

Praxislabor Digitale Geisteswissenschaften

Einführung in Git



git

Von Jason Long, CC BY 3.0

CC-BY 4.0

Gerrit Heim

g.heim@ub.uni-frankfurt.de

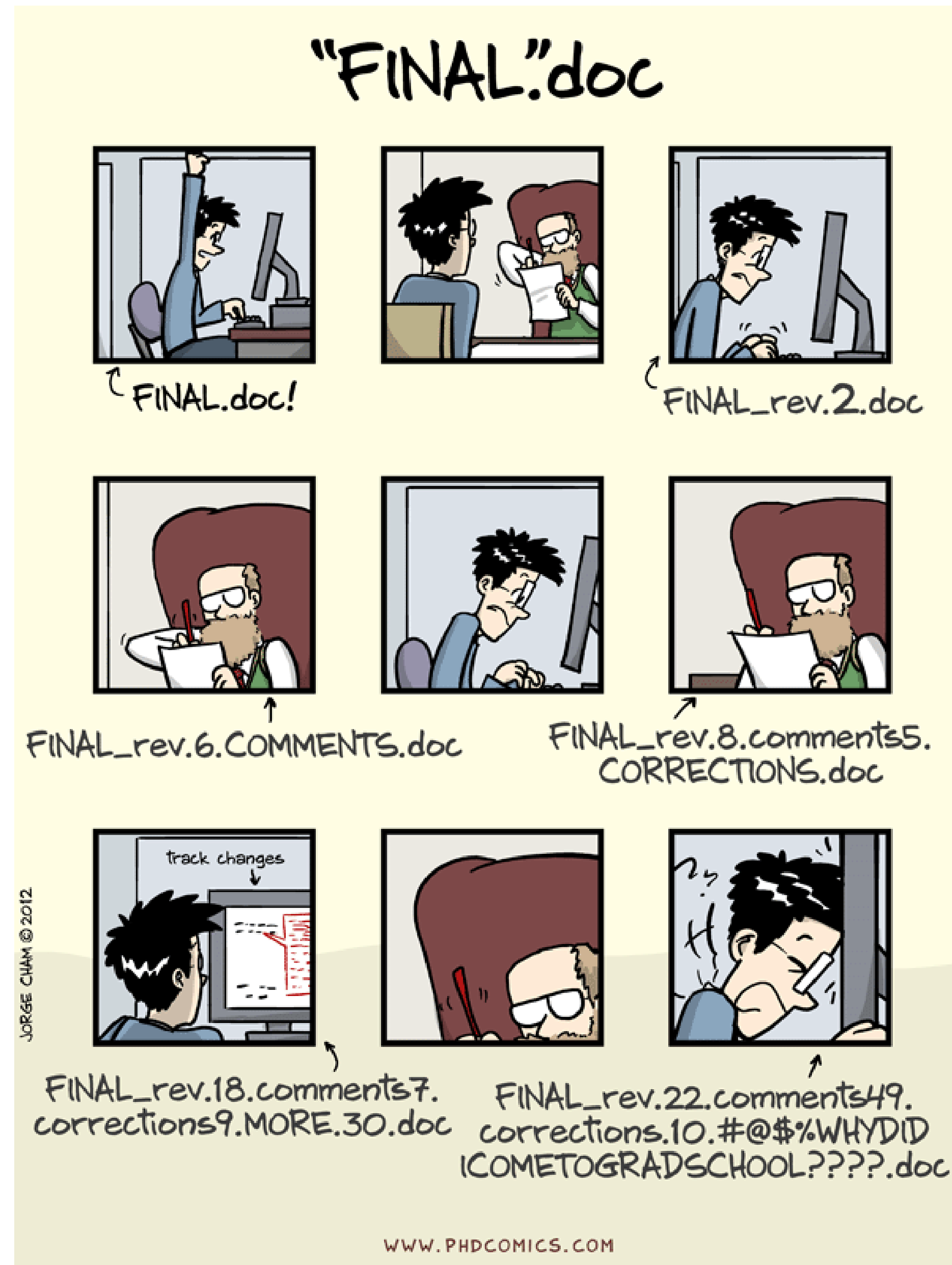
Vorstellungsrunde

Ein paar Leitfragen:

- Welches Betriebssystem?
- Installation schon erfolgreich durchgeführt?
- Erfahrungen mit der Bash/Shell?
- Erfahrungen mit Git?



Versionen



Ziel der Einführung in Git

- Verständnis für die Funktionsweise von Git
- Grundlagen der Bash
- Notwendige Werkzeuge für den Umgang mit (Text-)Daten
- Wo liegt der aktuelle und finale Arbeitsstand eines Projekts
- Wann wurde was von wem dort bearbeitet



Kursinhalt

1. Was ist Git
2. Vorteile / Nachteile
3. Anwendungsfälle in den Geisteswissenschaften?
4. Git / GitLab / GitHub
5. Versionen und Versionierung
6. Hands-on Übungen



Was ist Git



Git kommt aus der Softwareentwicklung und wurde von Linus Torvalds initiiert.



Es ist Open Source Software → GPL 2



Es steht für alle Betriebssysteme zu Verfügung



Git ist das vermutlich am weitesten verbreitete Werkzeug zur Versionsverwaltung unter Programmierer*innen.



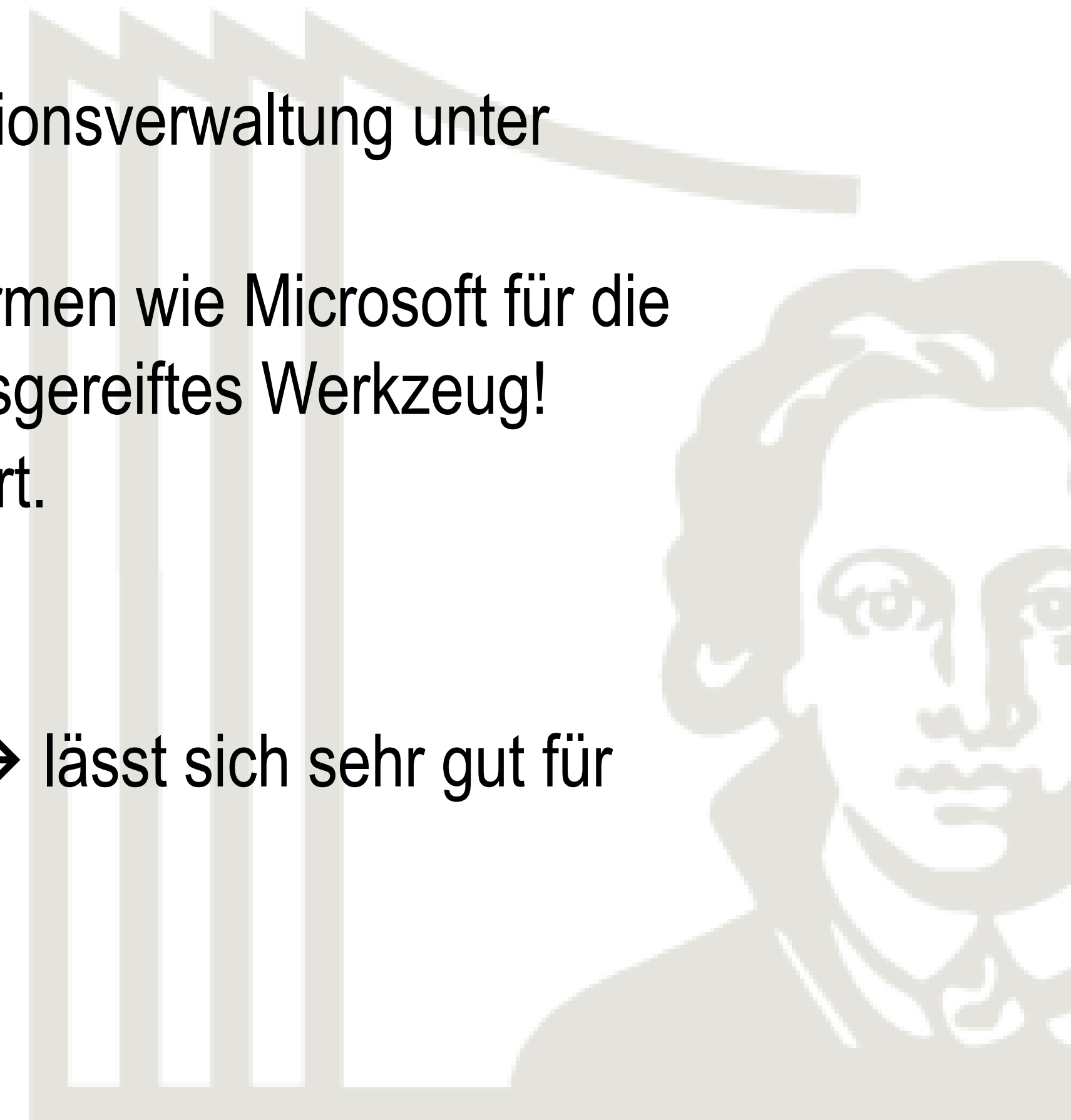
Git wird von großen Projekten im Linux-Kontext, aber auch von Firmen wie Microsoft für die Windows-Entwicklung eingesetzt → Git ist deshalb ein extrem ausgereiftes Werkzeug!



Viele Funktionen sind deshalb an einem Coding-Workflow orientiert.

ABER:

Es ist ein System zur verteilten Versionsverwaltung von Dateien → lässt sich sehr gut für andere Fachbereiche adaptieren.



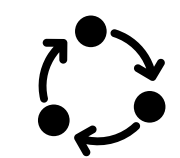
Bisherige Arbeitsweise



Organisation mit Verzeichnissen und Dateien in Windows, macOS oder Linux



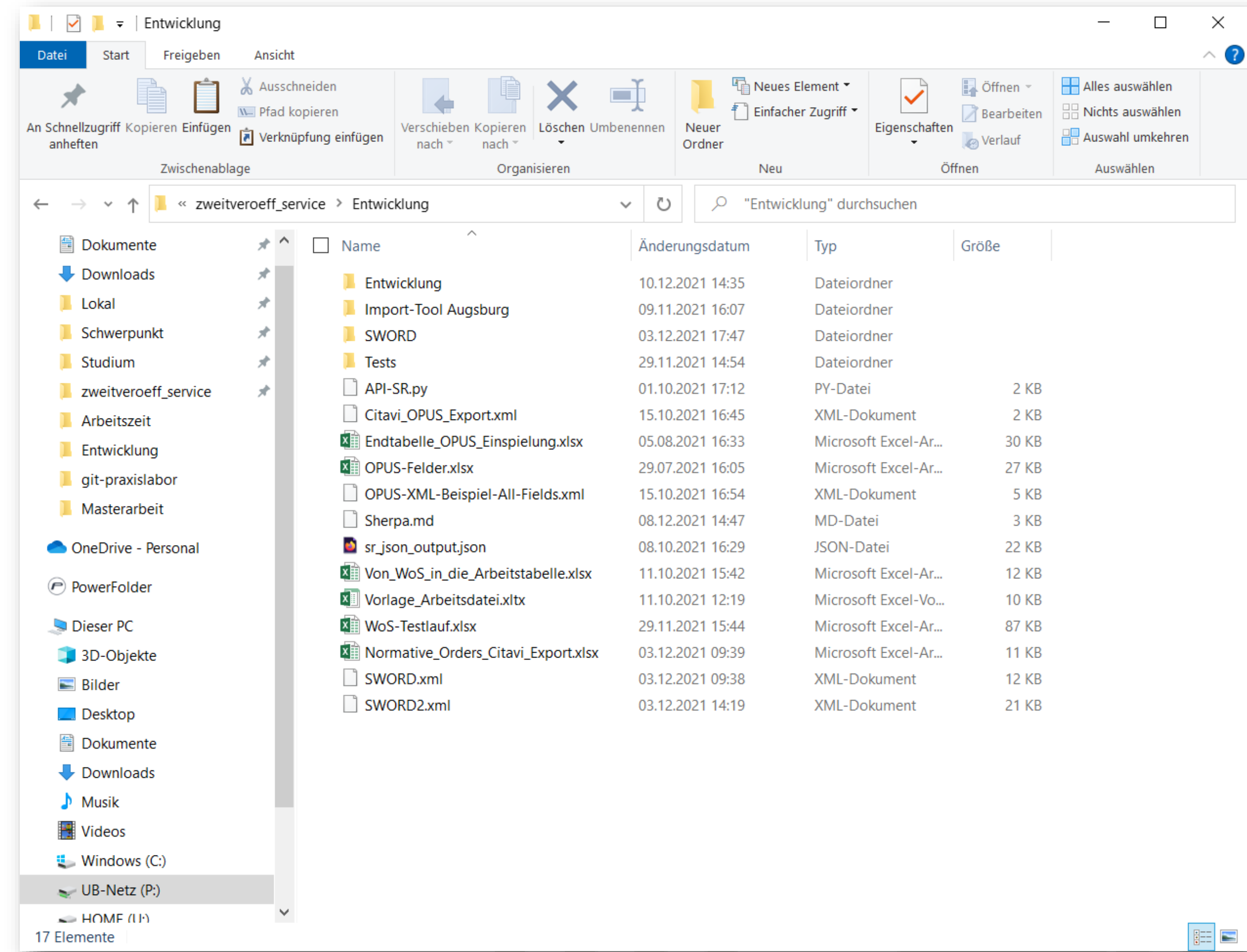
Mehr oder weniger stringente Benennung von Dateien



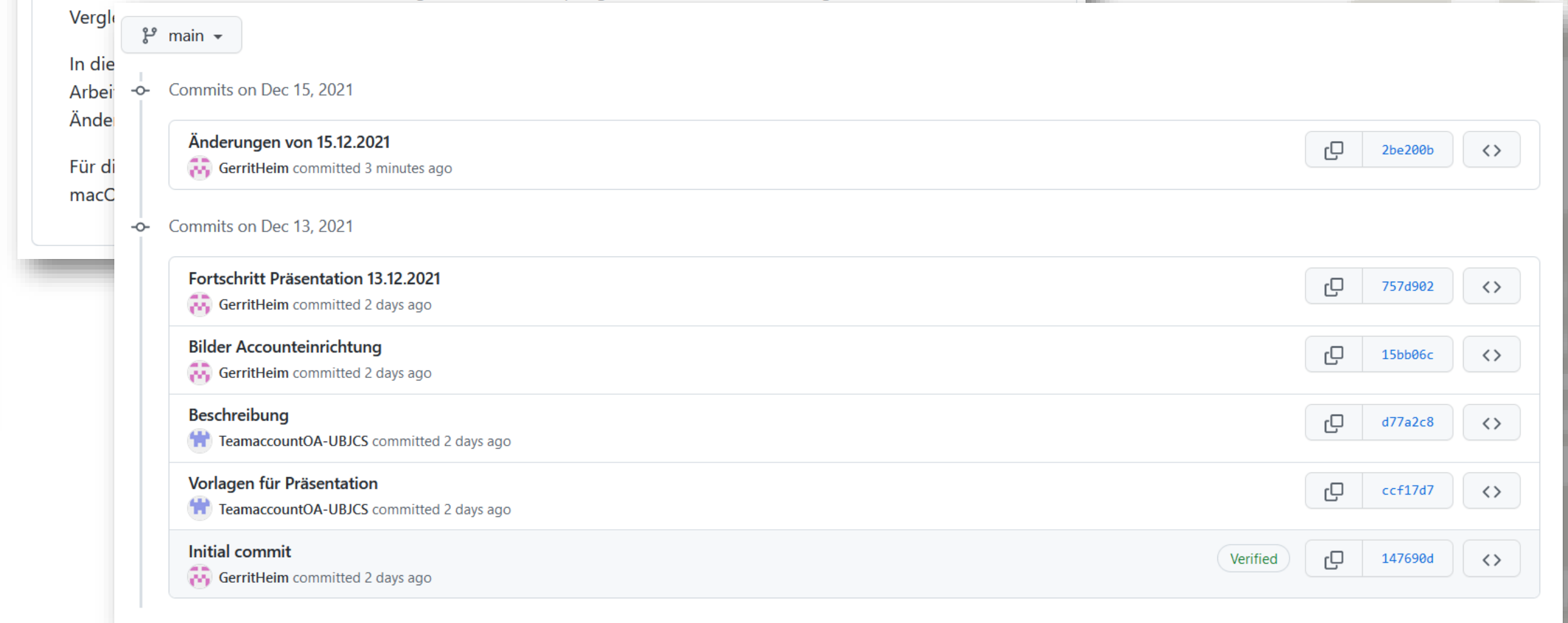
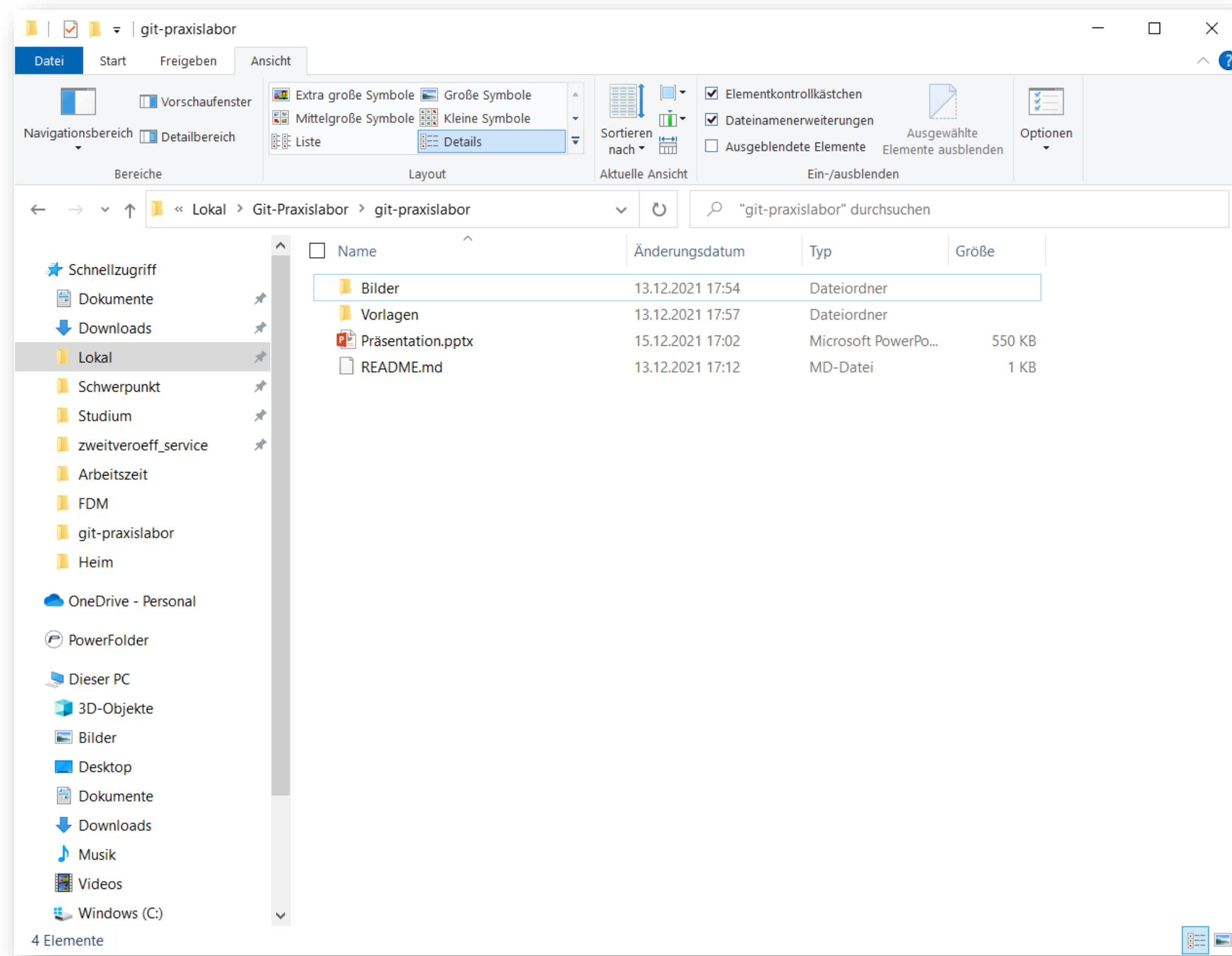
Unklar welcher Arbeitsstand sich hinter welcher Datei befindet.



Unklar wer an welchen Dateien gearbeitet hat.



Projekt in Git / GitHub



Vorteile

- Daten liegen lokal auf dem Gerät und auf dem Repository von GitHub / GitLab
- Die Versionierung ist robuster und weniger fehleranfällig als bei Cloudspeichern
- Die Versionierung ist aussagekräftiger und übersichtlicher als der Änderungsverlauf bei Office-Programmen
- Versionierte Arbeitsstände können jederzeit zurück geholt werden.
- Transparenz (Wer schrieb was wann)
- Konflikte bei paralleler Bearbeitung können erkannt und behoben werden.



Nachteile

- Git hat mehr Funktionen als die meisten Anwender benötigen
- Anfangs eine steile Lernkurve.
- Viele Funktionen stehen nicht für bestimmte Dateiformate zur Verfügung:
- Die Goethe-Universität hat keinen eigenen GitLab-Server



Anwendungsfälle im Bereich der Geisteswissenschaft

- Organisation der eigenen Projekte
 - Hausarbeiten / Abschlussarbeiten
 - Tabellarische Daten verwalten, z. B. exportiere Literaturlisten
 - Referate & Präsentationen erstellen
 - Code pflegen, z. B. XML oder Skripte
- Kollaboratives Arbeiten
- Große Datenbestände verwalten



Wissenschaftliches Arbeiten in den Geisteswissenschaften 2022

- Digitale Notizbücher → z. B. auf Markdown-Basis wie Joplin (<https://joplinapp.org/>)
- Sinnvolle Dateinamen, Datei-Templates und Übernahme von Praktiken aus der Programmierung
- Verwendung von Klartext-Formaten (LaTeX, Markdown, Plaintext etc.)
- Dokumentation für sich selbst und für andere
- Offenheit und Hilfsbereitschaft durch Transparenz und gute Dokumentation



Versionsvergleich

- 😊 **Versionierung und Inhaltsvergleich:**
 - Textdateien (Markdown, LaTeX, XML/TEI, Plain-Text)
 - Daten (CSV)
 - Code
 - Zum testen: Lässt sich die Datei im Editor öffnen?
- 😞 **Nur Versionierung:**
 - Proprietäre Dateien (OOXML → DOCX, XLSX etc.)
 - Gepackte Dateien
 - Binärdateien
 - Medien

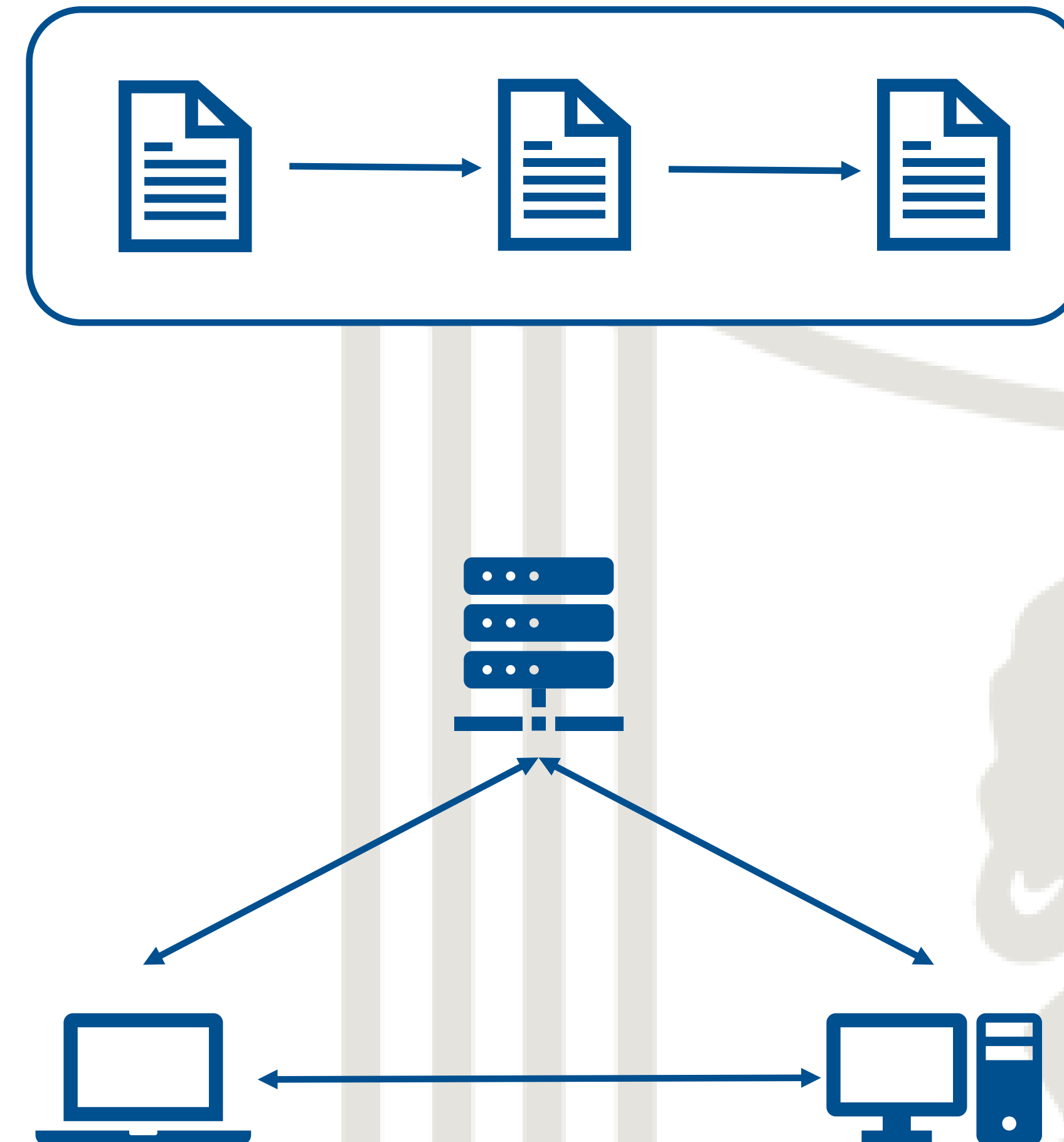


Git, GitLab, GitHub – Git was?

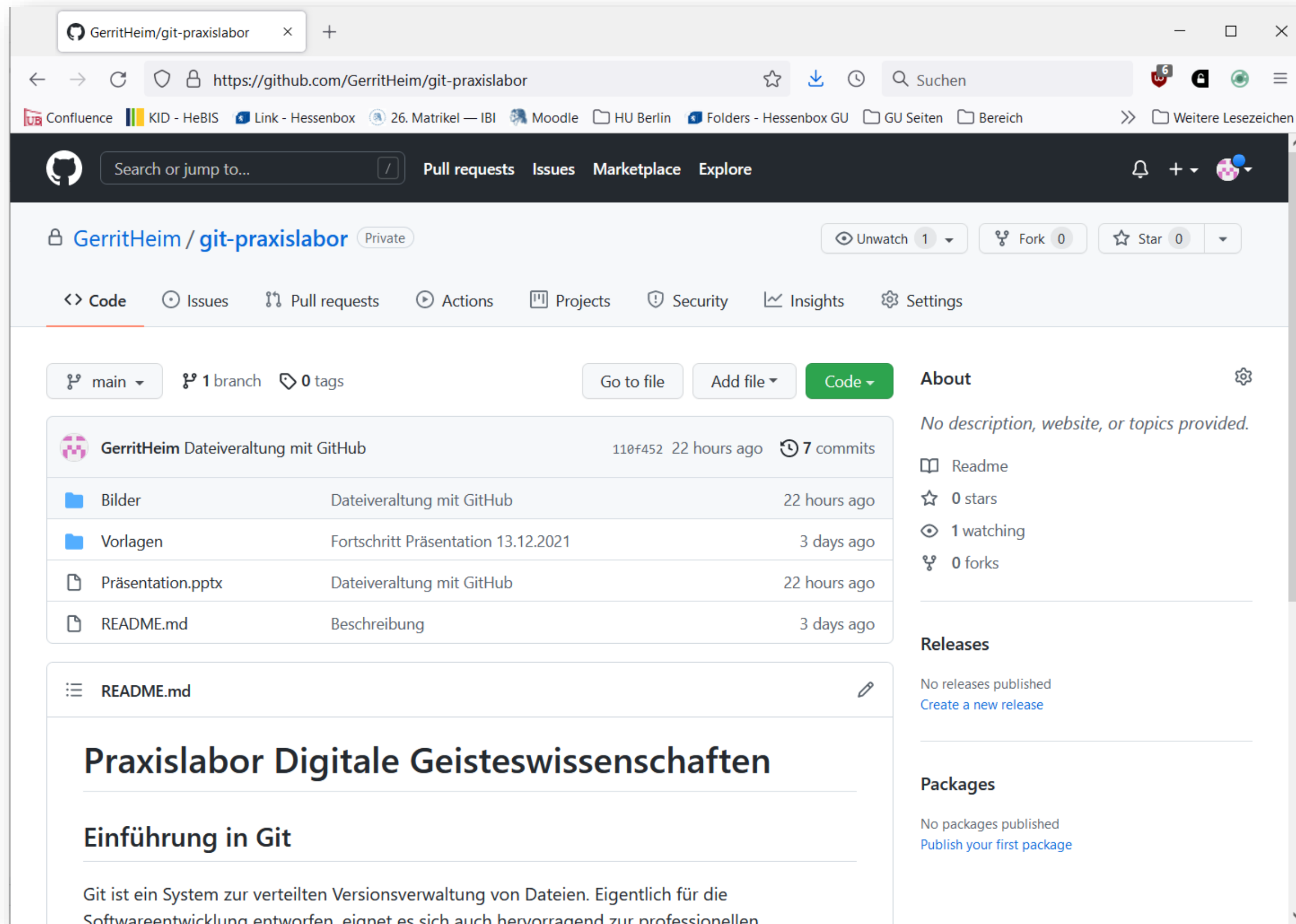
Git bezeichnet das Werkzeug. Es lässt sich theoretisch auch lokal auf einem Betriebssystem installieren oder nutzen (quasi als „lokale Versionsverwaltung“)

Seine volle Stärke kann Git aber nur ausspielen wenn mehrere Clients mit einem zentralen Repository kommunizieren (verteilte Versionsverwaltung)

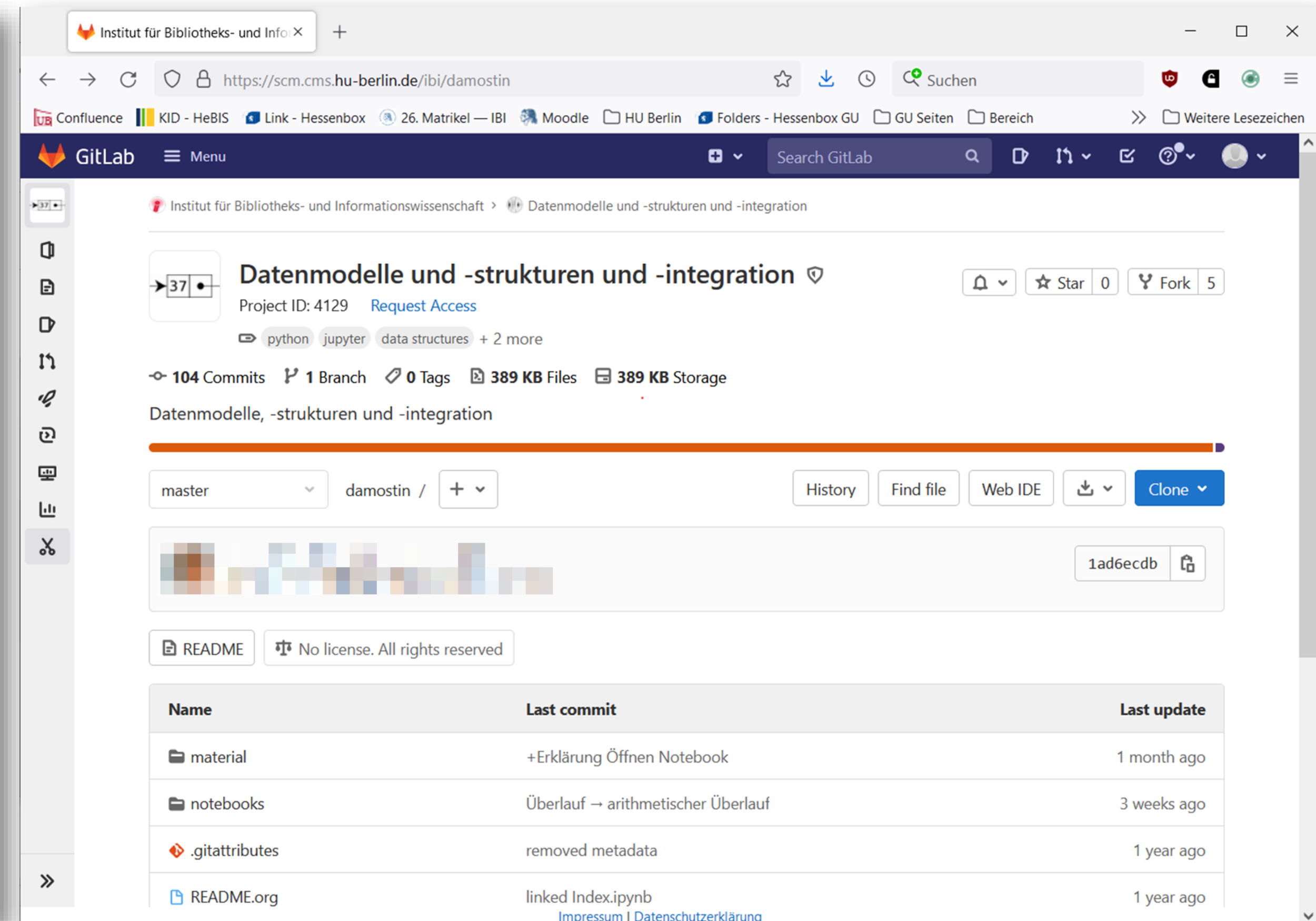
→ dazu benötigt: GitLab oder GitHub



GitHub / GitLab – Nicht abschrecken lassen!



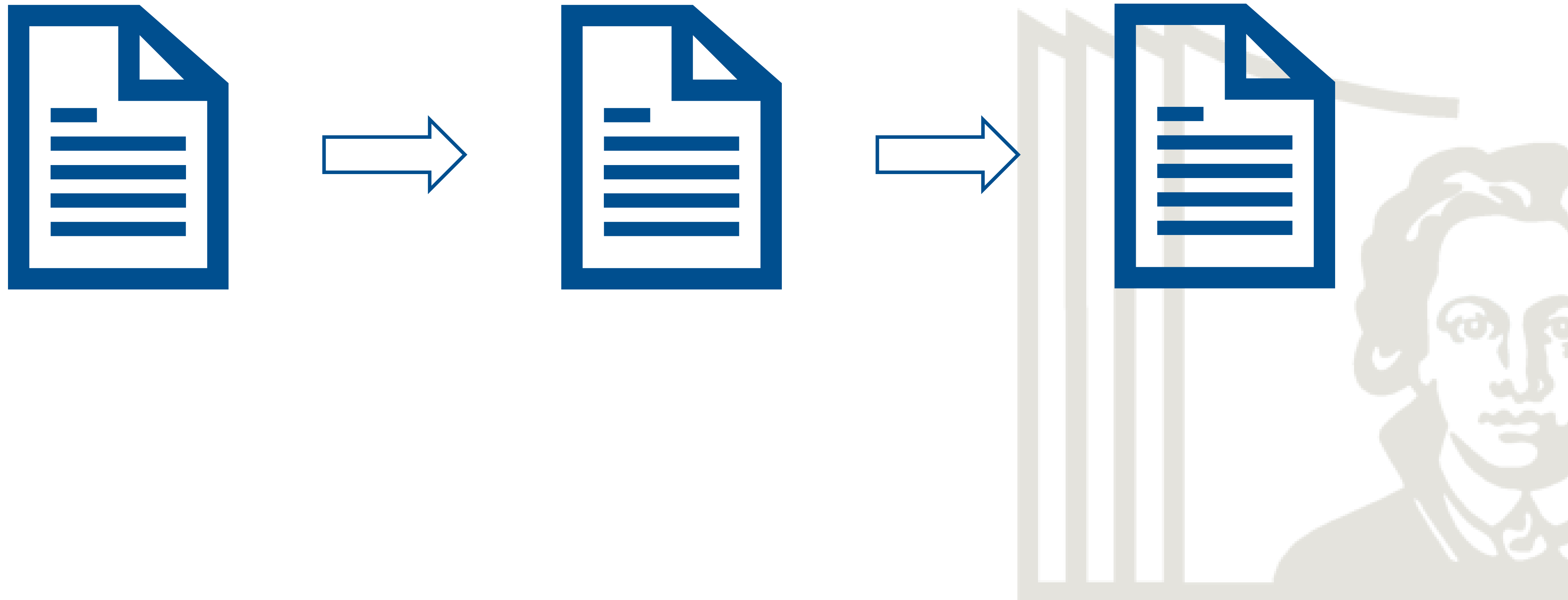
The screenshot shows the GitHub interface for the repository 'GerritHeim/git-praxislabor'. The repository is private and has 1 branch and 0 tags. The main branch is 'main'. The repository contains several files and folders, including 'Bilder', 'Vorlagen', 'Präsentation.pptx', and 'README.md'. The README file is selected, showing the title 'Praxislabor Digitale Geisteswissenschaften' and the subtitle 'Einführung in Git'. The repository has 110k452 commits, 22 hours ago, and 7 commits. The repository is described as 'Dateiverwaltung mit GitHub'.



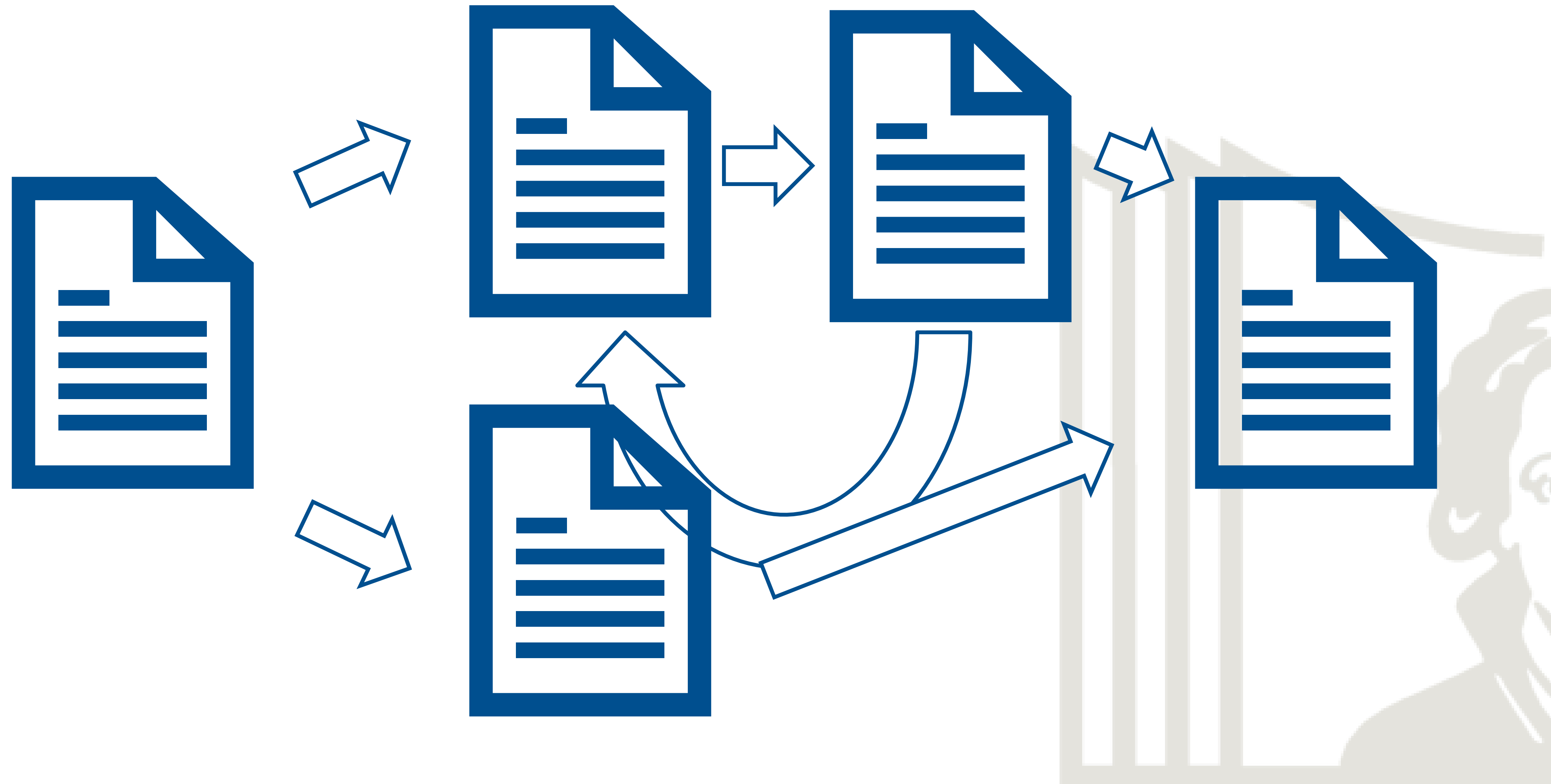
The screenshot shows the GitLab interface for the repository 'Institut für Bibliotheks- und Informationswissenschaft / Datenmodelle und -strukturen und -integration'. The repository is public and has 5 forks and 0 stars. The main branch is 'master'. The repository contains several files and folders, including 'material', 'notebooks', '.gitattributes', and 'README.org'. The repository has 104 commits, 1 branch, 0 tags, 389 KB files, and 389 KB storage. The repository is described as 'Datenmodelle, -strukturen und -integration'. The repository has 1ad6ecdb as the commit hash.


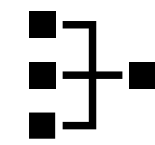

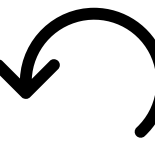
Name	Last commit	Last update
material	+Erklärung Öffnen Notebook	1 month ago
notebooks	Überlauf → arithmetischer Überlauf	3 weeks ago
.gitattributes	removed metadata	1 year ago
README.org	linked Index.ipynb Impressum Datenschutzerklärung	1 year ago

Linearer Arbeitsverlauf



Arbeitsverlauf in einem kollaborativen Projekt



-  Es gibt keine letzte Änderung
-  Es kann gleichzeitig und an mehreren Stellen gearbeitet werden
-  Änderungen brauchen Kommentare
-  Änderungen lassen sich revidieren

Versionen sind Varianten desselben Dokuments. Sie sind nie komplett gleich, aber einander ähnlich genug, um als das gleiche Dokument zu gelten

Versionierung ist der Mechanismus, um die Entwicklung eines Dokuments als Versionskette festzuhalten.

Vier Basisschritte

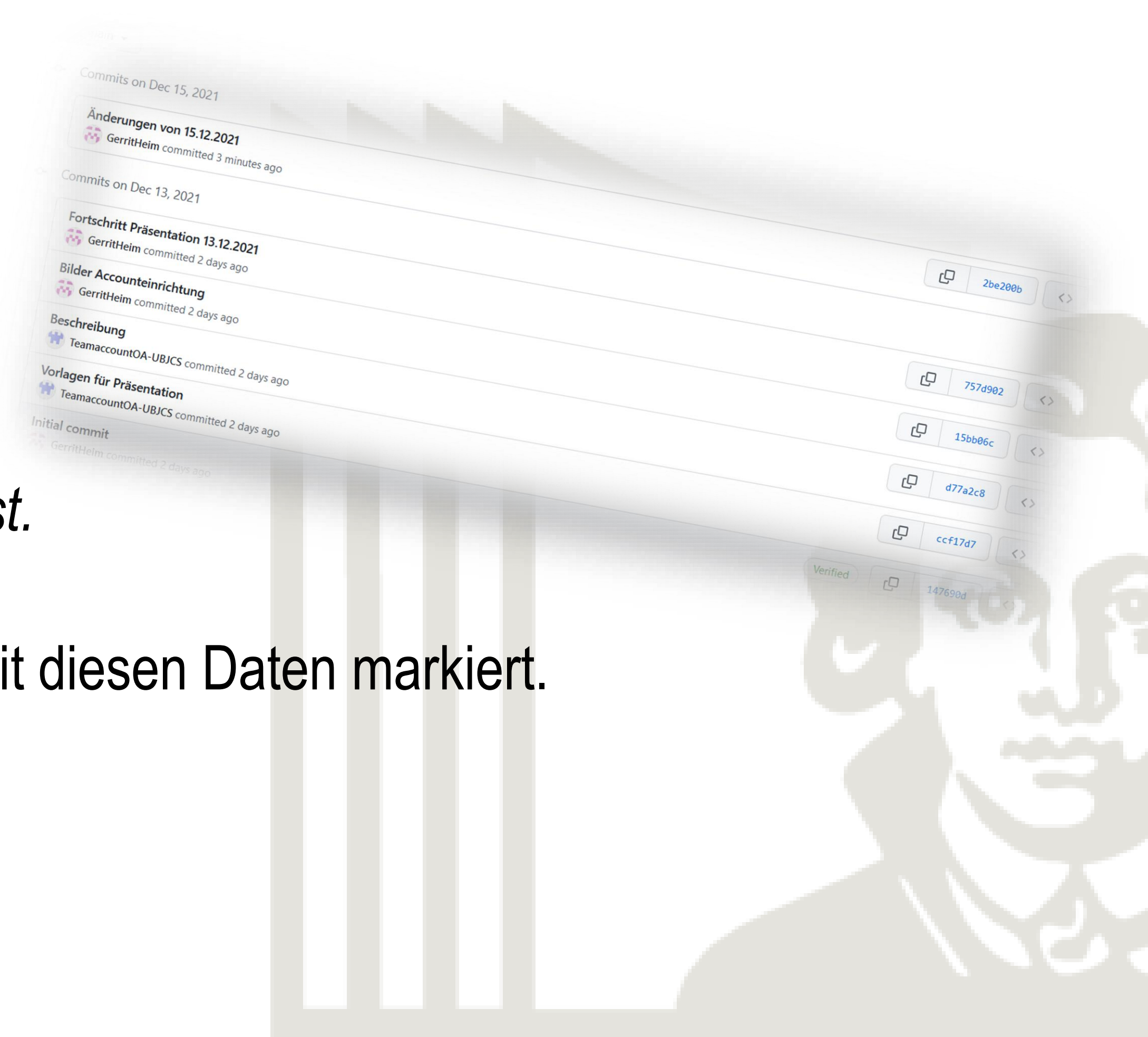
1. Dateien erstellen / bearbeiten
2. Änderungen betrachten
3. Änderungen für Abgabe vorbereiten (Staging)
4. Änderungen abgeben (Commiten)



Ein paar Fingerübungen auf der Bash. Festlegen des Benutzernamens und der E-Mail Adresse

```
$ git config --global user.name „Benutzername“  
  z. B. $ git config --global user.name „GerritHeim“  
$ git config --global user.email „name@domain.tl“  
  z. B. $ git config --global „g.heim@ub.uni-frankfurt.de“  
$ git config --global init.defaultBranch main  
  legt „main“ als Standardname anstelle von „master“ fest.
```

An GitHub oder GitLab übertragene Änderungen werden mit diesen Daten markiert.



Dateien im Repository: Lokales Repository anlegen

`$ cd /Pfad/zum/Ordner`

z. B. `$ cd /C/Users/gheim/Lokal/Git-Praxislabor/`
für Sie: `~/Desktop`

`$ mkdir praxislabor-uebung`

Erstellt ein Verzeichnis „praxislabor-uebung“

`$ cd praxislabor-uebung`

Wechselt in dieses Verzeichnis

`$ git init`

Initialisiert ein Git-Repo

`$ ls -la`

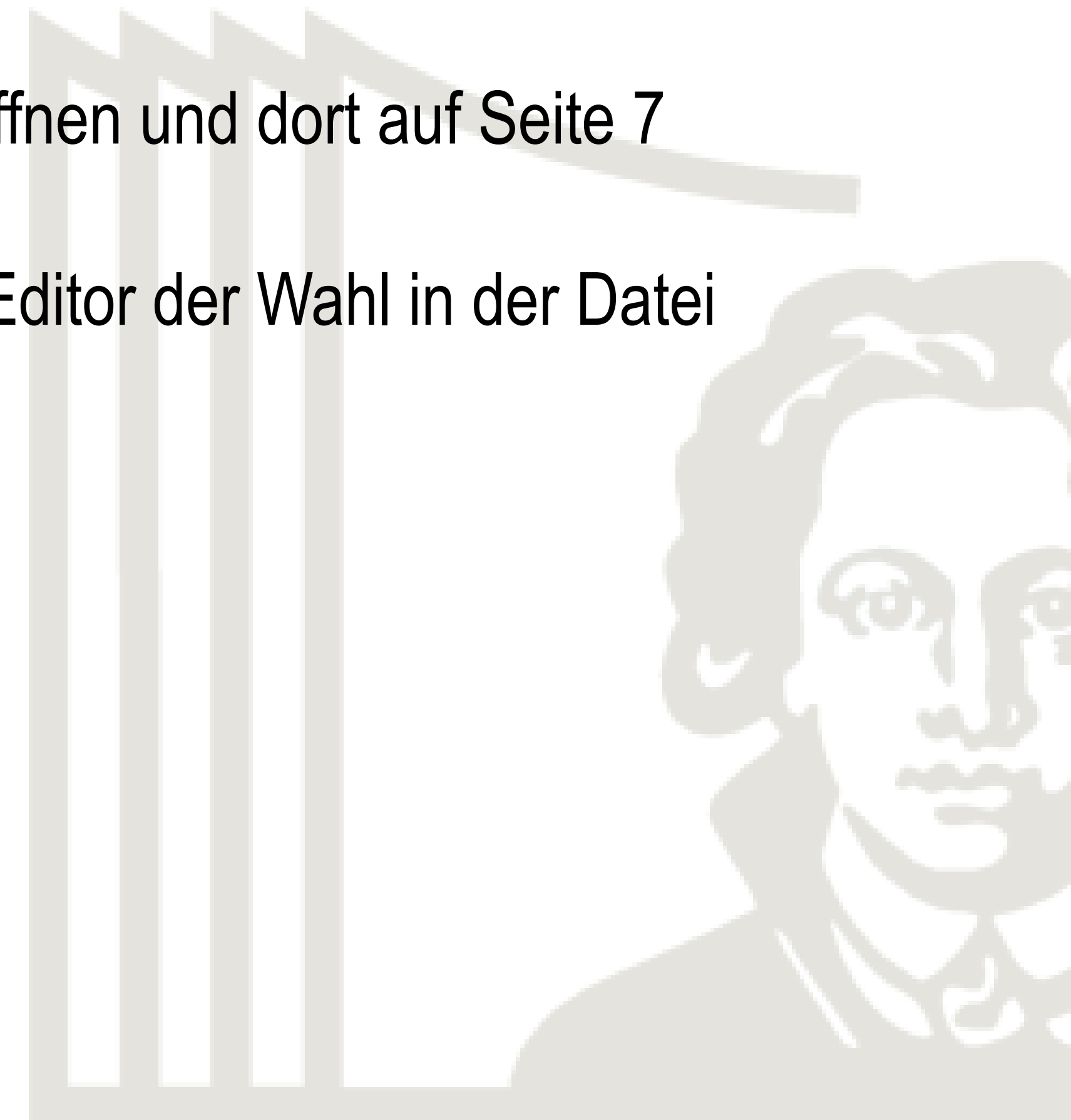
Zeigt die Dateistruktur im Verzeichnis

`$ git status`



Dateien im Repository: Hinzufügen

1. Hessenbox Material herunterladen: <https://hessenbox-a10.rz.uni-frankfurt.de/getlink/fi9yZPGZVxCqTVgmXhmFzDr/>
2. PDF-Dateien in das Git-Verzeichnis (praxislabor-uebung) kopieren
3. Text „Graf, Fadeeva et. al. (Hg.) 2020 - Bücher im Open Access“ öffnen und dort auf Seite 7 zum Beginn der Einleitung gehen.
4. Titel, Autor*innen und Text auf der Seite 7 kopieren und in einem Editor der Wahl in der Datei „Einleitung.md“ speichern. Gerne mit Markdown formatieren.



Dateien im Repository: Änderung vornehmen

`$ git status`

Gibt den Status aus

`$ git add .`

Fügt alle Änderungen einem Commit hinzu.

`$ git commit -m „Texte und Einleitung“`

Erstellt einen Commit mit Beschreibung

`$ git status` oder `$ git log`



Dateien im Repository: Änderung ergänzen und zurücksetzen

Datei im Verzeichnis erneut öffnen, weiteren Text hinzufügen und speichern.

```
$ git add . oder $ git add Einleitung.md
```

```
$ git status
```

```
$ git commit -m „Weitere Einleitungspassage“
```

Auf einen vorherigen Stand zurücksetzen

```
$ git log
```

```
$ git checkout <id> Einleitung.md
```



Rückblick und Ausblick

Was haben wir gemacht?

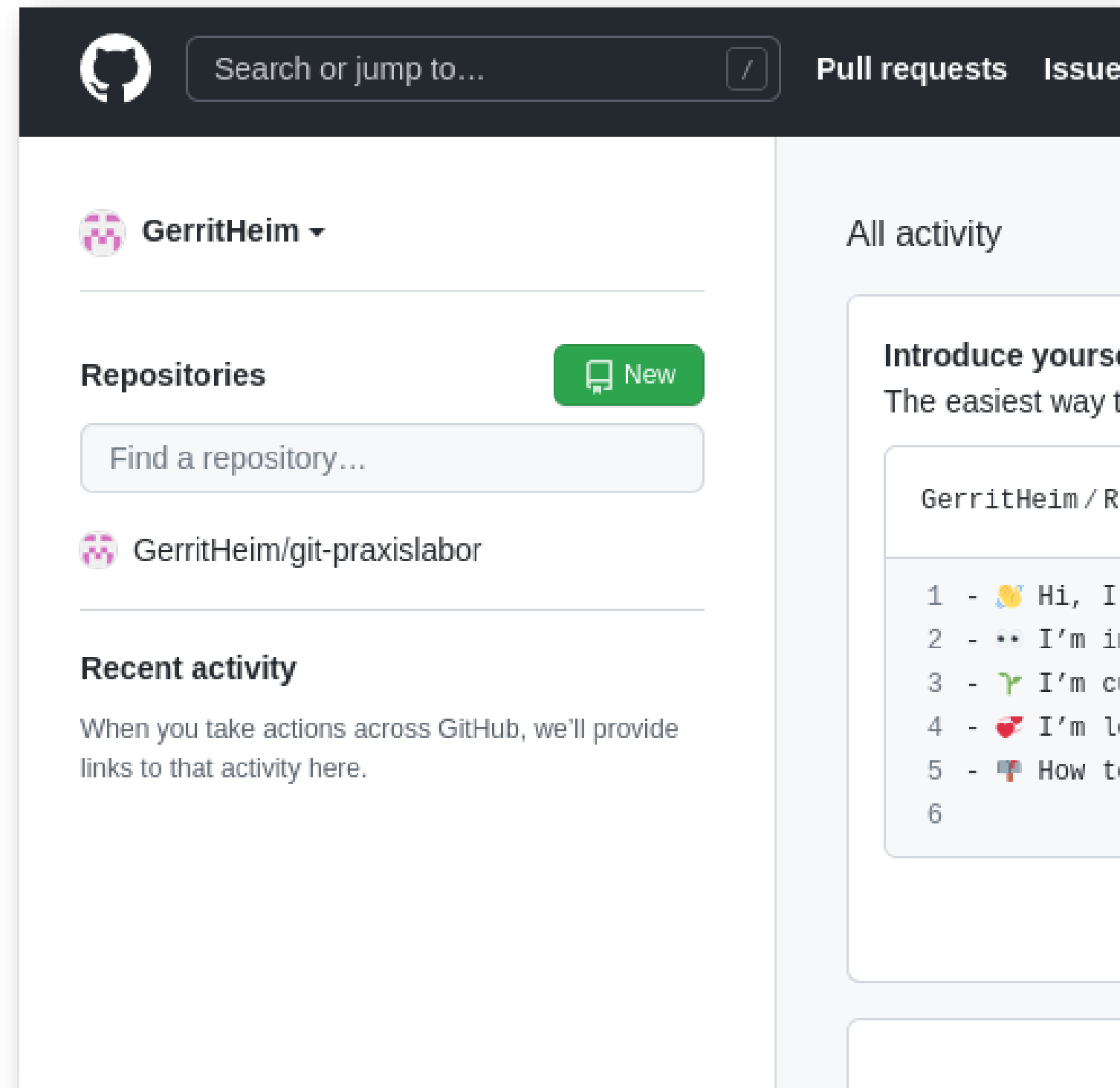
- Git konfiguriert
- Ein lokales Repository angelegt
- Dateien hinzugefügt und bearbeitet
- Commits erstellt
- Versionen zurückgesetzt

Und nächste Woche?

- Arbeiten mit GitHub!



Spoiler



The screenshot shows the GitHub profile page for Gerrit Heim. At the top, there is a search bar and navigation links for 'Pull requests' and 'Issues'. Below the profile name 'Gerrit Heim', there is a 'Repositories' section with a 'New' button and a search bar. A repository 'GerritHeim/git-praxislabor' is listed. The 'Recent activity' section explains that links are provided for actions across GitHub. On the right, the 'All activity' section is partially visible, showing a commit message starting with '1 - Hi, I\'...'.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

GerritHeim ▾

Repository name *

praxislabor-uebung ✓

Great repository names are short and memorable. Need inspiration? How about [musical-bassoon?](#)

Description (optional)

- Public**
Anyone on the internet can see this repository. You choose who can commit.
- Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

- Add a README file**
This is where you can write a long description for your project. [Learn more.](#)
- Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)
- Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository



Cheat Sheet Basis

`git init <Pfad>`

z.B. „`git init .`“ für alle geänderten Dateien

`git commit -m „Beschreibung der Änderung“`

Erzeug ein Commit

`git push`

Schiebt die lokalen Änderungen (Commits) auf den Server

`git pull`

Lädt die Änderungen vom Server herunter



Cheat Sheet Basis

`git branch`

Listet alle Branches

`git branch <name>`

Erzeug einen neuen Branch

