hauptstamm verbunden werden. Dieser Prozess, bekannt als Merging, integriert die Änderungen des Zweiges in den Hauptbranch und stellt sicher, dass die neuen Funktionen in die stabile Version des Projekts aufgenommen werden.

Darüber hinaus können mehrere Äste gleichzeitig wachsen und sich entwickeln, was die gleichzeitige Arbeit mehrerer Entwickler an verschiedenen Teilen des Projekts ermöglicht. Diese parallele Entwicklung wird durch die Struktur des Baumes unterstützt, der es erlaubt, dass verschiedene Zweige unabhängig voneinander existieren und sich entwickeln können.

2 Zur Installation

Für jedes Betriebssystem wird eine kurze Anleitung für die Installation von Git als Executable/Toolset gegeben.

Für alle Installer, sowie hier nicht näher eingegangene Installationsalternativen, die Ihnen helfen könnten, falls angegebene Varianten nicht funktionieren, können Sie diesem Link folgen (27/09/2024): <https://git-scm.com/downloads>

2.1 Windows

1. Herunterladen des Installers von der offiziellen Webseite:
<https://git-scm.com/downloads/win>¹ (27/09/2024)
2. Installer ausführen. *Use Git from the Windows Command Prompt* auswählen.

IntelliJ Idea:

1. Programm öffnen und navigieren zu: File > Settings > Version Control > Git
2. Wenn nicht automatisch vorhanden, muss hier der Pfad zum Git executable eingefügt werden. In der Regel ist das: `C:\Program Files\Git\bin\git.exe`
3. Wenn vorhanden, Test-Knopf drücken, um die Verbindung zu testen.

2.2 MacOS

1. Git ist normalerweise bereits vorinstalliert. Sie können das überprüfen, indem Sie ein Terminal öffnen und **git --version** eingeben
2. Wenn Sie keine positive Meldung erhalten, können Sie mit einer Eingabe von **brew install git** weitermachen. Wenn das nicht geht, folgen Sie diesem Link: (27/09/2024) <https://git-scm.com/downloads/mac>

IntelliJ Idea:

1. Programm öffnen und navigieren zu: File > Settings > Version Control > Git
2. Wenn nicht automatisch vorhanden, muss hier der Pfad zum Git executable eingefügt werden. In der Regel ist das: `/usr/local/bin/git`
3. Wenn vorhanden, Test-Knopf drücken, um die Verbindung zu testen.

2.3 Linux

Debian-basiert (z.B.: Ubuntu)

1. package list updaten: `sudo apt update`
2. git installieren: `sudo apt install git`
3. installation verifizieren: `git --version`

RPM-basiert (z.B.: Fedora)

1. git installieren: `sudo dnf install git`
2. installation verifizieren: `git --version`

¹In der Regel besitzt Ihr Computer eine x64-Architektur. (27/09/2024)

3 Zusatz: Java-Installation

Es kann vorkommen, dass ein Installer Java nicht zur Path-Systemvariable hinzugefügt hat, was Ihnen in IDEs Probleme machen kann. Bei Problemen, überprüfen Sie: <https://www.java.com/de/download/help/path.html>

3.1 Windows

1. Java-Download-Seite öffnen: Gehe zur [Java-Download-Seite](#).
2. Java herunterladen: Klicke auf den Download-Button und lade die Installationsdatei herunter.
3. Installation starten: Doppelklicke auf die heruntergeladene Datei und folge den Anweisungen des Installationsassistenten.
4. Installation überprüfen: Öffne die Eingabeaufforderung und gib `java -version` ein, um sicherzustellen, dass Java korrekt installiert ist.

3.2 MacOS

1. Java-Download-Seite öffnen: Gehe zur [Java-Download-Seite](#).
2. Java herunterladen: Lade die DMG-Datei für macOS herunter.
3. Installation starten: Doppelklicke auf die DMG-Datei und folge den Anweisungen des Installationsassistenten.
4. JAVA_HOME setzen: Öffne das Terminal und füge folgende Zeilen zu deiner `.zshrc` oder `.bash_profile` hinzu:

```
export JAVA_HOME=$(/usr/libexec/java_home)
export PATH=$JAVA_HOME/bin:$PATH
```
5. Installation überprüfen: `java -version`

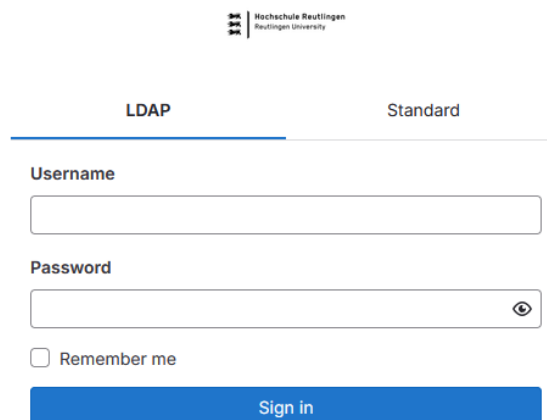
3.3 Linux

1. Paketliste aktualisieren: `sudo apt update`
2. Java installieren: `sudo apt install default-jdk`
3. Installation überprüfen: `java -version`

4 Zur Nutzung des Hochschul-GitLabs

Die Hochschule betreibt eine eigene GitLab-Instanz. So kann zwischen Personen mit Hochschulkonto gemeinsam geteilt und gearbeitet werden, und entwickelter Code, und somit Prüfungsleistung, bleibt für die Zeit der Entwicklung hochschulintern.

4.1 Registrierung



The image shows the login interface of the Hochschule Reutlingen GitLab instance. At the top, the university's logo and name are displayed. Below this, there are two tabs: 'LDAP' (which is selected and highlighted with a blue underline) and 'Standard'. The 'LDAP' tab contains a 'Username' label above a text input field, a 'Password' label above a password input field with a toggle eye icon, and a 'Remember me' checkbox. At the bottom of the form is a blue 'Sign in' button.

Abbildung 1: Nachdem Sie <https://gitlab.reutlingen-university.de/> besucht haben, melden sie sich mit Ihrem Hochschul-**Benutzernamen** und Ihrem Passwort an.

4.2 Team-Erstellung

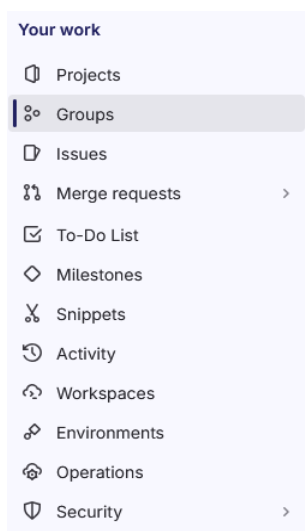


Abbildung 2: Wählen Sie im linken Reiter Groups/-Gruppen aus.

Groups

Explore groups [New group](#)

Created date

Abbildung 3: Wählen Sie nun aus, über New Group / Neue Gruppe, ein Team zusammenzustellen. Im folgenden Fenster wählen Sie Create Group/Gruppe erstellen.

Group name

Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.

Group URL

Visibility level
Who will be able to see this group? [View the documentation](#)

☒ **Private**
The group and its projects can only be viewed by members.

☐ **Internal**
The group and any internal projects can be viewed by any logged in user except external users.

☐ **Public**
The group and any public projects can be viewed without any authentication.

Now, personalize your GitLab experience
We'll use this to help surface the right features and information to you.

Role

Who will be using this group?
☒ My company or team ☐ Just me

What will you use this group for?

Invite Members (optional)
Invited users will be added with developer level permissions. [View the documentation](#) to see how to change this later.

Email 1

Abbildung 4: Erstellen Sie einen Teamnamen und füllen Sie wenn möglich den Rest der Felder aus. Sie können Ihre Teammitglieder einladen.

In manchen Fällen ist es eine Vorgabe von Modulen/Prüfungsleuten, ihre Tutoren und Dozenten über deren Hochschulemailadressen gleichermaßen hinzuzufügen. Dies können Sie aber auf Repositories mit Prüfungsleistungen begrenzen.

4.3 Repository-Erstellung

Group DieBeispielGruppe was successfully created.

DieBeispielGruppe

Recent activity: Last 30 days | Merge requests created: 0 | Issues created: 0 | Members added: 1

Subgroups and projects | Shared projects | Inactive

Name

Create new subgroup
Groups are the best way to manage multiple projects and members.

Create new project
Projects are where you can store your code, access issues, wiki, and other features of GitLab.

Abbildung 5: Über die Auswahl von New Project / Neues Projekt kann eine neue Repository erstellt werden. Im Folgenden Fenster arbeiten Sie in der Regel mit einem leeren/blank project.



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL

Project slug

Visibility Level

☒ Private

Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Internal

The project can be accessed by any logged in user except external users.

☐ Public

The project can be accessed without any authentication.

Project Configuration

☒ Initialize repository with a README

Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)

Analyze your source code for known security vulnerabilities. [Learn more.](#)

Abbildung 6: Hier können Sie ihrem Projekt/der Repository einen Namen geben. Dieser ist in der Regel dann auch Teil der URL, wenn nicht anders ausgewählt.

4.4 Mitgliedereinladung und Rechte

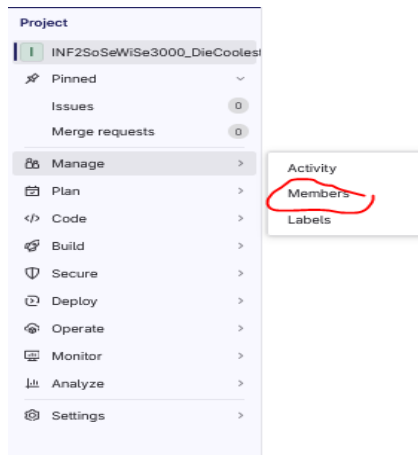


Abbildung 7: Wählen Sie im linken Reiter ihres Projekts/Repositories unter Manage -> Members aus. Hier können Sie unter Invite Members / Mitglieder per Hochschuladresse einladen. Denken Sie daran, Prüfer und Tutoren in der Regel mit mindestens Maintainer einzuladen

5 Nutzung eines Repositories als Kleinteam

Für Projekte mit Java wird zur Zeit in der Regel das JetBrains-Produkt IntelliJ Idea sowie das Microsoft-Produkt Visual Studio Code genutzt. Diese kommen mit einer eingebauten GUI für Git-Funktionen. Es wird jeweils auf die Funktionalitäten eingegangen, und dann für den derzeitigen Stand (23. Juni 2025) in jeweiligen IDEs erklärt.

Sie installieren die IDEs in der Regel durch einen Installer. Folgen Sie für Weiteres den Anweisungen auf:

IntelliJ: <https://www.jetbrains.com/idea/> (28.09.24)

Visual Studio Code: <https://code.visualstudio.com/> (28.09.24)

5.1 Git-Grundfunktion: Feature-Branch-Workflow

Ein klar definierter Workflow ist entscheidend, damit Sie in einem Team den Überblick behalten können.

Ein häufig verwendeter Ansatz ist der “Feature-Branch-Workflow”. Dabei erstellt jedes Teammitglied für jede neue Funktion oder Änderung einen eigenen Branch vom Haupt- oder Entwicklungsbranch. Oft wird dieser *master* oder *main* genannt. Nach Abschluss der Arbeit wird ein Pull-Request (oder Merge-Request) erstellt, der von mindestens einem anderen Teammitglied überprüft werden sollte, bevor die Änderungen in den Hauptbranch integriert werden. Dabei wird der Branch mit der neuen Funktion vereint.

Regelmäßige Commits mit aussagekräftigen Nachrichten sind ebenfalls wichtig. Diese sollten beschreiben, was geändert wurde und warum, um die Nachvollziehbarkeit zu gewährleisten. Es ist auch ratsam, kleinere, häufigere Commits zu machen, anstatt große, unübersichtliche Änderungen auf einmal vorzunehmen.

Ein weiterer wichtiger Aspekt ist das regelmäßige Synchronisieren mit dem Hauptbranch, um Merge-Konflikte zu minimieren. Dies kann durch häufiges Pullen der neuesten Änderungen aus dem Hauptbranch erreicht werden. Dies können Sie aus ihrem Funktions-Branch heraus machen, ohne das Sie ihren Code simultan zum Main-Branch pushen.

Sie erstellen einen Branch -> Sie verändern Code -> Sie commiten dies -> Sie verändern Code -> ... -> Sie pushen dies in Ihren Branch online -> Sie updaten Ihren Branch mit Main -> ... -> Sie stellen eine Merge-Request ihres Branches rein in main.

5.2 Cloning: Die Repository vom Hoster runterladen

Um damit beginnen zu können, auf Ihrem Gerät an einem Projekt zu arbeiten, müssen sie dieses *clonen*. Das bedeutet, Sie müssen den Code, die Repository, von dem Server, der dieses hostet, herunterladen können.

5.2.1 Wichtiger Hinweis: Anmeldung mit git credentials

Um das arbeiten mit GIT so einfach wie möglich zu machen, nutzt JetBrains (das bedeutet IntelliJ und für spätere Semester auch mal WebStorm) eine geführte Initialisierung einer Repository. Das bedeutet, dass wenn Sie über IntelliJ eine Repository clonen, sie über Pop-Ups und vorgefertigte Weiterleitungen erklärt bekommen, was sie tun müssen, um sich mit Ihren GitLab-Credentials zu authentifizieren.

Unterschieden wird dabei zwischen einer Authentifizierungen mit Username & Passwort und mit einem Token.

Um sich von Ihrem System aus dem Hochschulgitlab-Server zu authentifizieren können Sie nur einen Token nutzen, da dieser den Vorteil gegenüber einem Username/Passwort approach bringt, dass nur bestimmte Rechte und Vorgänge zugelassen werden können für einen bestimmten Token, welche Sie selbst auswählen können.

In IntelliJ wird Ihnen im Pop-Up-Fenster ein Knopf mit "Generate Token" präsentiert. Wenn Sie diesen klicken, werden sie automatisch über einen Browseraufruf auf den Hochschulserver geleitet, auf einen Bildschirm, an dem die nötigen Rechte bereits ausgewählt sind.

Nötig für eine IDE (oder Ihrer lokalen Git Toolbox), um zu clonen, Branches zu erstellen, zu pushen und merge Requests zu erstellen, sind die Rechte **api** und **read_repository**.

Sie können manuell in der GitLab-Oberfläche über Profile (der Kreis mit Ihrem Profilbild), Preferences und Token (im linken Reiter) einen Token erstellen.

Nach Erstellung können Sie den Token kopieren und in die Authentifizierungsbox pasten.

Das Anmeldefenster für die lokale Git Installation für GitLab kann zwischen Passwort und Token wechseln.

5.2.2 Manuell

1. Öffnen Sie ihre Kommandozeile/Terminal
2. Wählen Sie den Ort aus, an dem Sie später ihre Repository-Daten halten möchten, via: `cd Weg/zu/Ihrem/Ordner`
3. nutzen Sie diesen Befehl, und tauschen Sie dabei den Link mit dem Link zu Ihrem Projekt: `git clone https://github.com/username/repository.git`

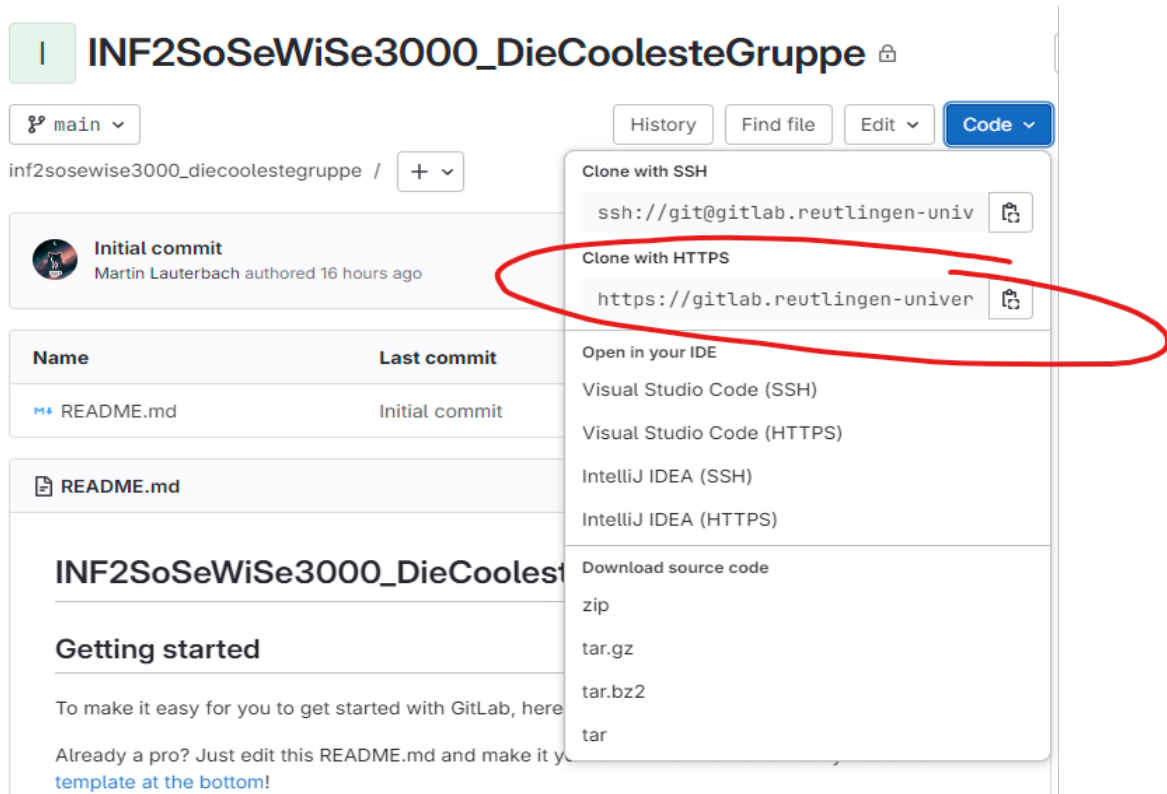


Abbildung 8: Auf der Projektseite in GitLab können Sie einen Link erhalten, indem Sie auf **Code** drücken und den Link unter dem Protokoll HTTP kopieren.

5.2.3 IntelliJ Idea

1. Öffnen Sie IntelliJ IDEA.
2. Navigieren Sie zur Option *Get from Version Control*:
 - Wenn Sie den Begrüßungsbildschirm sehen, klicken Sie auf **Get from Version Control** / **Get from VCS** (oben rechts).
 - Wenn Sie ein Projekt geöffnet haben, gehen Sie zu **Datei** > **Neu** > **Projekt aus Versionskontrolle**.
3. Wählen Sie Git:
 - Im erscheinenden Dialog wählen Sie **Git** aus der Liste auf der linken Seite.
4. Geben Sie die Repository-URL ein:
 - Kopieren Sie die URL des Repositories, das Sie klonen möchten (z.B. von GitHub).
 - Fügen Sie die URL in das *URL-Feld* in IntelliJ IDEA ein.
5. Wählen Sie das Verzeichnis:
 - Wählen Sie das Verzeichnis aus, in das Sie das Repository klonen möchten.
6. Klicken Sie auf *Klonen*:
 - IntelliJ IDEA wird das Repository in das ausgewählte Verzeichnis klonen und es als neues Projekt öffnen.

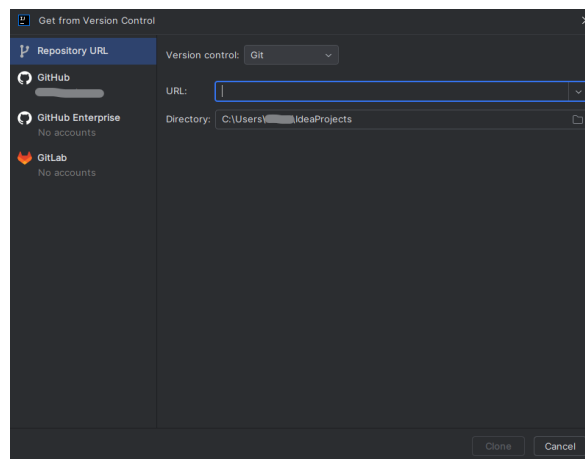


Abbildung 9: Wenn Sie auf der Begrüßungsseite "Get from VCS"(englisch) ausgewählt haben, erscheint dieser Bildschirm. Hier können Sie den Link einfügen und den Ort, an dem die Daten gelegt werden sollen.

5.2.4 Visual Studio Code

1. Öffnen Sie Visual Studio Code.

2. Navigieren Sie zur Option *Source Control*:

- Klicken Sie auf das Symbol **Source Control** in der Seitenleiste (oder verwenden Sie die Tastenkombination **Strg+Shift+G** auf Windows bzw. **Cmd+Shift+G** auf Mac).

3. Wählen Sie **Repository klonen**:

- Klicken Sie auf die Schaltfläche **Clone Repository** im Source Control-Bereich.

4. Geben Sie die **Repository-URL** ein:

- Kopieren Sie die URL des Repositories, das Sie klonen möchten.
- Fügen Sie die URL in das entsprechende Feld ein und drücken Sie **Enter**, um den Klonvorgang zu starten.

5. Wählen Sie das **Verzeichnis**:

- Wählen Sie das Verzeichnis aus, in das Sie das Repository klonen möchten.

6. Öffnen Sie das **geklonte Repository**:

- Nach Abschluss des Klonvorgangs wird Visual Studio Code das geklonte Repository öffnen und Sie können mit der Arbeit beginnen.

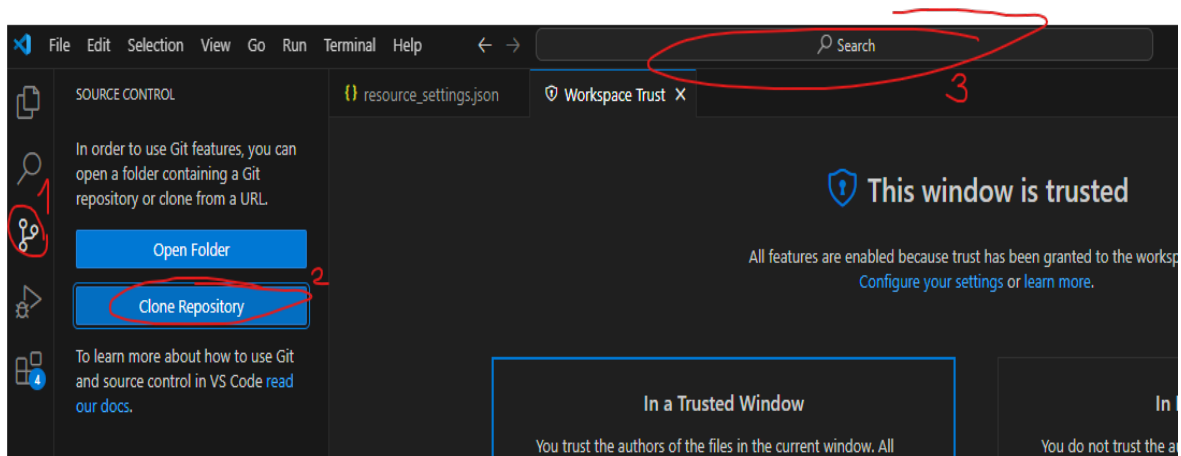


Abbildung 10:

1. Icon des Source Controls
2. Knopf für das Klonen
3. Eingabe-Zeile, die sich dann für das Eintragen der URL öffnet.

5.3 Git-Ignore

Eine .gitignore Datei kann von Git im Root ihres Projektverzeichnis genutzt werden, um bestimmte Dateien von zukünftigen Commits auszuschließen. Das ist besonders nützlich, um unnötige oder sensible Dateien, wie z.B. Konfigurationsdateien, temporäre Dateien oder Build-Artefakte, nicht in das Repository aufzunehmen. Dadurch bleibt das Repository sauber und übersichtlich, und es werden keine vertraulichen Informationen versehentlich geteilt.

Ein neues .gitignore-File ist in der Regel so gut wie leer. Folgende Einträge sollten Sie einer bestehenden oder von Ihnen erstellten Datei hinzufügen:

1. Log-Dateien:

- *.log

2. Systemdateien:

- .DS_Store

3. Build-Artefakte:

- /build/

4. Abhängigkeiten:

- /node_modules/

5. Sensible Daten:

- .env

6. Temporäre Dateien:

- *.tmp

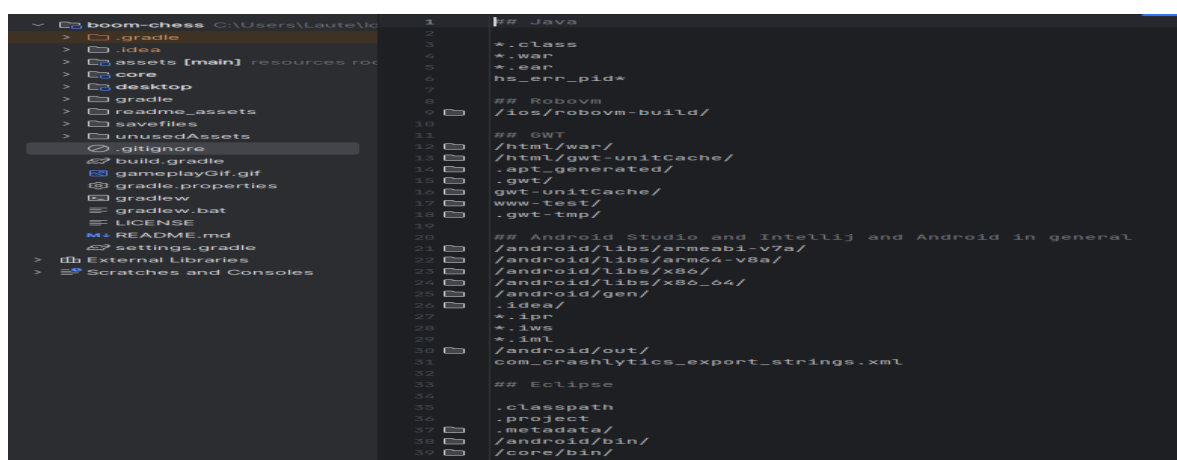


Abbildung 11: So sieht die Datei aus, wenn Sie für ein großes Projekt konfiguriert wurde.

5.4 Branches, Commits und Pushes

5.4.1 IntelliJ Idea

BRANCHES

1. Öffnen Sie Ihr Projekt in IntelliJ IDEA.
2. Öffnen Sie das Git-Tool-Fenster:
 - Klicken Sie auf das Git-Symbol in der unteren rechten Ecke des Fensters.
3. Erstellen Sie einen neuen Branch:
 - Klicken Sie auf den aktuellen Branch-Namen und wählen Sie **New Branch**.
 - Geben Sie einen Namen für den neuen Branch ein und klicken Sie auf **Create**.

COMMITTS

1. Wählen Sie die Dateien aus, die Sie committen möchten:
 - Öffnen Sie das **Commit-Tool-Fenster** auf der linken Seite.
 - Wählen Sie die entsprechenden Dateien oder die gesamte Changelist aus.
2. Geben Sie eine Commit-Nachricht ein:
 - Schreiben Sie eine Nachricht, die die Änderungen beschreibt.
3. Committen Sie die Änderungen:
 - Klicken Sie auf **Commit** oder **Commit and Push**.

PUSHES

1. Öffnen Sie das Push-Dialogfeld:
 - Gehen Sie zu **VCS > Git > Push** oder verwenden Sie die Tastenkombination **Strg+Shift+K** (Windows) bzw. **Cmd+Shift+K** (Mac).
2. Überprüfen Sie die Commits:
 - Überprüfen Sie die Commits, die gepusht werden sollen.
3. Pushen Sie die Änderungen:
 - Klicken Sie auf **Push**.

5.4.2 Visual Studio Code

BRANCHES

1. Öffnen Sie Ihr Projekt in Visual Studio Code.
2. Navigieren Sie zur Source Control Ansicht:
 - Klicken Sie auf das Source Control Symbol in der Seitenleiste (oder verwenden Sie die Tastenkombination **Strg+Shift+G** auf Windows bzw. **Cmd+Shift+G** auf Mac).
3. Erstellen Sie einen neuen Branch:
 - Klicken Sie auf den aktuellen Branch-Namen in der unteren linken Ecke und wählen Sie **Create Branch**.
 - Geben Sie einen Namen für den neuen Branch ein und drücken Sie **Enter**.

COMMITTS

1. Wählen Sie die Dateien aus, die Sie committen möchten:
 - Öffnen Sie die Source Control Ansicht und wählen Sie die entsprechenden Dateien aus.
2. Geben Sie eine Commit-Nachricht ein:
 - Schreiben Sie eine Nachricht, die die Änderungen beschreibt, in das obere Textfeld.
3. Committen Sie die Änderungen:
 - Klicken Sie auf das Häkchen-Symbol oder drücken Sie **Strg+Enter** (Windows) bzw. **Cmd+Enter** (Mac).

PUSHES

1. Öffnen Sie das Push-Dialogfeld:
 - Klicken Sie auf die drei Punkte in der Source Control Ansicht und wählen Sie **Push**.
2. Überprüfen Sie die Commits:
 - Überprüfen Sie die Commits, die gepusht werden sollen.
3. Pushen Sie die Änderungen:
 - Klicken Sie auf **Push**, um die Änderungen an das Remote-Repository zu senden.

5.5 Branch mit Main updaten

Dieser Vorgang aktualisiert Ihren Feature-Branch mit den neuesten Änderungen aus main. Der main-Branch bleibt dabei unverändert. Durch regelmäßiges Updaten Ihres Branches mit **main** können Sie Merge-Konflikte minimieren, die auftreten können, wenn der Branch eines anderen Entwicklers mittlerweile in den main-branch gemerged wurde.

5.5.1 IntelliJ Idea

1. Navigieren Sie zur Git-Ansicht:

- Klicken Sie auf das Git-Symbol in der unteren rechten Ecke oder verwenden Sie die Tastenkombination **Alt+9**.

2. Holen Sie die neuesten Änderungen vom Remote-Repository:

- Klicken Sie auf **VCS** im oberen Menü, wählen Sie **Git** und dann **Pull**.

3. Mergen Sie die Änderungen in Ihren Branch:

- Klicken Sie auf **VCS**, wählen Sie **Git** und dann **Merge Changes**.
- Wählen Sie **main** (oder den entsprechenden Branch) und klicken Sie auf **Merge**.

5.5.2 Visual Studio Code

1. Navigieren Sie zur Source Control Ansicht:

- Klicken Sie auf das Source Control Symbol in der Seitenleiste (oder verwenden Sie die Tastenkombination **Strg+Shift+G** auf Windows bzw. **Cmd+Shift+G** auf Mac).

2. Holen Sie die neuesten Änderungen vom Remote-Repository:

- Klicken Sie auf die drei Punkte in der Source Control Ansicht und wählen Sie **Pull**.

3. Mergen Sie die Änderungen in Ihren Branch:

- Klicken Sie auf den aktuellen Branch-Namen in der unteren linken Ecke und wählen Sie **Merge Branch**.
- Wählen Sie **main** (oder den entsprechenden Branch) und klicken Sie auf **Merge**.

5.6 Merge-Request

Wenn Sie das Ziel ihres Branches erfüllt haben - zum Beispiel: Hintergrund-Bild-System hinzufügen - können Sie diesen Branch in den Main-Branch mergen. Dazu stellen Sie einen *Request/Anfrage*.

Hier soll die Funktionalität via IntelliJ, VSC und per Webbrowser in GitLab gezeigt sein.

5.6.1 IntelliJ Idea

1. **Öffnen Sie Ihr Projekt in IntelliJ IDEA.**
2. **Navigieren Sie zur Git-Ansicht:**
 - Klicken Sie auf das Git-Symbol in der unteren rechten Ecke oder verwenden Sie die Tastenkombination **Alt+9**.
3. **Erstellen Sie einen Merge-Request:**
 - Klicken Sie auf **VCS** im oberen Menü, wählen Sie **Git** und dann **Create Pull Request**.
 - Füllen Sie die erforderlichen Felder aus und klicken Sie auf **Create**.

5.6.2 Visual Studio Code

1. **Öffnen Sie Ihr Projekt in Visual Studio Code.**
2. **Navigieren Sie zur Source Control Ansicht:**
 - Klicken Sie auf das Source Control Symbol in der Seitenleiste (oder verwenden Sie die Tastenkombination **Strg+Shift+G** auf Windows bzw. **Cmd+Shift+G** auf Mac).
 - **Erstellen Sie einen Merge-Request:**
 - Klicken Sie auf die drei Punkte in der Source Control Ansicht und wählen Sie **Create Pull Request**.
 - Füllen Sie die erforderlichen Felder aus und klicken Sie auf **Create**.

5.6.3 GitLab via Webbrowser

1. Öffnen Sie Ihr Projekt in GitLab:

- Navigieren Sie zu Ihrem Projekt auf der GitLab-Webseite und melden Sie sich an, falls erforderlich.

2. Navigieren Sie zur Merge-Request-Seite:

- Klicken Sie im linken Seitenmenü auf **Merge Requests**.
- Klicken Sie auf die Schaltfläche **New merge request**.

3. Wählen Sie die Branches aus:

- Wählen Sie den Source-Branch (den Branch, den Sie mergen möchten) und den Target-Branch (in der Regel **main** oder **master**).
- Klicken Sie auf **Compare branches and continue**.

4. Füllen Sie die Merge-Request-Details aus:

- Geben Sie einen Titel und eine Beschreibung für den Merge-Request ein.
- Fügen Sie optional Labels, Meilensteine oder Assignees hinzu.

5. Erstellen Sie den Merge-Request:

- Klicken Sie auf **Submit merge request**, um den Merge-Request zu erstellen.

6. Im Besten Fall überprüfen Ihr Kleinteam und Sie gemeinsam den Code, und akzeptieren den Merge.

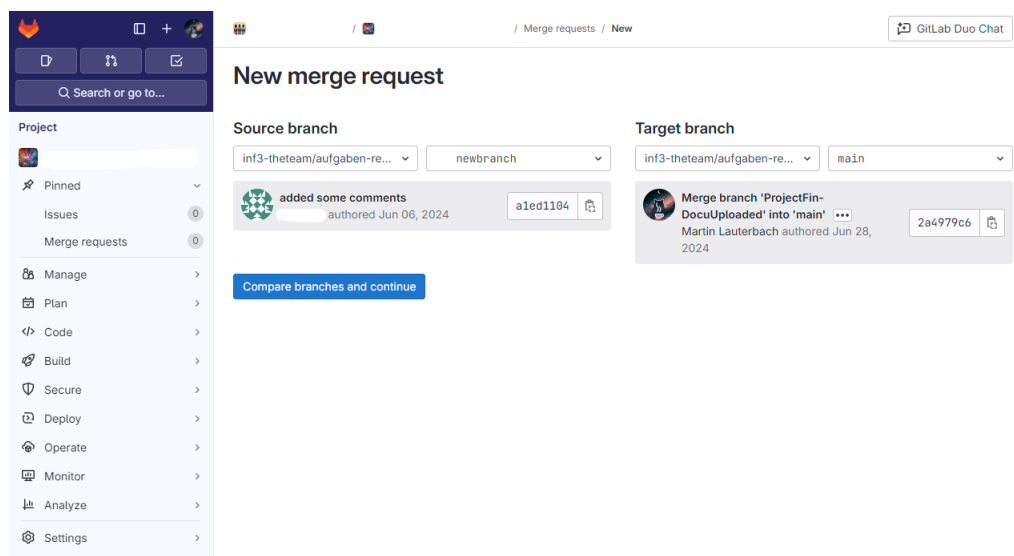


Abbildung 12: Beispiel eines Merge-Requests im GitLab der Hochschule

6 Anmeldung und Nutzung von Microsoft Teams

Microsoft Teams wird in vielen Modulen für die Text- und Videokommunikation, sowie für Meetings genutzt.

Hier wird für Sie kurz darauf eingegangen, wie Sie sich anmelden können, sowie Wie sie einem bereits erstellten öffentlichen und auf Einladung basiertem Team beitreten können.

Registrierung/Anmeldung

1. Öffnen Sie die Microsoft Teams-Anwendung oder besuchen Sie die Website <https://teams.microsoft.com> (28.07.24).
2. Klicken Sie auf **Anmelden**.
3. Geben Sie Ihre Hochschul-E-Mail-Adresse ein und klicken Sie auf **Weiter**.
4. Geben Sie Ihr Passwort ein und klicken Sie auf **Anmelden**.
5. Folgen Sie den Anweisungen zur Zwei-Faktor-Authentifizierung, falls diese aktiviert ist.

6.1 Beitritt zu einem privaten Team via Code

1. Melden Sie sich in Microsoft Teams an.
2. Klicken Sie auf der linken Seite auf **Teams**.
3. Wählen Sie **Team beitreten oder erstellen** aus.
4. Geben Sie den Teamcode, den Sie vom Team-Administrator erhalten haben, in das Feld **Code eingeben** ein.
5. Klicken Sie auf **Team beitreten**.

6.2 Beitritt zu einem intern-öffentlichen Team

1. Melden Sie sich in Microsoft Teams an.
2. Klicken Sie auf der linken Seite auf **Teams**.
3. Wählen Sie **Team beitreten oder erstellen** aus.
4. Durchsuchen Sie die Liste der verfügbaren Teams oder nutzen Sie die Suchfunktion, um das gewünschte Team zu finden.
5. Klicken Sie auf **Beitreten** neben dem Teamnamen.