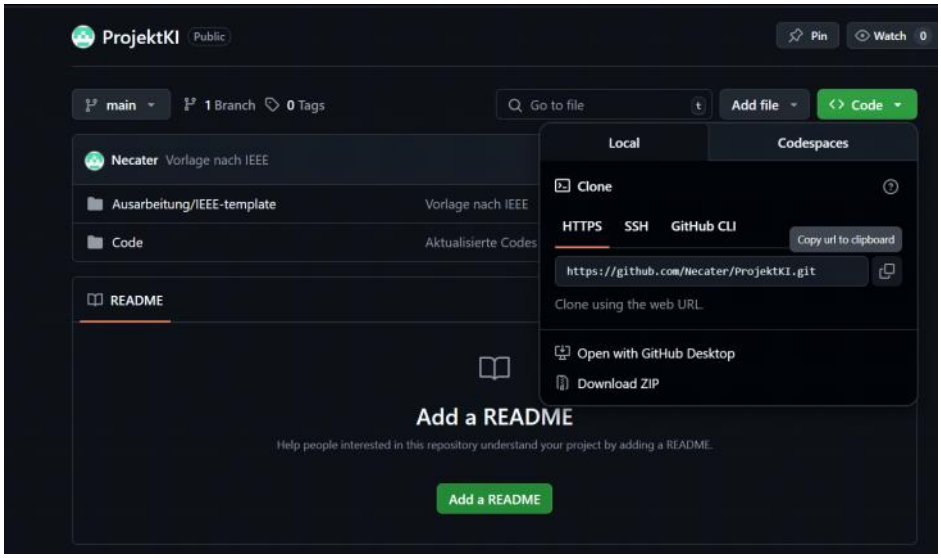
GitHub Anleitung

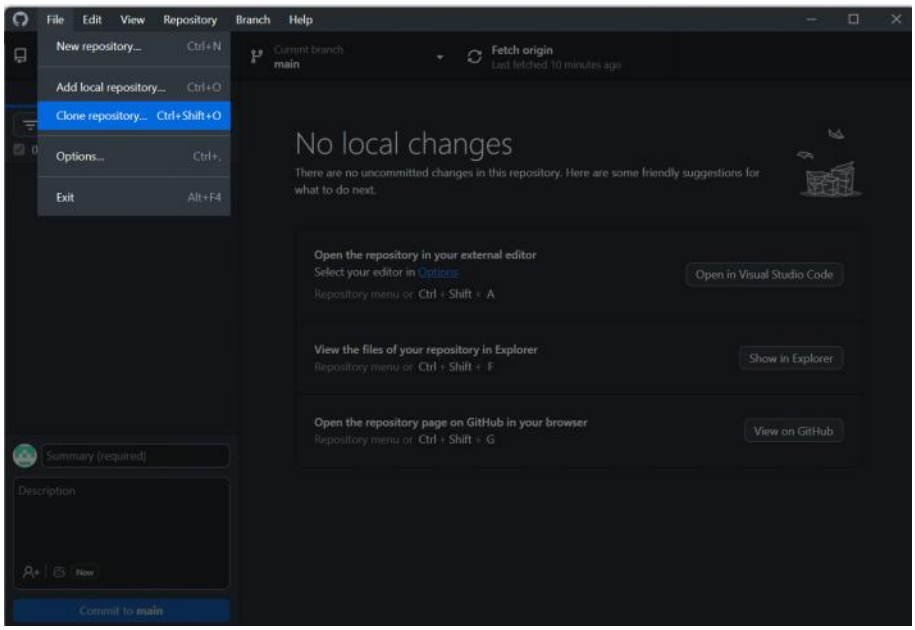
Mittwoch, 2. Juli 2025 09:56

1. GitHub Desktop herunterladen
   1.1 Alternativ: Auf dem Terminal (Git Bash) lässt sich auch arbeiten, aber für Einsteiger ist GitHub Desktop wahrscheinlich benutzerfreundlicher

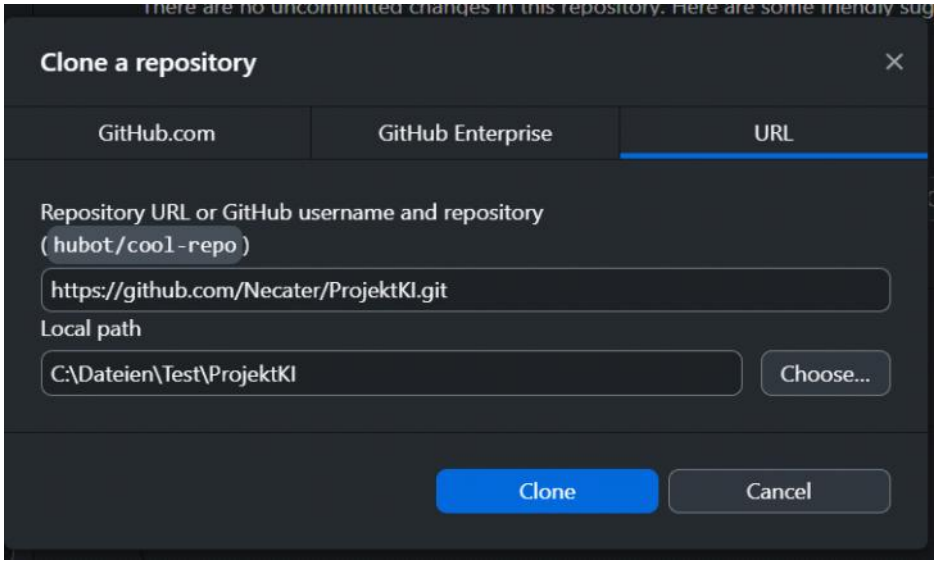
2. Repository clonen
   2.2 GitHub im Webbrowser öffnen
   2.3 Projekt öffnen
   2.4 Auf den grünen Code drücken
   2.5 HTTPS Link kopieren



   2.6 GitHub Desktop öffnen
   2.7 Auf File drücken
   2.8 Auf Clone Repository drücken



   2.9 Auf URL drücken
   2.10 Den HTTPS Link einfügen
   2.11 Gewünschten Pfad auswählen
   2.12 Auf Clonen drücken

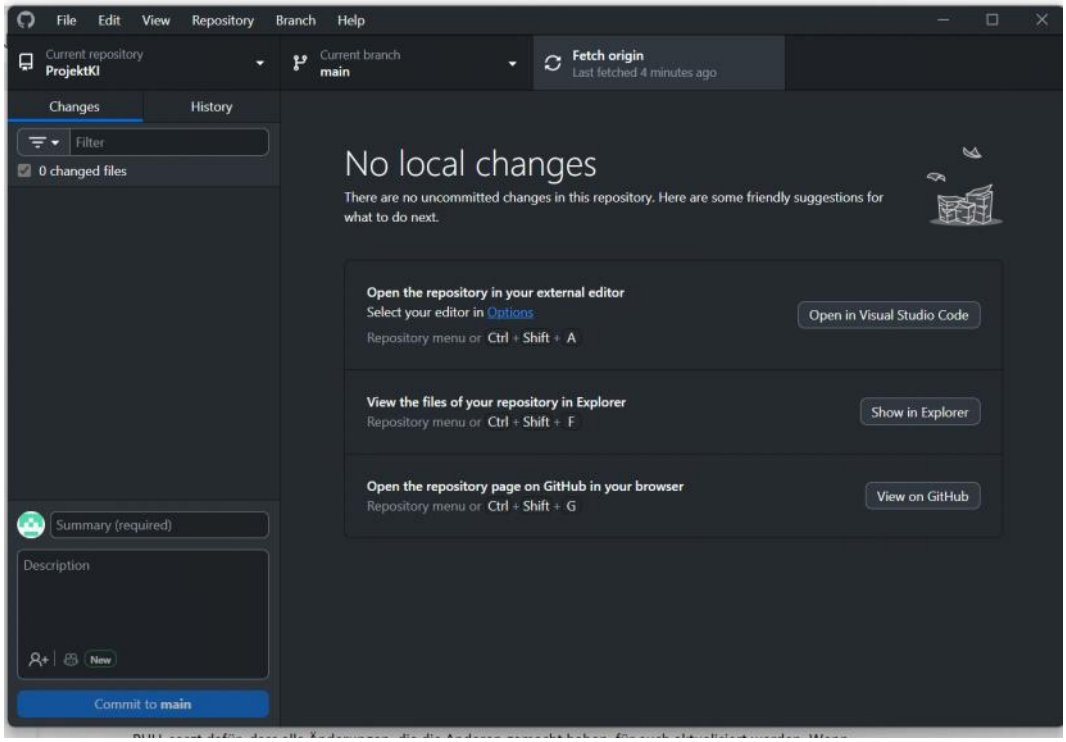


Super, Repo wurde hoffentlich gecloned!

3. Wichtige Dinge zu beachten
   3.1 Nachdem das Repo gecloned wurde, kann jetzt alles in dem Ordner wie gewohnt bearbeitet werden als wären die Sachen lokal auf dem PC.
   3.2 ABER damit ihr und alle anderen immer auf dem neusten Stand sind, muss immer Folgendes (am besten in dieser Reihenfolge) gemacht werden:
   PULL
   ADD
   COMMIT
   PUSH

   PULL sorgt dafür, dass alle Änderungen, die die Anderen gemacht haben, für euch aktualisiert werden. Wenn nichts gepulled wird, könnt ihr die neusten Änderungen nicht sehen.

   Bei GitHub Desktop drückt ihr einfach dafür auf "Fetch origin" und dann müsste da "Pull" stehen. Ihr müsstet dann nochmal auf "Pull" klicken und die Änderungen sollten übernommen sein.



ADD fügt alle eure Änderungen lokal in den Stash (Ihr würdet diese Änderungen bei GitHub Desktop links bei Changes sehen) - Diese Änderungen sind quasi zwischengespeichert und können auch wieder verworfen werden (Mit Discard oder ihr drückt manuell mit Rechtsklick auf die Änderung, um sie zu verwerfen). Das ist wichtig, falls ihr irgendwas rückgängig machen wollt. Wenn ihr auf GitHub Desktop arbeitet, müsst ihr keinen Befehl für ADD machen, es passiert automatisch, wenn ihr irgendwas ändert.

COMMIT bestätigt jetzt quasi nun alle Änderungen im Stash und macht sie PUSH-bereit. Ihr müsst commiten zum Pushen. Aber wenn ihr commited habt, könnt ihr die Änderungen nicht mehr mit Discard so einfach verwerfen.
SEHR WICHTIG HIERBEI: Ihr müsst immer eine Commit-Nachricht schreiben! Die Nachricht schreibt ihr einfach in dieses Feld Summary (Die Description muss nicht ausgefüllt werden)

PUSH sorgt dafür, dass alles, was ihr commited habt, nun für die Anderen gepushed wird. Das heißt, alle Anderen, die auf PULL drücken, werden genau diese Änderungen auch erhalten. Alle Änderungen, die ihr gemacht habt, sind vor dem PUSH bis dahin für die Anderen nicht sichtbar.

4. MERGE Konflikte
   4.1 Welp, es gibt sowas wie Merge-Konflikte. Das kann passieren, wenn zwei Leute an derselben Datei arbeiten und beide das Gleiche in dieser Datei verändert haben (Z.B. jemand hat eine Methode gelöscht, während die andere Person die Methode nur bearbeitet hat). Git weiß dann nicht, welche Änderungen er übernehmen soll. Dann müsst ihr selber angeben, welche Änderung nun die Aktuellste sein soll.

   Meistens kann dies mit einem automatischen Merge behoben werden, ansonsten muss man individuell mal schauen. Am besten wäre es, wenn keine MERGE Konflikte auftreten oder wenn, die sehr einfach zu beheben sind :D Aber wenn ein Merge-Konflikt auftritt, dann wird einem auch so ein Fenster angezeigt, wo man die einzelnen Merge-Konflikte sieht (Sorry, ich hab grad keine Merge-Konflikte, um dies zu zeigen).

   Auf unsere Ausarbeitung bezogen
   Wenn mehrere Personen an einer einzigen .tex Datei arbeiten, dann kann ein MERGE Konflikt auftreten wegen der pdf. Das ist nicht weiter schlimm, das ist einfach zu beheben, indem die Person die pdf nochmal neu kompilieren lässt.

5. Es gibt auch sowas wie BRANCHES, aber das ist jetzt für unser Projekt nicht relevant, aber wenn ihr euch dafür interessiert, könnt ihr euch auch darüber noch informieren. Ein Branch ist quasi eine Abspaltung einer weiteren Version von einer bereits vorhandenen Version. Aber für unser Projekt brauchen wir das nicht, das wäre auch eher bisschen nervig, da man Branches immer mergen muss, um sie wieder zusammenzuführen.