

# Abschlussprojekt

von Fabio Kälin

GitNas

## Inhaltsverzeichnis

Planung .....	3
Idee .....	3
User Cases .....	3
Risiken .....	3
Konzept (UI) .....	4
Konzept (Code) .....	5
Technologien und Tools .....	6
Neues zum Lernen .....	6
Arbeitspakete .....	6
Zeitplanung .....	7
Dokumentation .....	8
Programm-Ablauf .....	8
System-Architektur .....	9
Vorschau .....	10
Repositories .....	10
Repository erstellen .....	11
Leeres Repository .....	12
Explorer .....	13
Commits .....	14
Repository-Einstellungen .....	15
Editor .....	16
Tags .....	17
Arbeitspakete-Überprüfung .....	18
Reflexion .....	19
Testprotokoll .....	19
Quellen .....	21

## Planung

### Idee

Ich habe im April 2022 auf meinem Nas ein Git-Server eingerichtet und möchte nun eine Übersicht dafür machen. Bevor ich meine Projekte auf das Nas hochgeladen habe, nutzte ich GitHub und bin mir diese Ansicht gewohnt. Daher möchte ich dasselbe für mein Nas machen.

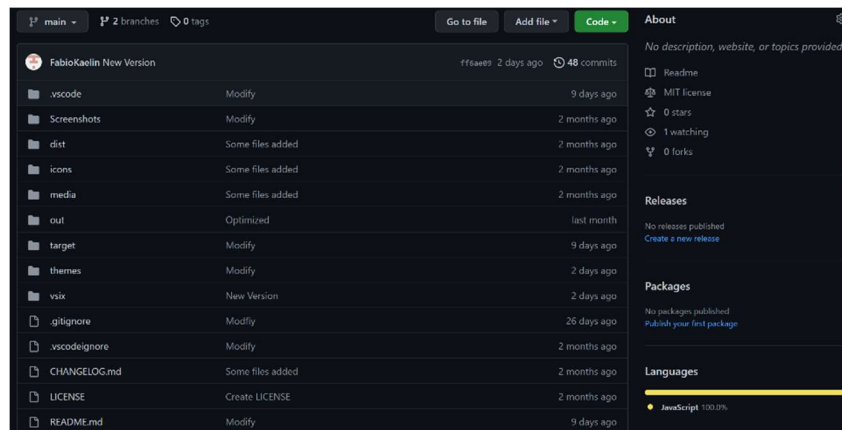


Abbildung 1: Übersicht von GitHub

### User Cases

Ich möchte meine Repositories auf einen Blick sehen und mit einer Beschreibung versehen, damit ich mir ein Bild von dem Projekt machen kann.

Wenn ich in einem Repository bin, möchte ich einen Fileexplorer haben, damit ich meine Files angenehm suchen kann.

Ein vorbereiteter Git-Command soll kopiert werden können, um das Projekt zu klonen, pushen und downloaden, damit ich diese Befehle nicht auswendig wissen muss.

Ich möchte die Files mit einem Syntax-Highlight oder einem Bild ansehen, damit ich überprüfen kann, ob es die richtige Version ist.

Im Fileexplorer soll es ein Icon haben, welches mir sagt, was für ein File Type es ist, um eine bessere Übersicht zu haben.

Ich möchte neue Repositories erstellen und auch wieder löschen können, damit ich dies nicht in der Kommando-Zeile machen muss.

Man soll mit einem Knopf zwischen den Branches wechseln können, damit man auch andere Branches ansehen kann, um zu überprüfen, ob alles richtig ist.

### Risiken

1. Ich könnte Mühe mit den Git-Commands haben und lange nach dem richtigen Befehl suchen.
2. Da ich es mit Python programmiere, muss ich darauf achten, dass die Ladezeit nicht zu lange dauert und die Geschwindigkeit optimieren wird.
3. Ich könnte die Dokumentation vergessen oder vernachlässigen.

## Konzept (UI)

Die Home-Seite wird eine Übersicht aller Repositories mit Beschreibung und Icon.

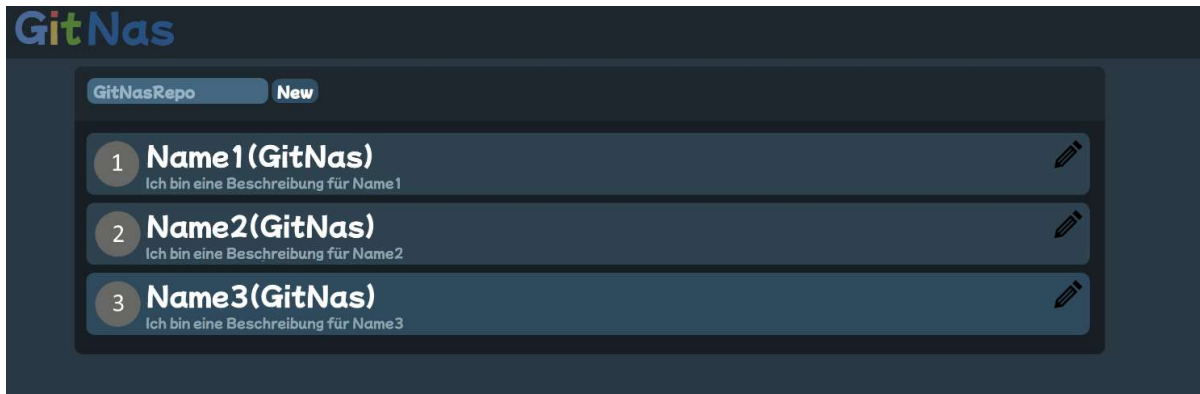


Abbildung 2: Konzept der Repository-Übersicht

Wenn ich in ein Repository hineingehe, möchte ich dieselbe Ordnerstruktur haben wie im Projekt und die Files mit dessen Icons sowie das letzte Änderungsdatum sehen. Im oberen Teil soll zudem der Pfad zu sehen sein.

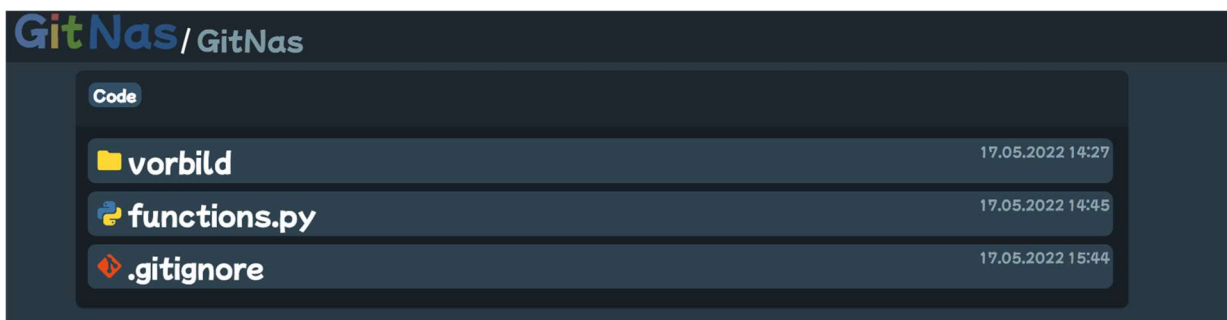


Abbildung 3: Konzept des Explorers

Im File möchte ich ein leichtes Syntax-Highlight haben, um die Lesbarkeit zu verbessern. Wie auch im Explorer soll es den Pfad haben. Zudem möchte ich weitere Informationen zum File wie z.B. Grösse, Linien und Type.

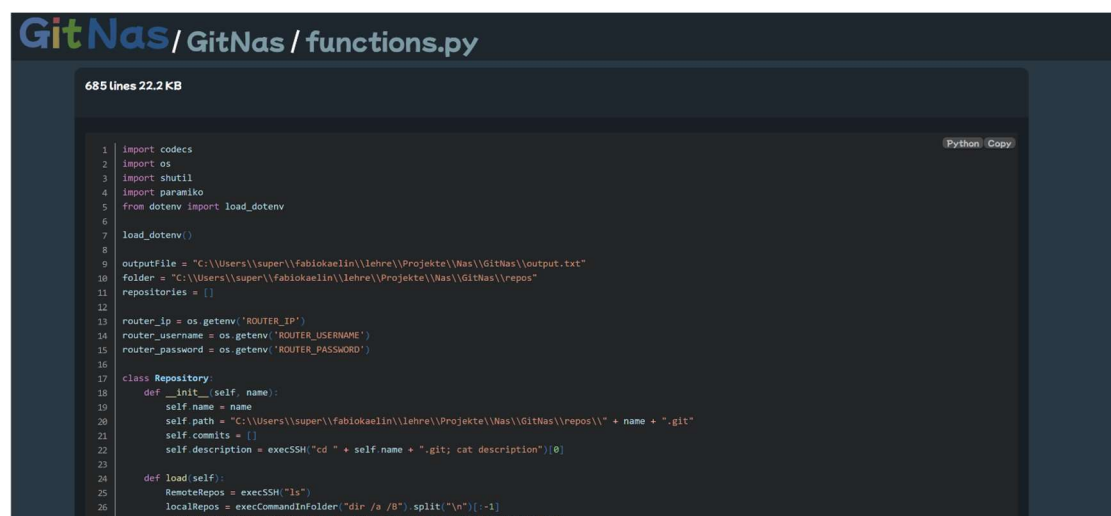


Abbildung 4: Konzept des Editors

## Konzept (Code)

Ich möchte objektorientiert programmieren und so meinen Code übersichtlicher und strukturierter machen.

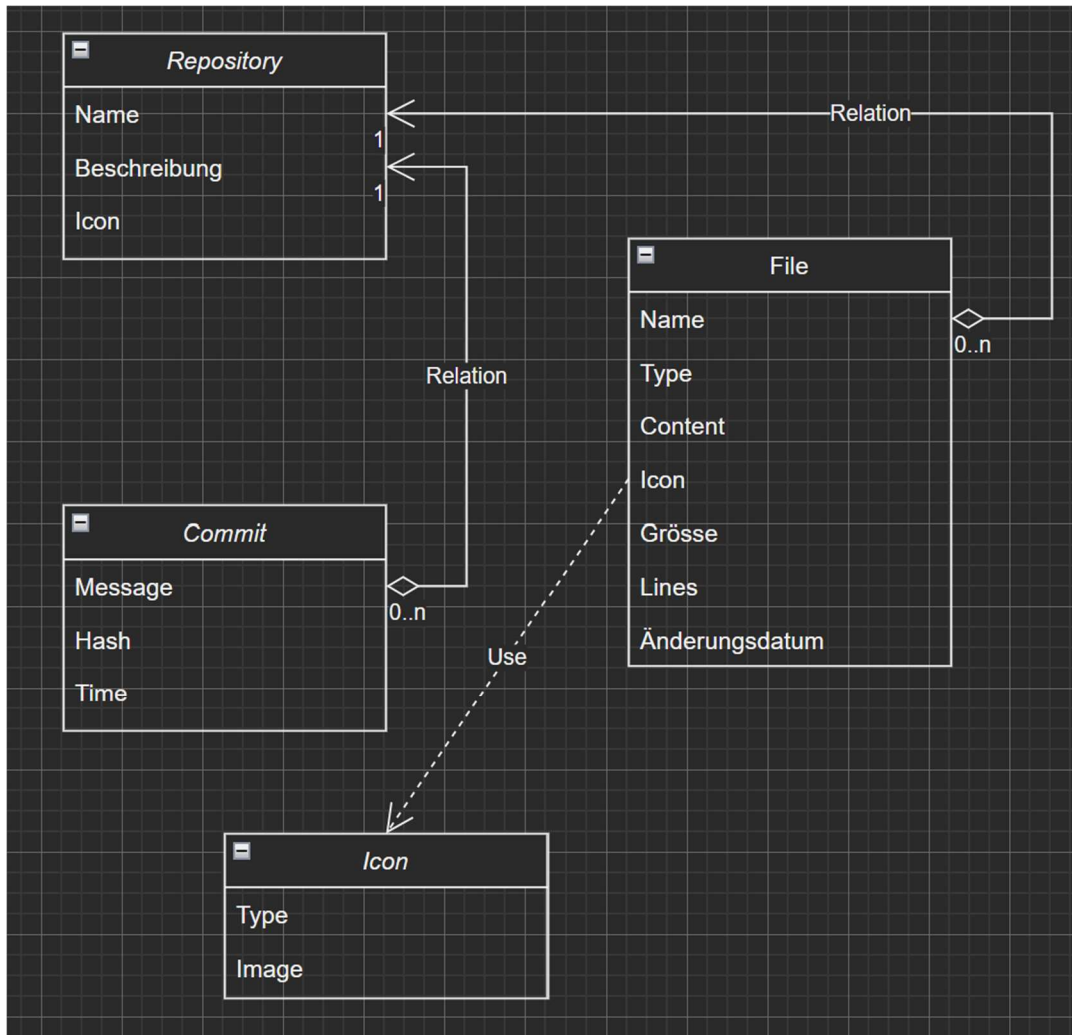


Abbildung 5: Klassen-Diagramm

Zudem erstelle ich eine Ordnerstruktur, welche sortiert, aber nicht künstlich verkompliziert wird.

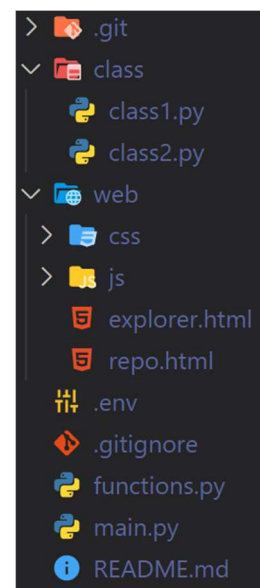


Abbildung 6: Konzept der Ordnerstruktur

## Technologien und Tools

Ich werde mein Projekt mit **Python** machen und als GUI nutze ich das **Package «eel»**, welches mir ein Browser-Popupfenster bereitstellt, in welchem ich mit HTML, CSS und JavaScript arbeiten kann.

Um das Syntax-Highlight bereitzustellen, nutze ich **Prismjs**, welches dies für mich macht.

Für die File-Icons werde ich die Icons der VSCode-Extension **Material Icon Theme** nutzen.

Um etwas mit Git machen zu können, muss ich **Git** überhaupt haben.

Ich werde GitNas für **Windows 11** machen, da ich damit am meisten Erfahrung habe und es auch nutze.

## Neues zum Lernen

Ich werde das Package «eel» lernen müssen, wie man es mit Python sowie JavaScript nutzt und auch neue Git-Befehle.

## Arbeitspakete

- Idee beschreiben
- User Cases formulieren
- Risiken beschreiben und Konzept (UI) machen
- Konzept (Code) und Technologien erstellen
- Arbeitspakete erstellen
- Zeitplanung zusammenstellen
- Dokumentation Vorlage erstellen
- Puffer
- Ordnerstruktur aufsetzen
- Klasse erstellen
- HTML und CSS
- Funktionen schreiben
- Repositories ansehen
- Beschreibung bearbeiten
- Dokumentieren
- JS für Repository
- Explorer anzeigen
- File-Icons anzeigen
- Pfad erstellen
- Dokumentieren
- Editor anzeigen
- Pfad und Sonderzeichen
- Daten über das File
- Neues Repository erstellen
- Leeres Repository Übersicht
- Commits anzeigen
- Puffer
- Dokumentieren
- Branches anzeigen
- Branches wechseln
- Repository als Zip
- Repository-Icon
- About einbetten
- Dokumentation einbetten
- Tags
- Release
- Puffer
- Dokumentieren
- Dokumentieren
- Tests formulieren
- Tests ausführen
- Testprotokoll führen
- Bugs beheben
- Puffer
- Präsentation erstellen
- Präsentation üben
- Factsheet erstellen
- Puffer

## Zeitplanung

Tag 1	Idee beschreiben User Cases formulieren Risiken beschreiben und Konzept (UI) machen Konzept (Code) und Technologien erstellen	Planung
Tag 2	Arbeitspakete erstellen Zeitplanung zusammenstellen Dokumentation Vorlage erstellen Puffer	Planung
Tag 3	Ordnerstruktur aufsetzen Klasse erstellen HTML und CSS Funktionen schreiben	Umsetzung
Tag 4	Repositories ansehen Beschreibung bearbeiten Dokumentieren JS für Repository	Umsetzung
Tag 5	Explorer anzeigen File-Icons anzeigen Pfad erstellen Dokumentieren	Umsetzung
Tag 6	Editor anzeigen Pfad und Sonderzeichen Daten über das File Neues Repository erstellen	Umsetzung
Tag 7	Leeres Repository Übersicht Commits anzeigen Puffer Dokumentieren	Umsetzung
Tag 8	Branches anzeigen Branches wechseln Repository als Zip Repository-Icon	Umsetzung
Tag 9	About einbetten Dokumentation einbetten Tags Release	Umsetzung
Tag 10	Puffer Dokumentieren Dokumentieren Tests formulieren	Finalisieren
Tag 11	Tests ausführen Testprotokoll führen Bugs beheben Puffer	Finalisierung
Tag 12	Präsentation erstellen Präsentation üben Factsheet erstellen Puffer	Abschluss

## Dokumentation

### Programm-Ablauf

Um den Inhalt und die Commits zu bekommen, nutze ich folgenden Ablauf:

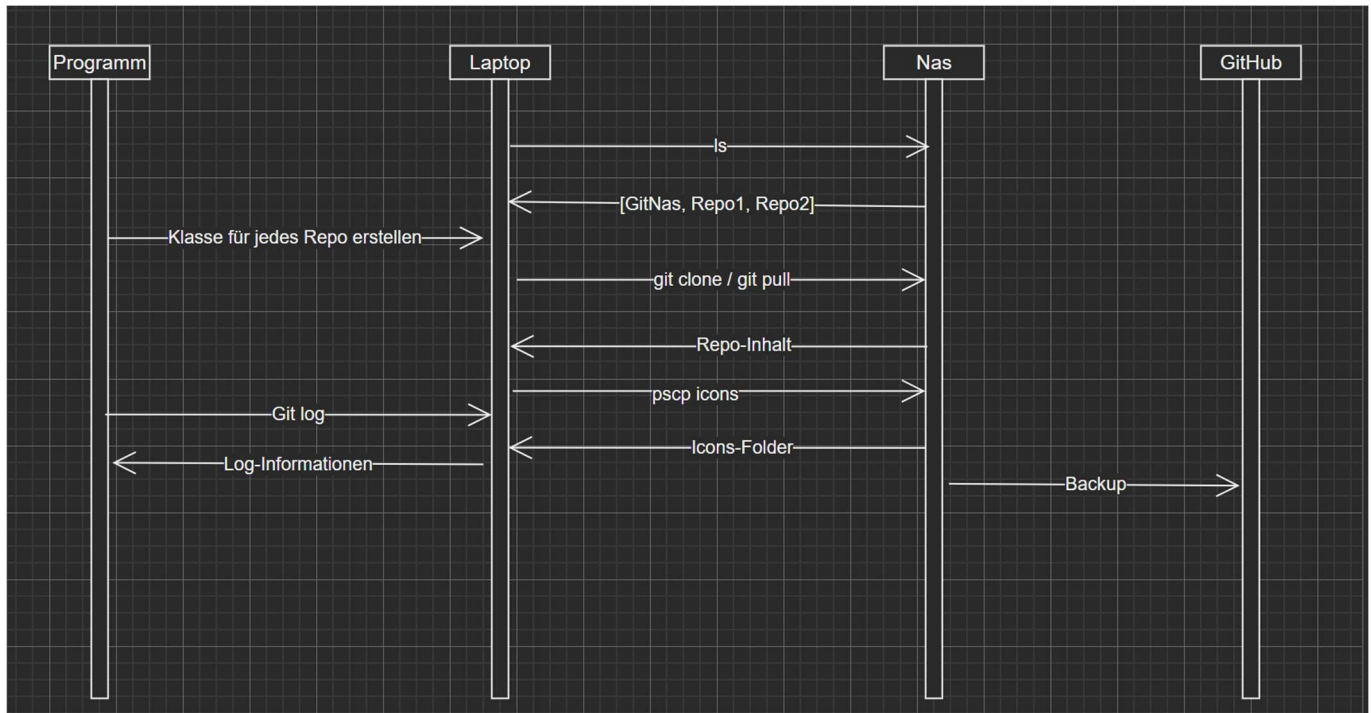


Abbildung 3: Kommunikations-Ablauf

Ich nehme alle Namen von den Repositories mit einem «ls» und erstelle für jedes Repository eine Klasse in meinem Programm. Danach mache ich ein Klone oder Pull (je nachdem ob das Projekt bereits in dem Ordner ist oder nicht).

Wenn ich den Inhalt habe, nehme ich alle Logs mit dem Inhalt eines Projektes und erstelle eine Klasse dafür. Um die Geschwindigkeit zu erhöhen und den Rest des Programmes schneller zu starten, mache ich dies alles im Hintergrund als Thread.

Auch im Hintergrund als Thread werden die Repository-Icons mit «pscp» auf meinen Laptop kopiert. Ebenfalls wird ein Backup auf GitHub von den Repositories auf dem Nas erstellt und diese laufen ebenso im Hintergrund.



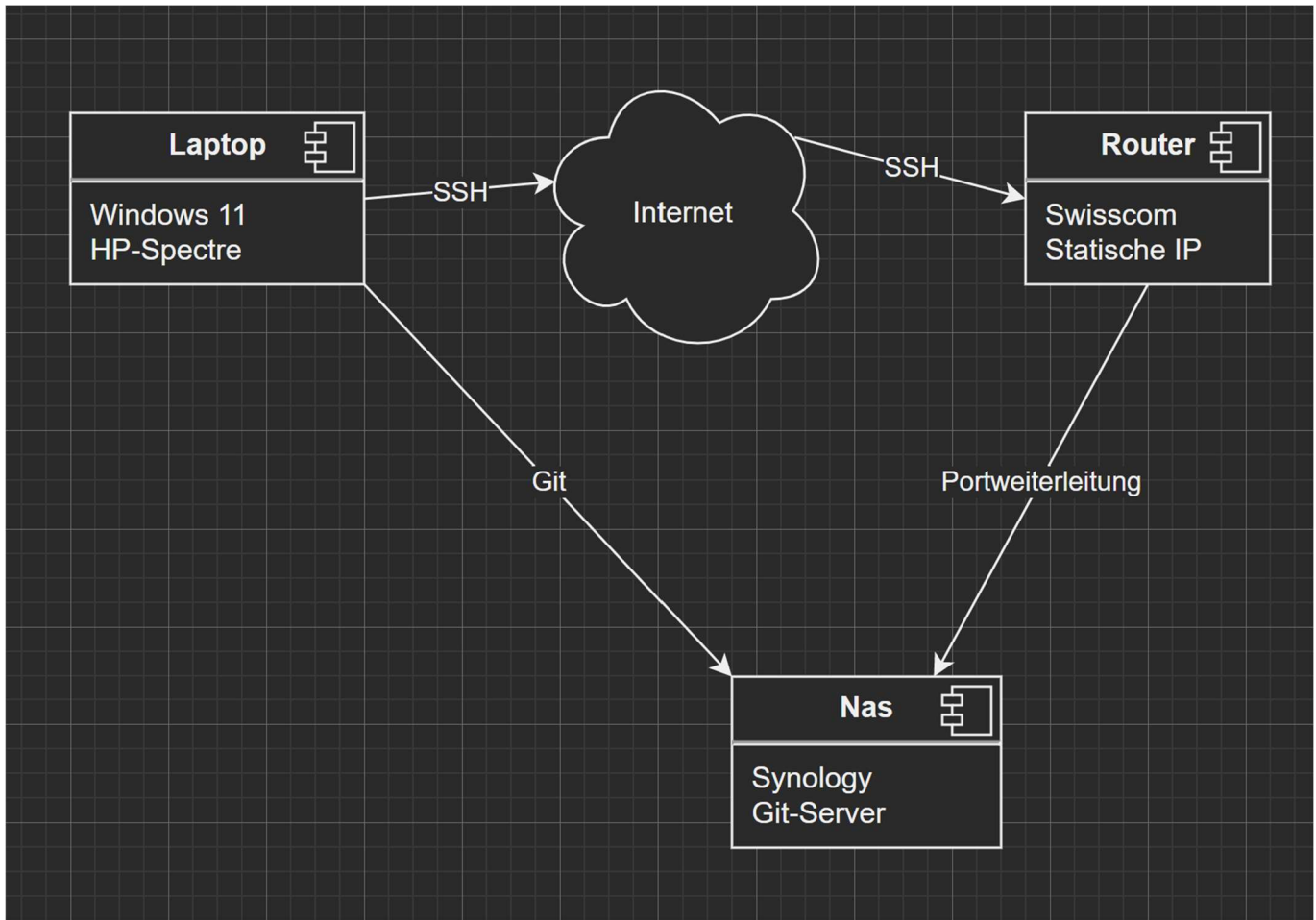


Abbildung 4: Skizze der System-Architektur

Der Aufbau der Verbindungen ist ziemlich einfach. Der Ursprung ist der Laptop, auf welchem das Programm läuft und an welchem auch der Benutzer arbeitet. Ich habe das Programm auf Windows 11 programmiert und es dementsprechend auch dafür ausgelegt. Auf Windows 10 könnte es funktionieren aber auf Linux oder Mac funktioniert es sicher nicht.

Um eine Verbindung mit dem Synology Nas aufzubauen, verbinde ich mich mit SSH auf meinen Router zuhause, welcher eine statische IP hat. Dieser leitet das Port 22 auf das Nas weiter.

Um nicht immer das Passwort eingeben zu müssen, habe ich ein SSH-Key eingerichtet, welcher das Passwort umgeht. Ich kann so auch git push/git pull ausführen, ohne mich manuell authentifizieren zu müssen.

## Vorschau

### Repositories

Wenn man das Programm startet, kommt als erstes ein Ladebildschirm, welcher angezeigt wird bis die Repositories geladen sind.

Nun erblickt man eine Übersicht mit allen Repositories. In dieser Übersicht kann man den Namen des Repositories, die Beschreibung und ein Icon sehen. Zudem hat man ein Zahnrad, welches einem zu den Einstellungen des Repositories bringt.

Da ich nicht für jedes Projekt ein Icon machen will, habe ich es so gemacht, dass wenn kein Icon angegeben wird, wird ein Icon erstellt, welches eine zufällige Farbe hat und nach einem Programm-Neustart immer noch dieselbe ist.

Über den Repositories hat man eine Zeile, in der man verschiedene Tools hat wie zum Beispiel eine Suche. Die Suche funktioniert so wie man es erwarten könnte, es überprüft, ob ein Repository-Name den Teil beinhaltet und zeigt dasjenige an bei welchen das zutrifft.

Es hat auch einen Knopf um ein neues Repository zu erstellen.

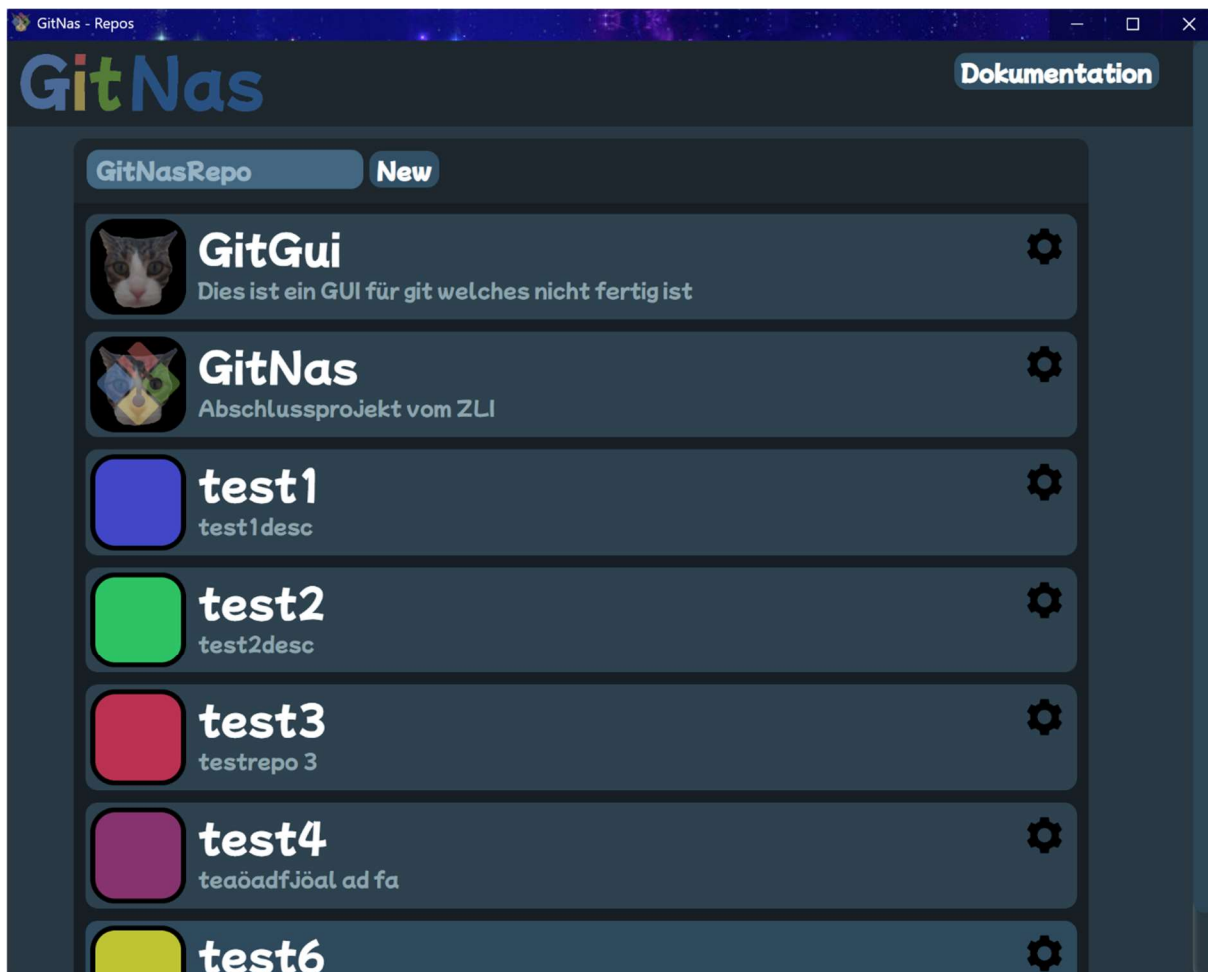


Abbildung 5: Repository-Übersicht

## Repository erstellen

Bei der Erstellung eines neuen Repository kann man 3 Dinge definieren. Der Name ist das Erste. Er darf nicht leer sein sowie nicht bereits besetzt sein. Das Zweite ist die Beschreibung, in welcher man kurz erklären kann, was das Projekt ist. Auch dies hat Einschränkungen, sie ist unkompliziert und lautet: Die Beschreibung darf nicht leer sein.

Als Letztes ist noch das Icon. Hier gibt es zwei Fälle. Der Erste ist, wenn man ein Icon auswählt, dann wird es zugeschnitten damit es quadratisch ist. Der Zweite ist, wenn man kein Icon auswählt, dann wird ein Icon mit einer zufälligen Farbe erstellt.

Nun kann man bestätigen und wird auf die Seite des Repositories weitergeleitet. Da aber das Repository leer ist, wird man auf eine Spezialseite weitergeleitet nämlich «leeres Repository».

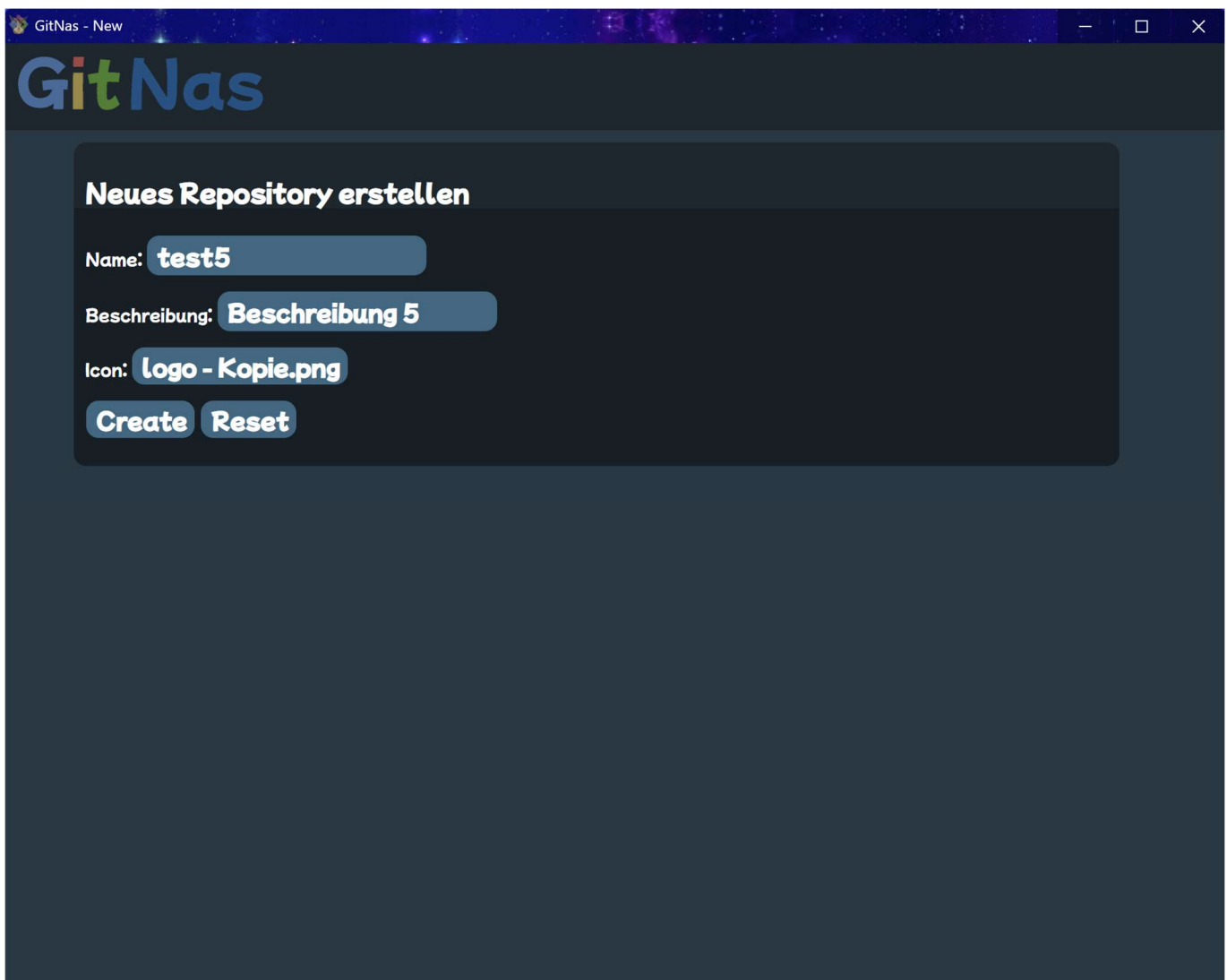
The image shows a web browser window titled 'GitNas - New'. The page has a dark theme with a blue header bar containing the 'GitNas' logo. Below the header, there is a form titled 'Neues Repository erstellen'. The form contains three input fields: 'Name:' with the value 'test5', 'Beschreibung:' with the value 'Beschreibung 5', and 'Icon:' with the value 'logo - Kopie.png'. At the bottom of the form, there are two buttons: 'Create' and 'Reset'.

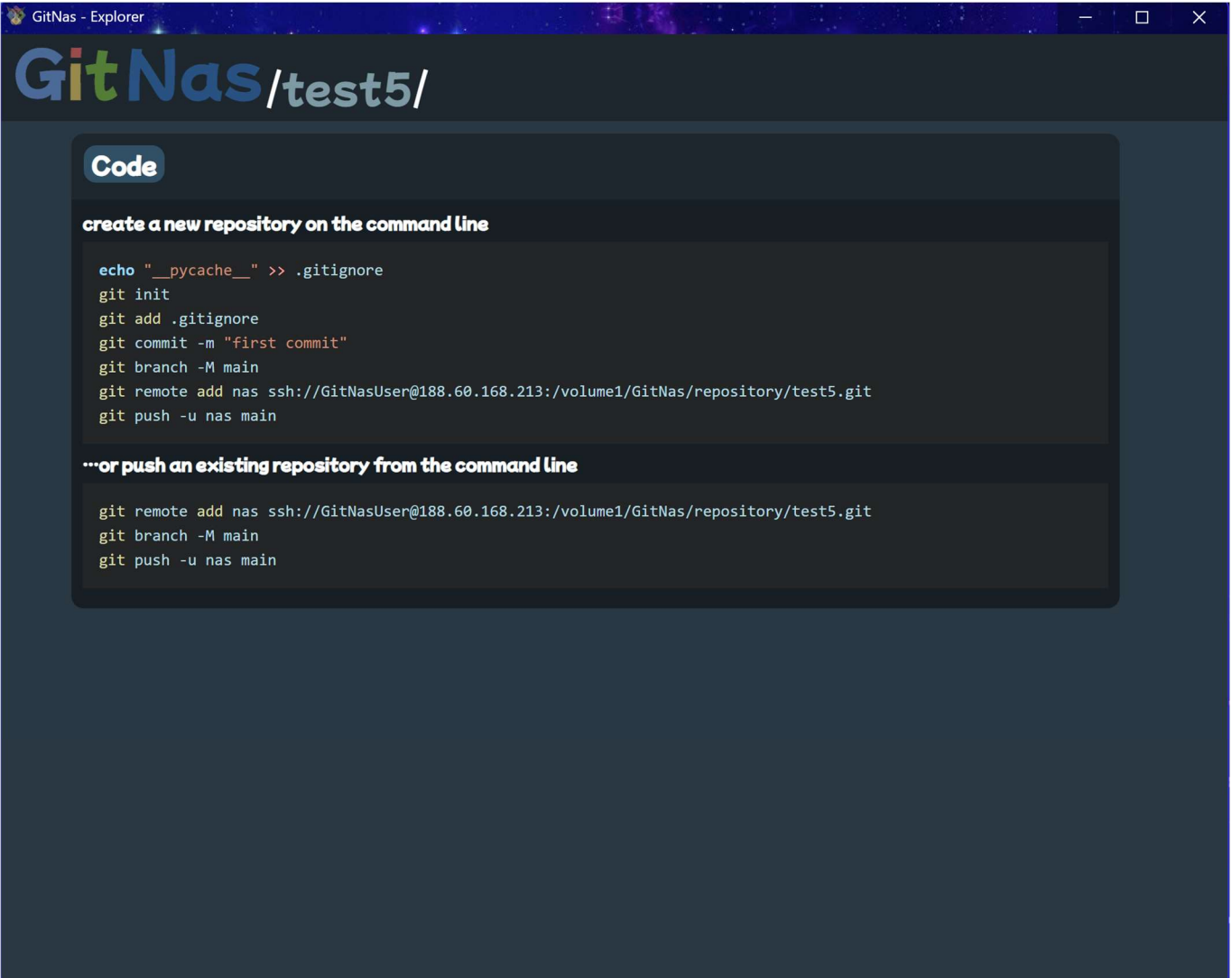
Abbildung 6: Neues Repository erstellen

## Leeres Repository

Dies ist die Seite, die angezeigt wird, wenn man in ein leeres Repository klickt. Die Seite besteht aus zwei Gruppen von Commands, welche man kopieren und im Terminal einfügen kann um Zeit zu sparen.

Die erste Gruppe von Commands ist dafür da, um ein neues Repository in einem leeren Ordner zu erstellen. Dabei übernimmt es das Erstellen vom Git-Ignore, initialisieren des Repositories, Master-Branch zu Main umbenennen, Remote-Punkt hinzufügen und pushen.

Die zweite Gruppe ist geeignet, wenn man bereits ein Repository hat aber nur lokal. Dabei übernimmt es drei Dinge. Remote-Punkt hinzufügen, Branch umbenennen und pushen.



```
GitNas - Explorer
GitNas/test5/

Code

create a new repository on the command line

echo "__pycache__" >> .gitignore
git init
git add .gitignore
git commit -m "first commit"
git branch -M main
git remote add nas ssh://GitNasUser@188.60.168.213:/volume1/GitNas/repository/test5.git
git push -u nas main

...or push an existing repository from the command line

git remote add nas ssh://GitNasUser@188.60.168.213:/volume1/GitNas/repository/test5.git
git branch -M main
git push -u nas main
```

Abbildung 7: Leeres Repository

## Explorer

Wenn man nun aber Files in einem Repository hat, kommt der Explorer zum Einsatz. In diesem kann man im Projekt navigieren und so zu den Files kommen.

Ganz oben auf der Seite kann man den Pfad zum aktuellen Ort sehen. Also zum Beispiel «GitNas/GitGui/web». Das Erste (GitNas) ist das Programm, GitGui ist das Repository und web ist der Pfad im Repository zur aktuellen Position. Dieser ist auch das Werkzeug, um zurück zu navigieren. Indem man auf den Schriftzug von GitNas klickt, kommt man immer zurück zu der Repository-Übersicht und dies von jedem Punkt aus.

Über den Files ist noch eine Werkzeug-Leiste, in welcher verschiedene Funktionen eingebunden sind.

Der Knopf «Code» ist dafür da, um einige Befehle zu kopieren. Der Erste ist die URL für eine SSH-Verbindung. Um einen Remote-Punkt hinzuzufügen kann man den zweiten Befehl nutzen. Um das Repository zu klonen ist der dritte Befehl geeignet und um etwas zu pushen ist der letzte Befehl gemacht.

Um die Commits in diesem Repository anzusehen, habe ich eine eigene Seite erstellt, welche man durch einen Klick auf «Commits» erreichen kann.

Neben dem Knopf für die Commits ist auch noch ein Knopf, um die Tags anzuzeigen.

Ein Knopf in der Werkzeug-Leiste ist, dazu da, die Branches zu sehen und mit einem Klick zu wechseln.

Um das Projekt im aktuellen Branch herunterzuladen, habe ich den Knopf «Download als ZIP» programmiert. Dabei wird der «.git»-Ordner nicht beachtet, da man es sonst klonen könnte und man so Speicherplatz spart.

Wenn man mit der Beschreibung nicht zufrieden ist oder das Icon ändern möchte, kann man dies in den Settings ausführen. Diese sind durch den Knopf «Settings» verlinkt.

Nun ist der Hauptteil des Explorer an der Reihe beschrieben zu werden. Die Files und Ordner. Die Reihenfolge der Files und Ordner ist alphabetisch, wobei zuerst die Ordner kommen und danach die Files. Die Ordner haben immer dasselbe Icon jedoch haben die Files je nach Dateiendung ein anderes Icon. Diese habe ich auf ein paar wenige begrenzt, welche ich häufig nutze und welche im Projekt vorkommen.

Durch einen Klick auf ein File wird man auf den Editor weitergeleitet. Wenn man in einen Ordner klickt, bleibt man im Explorer, hat jedoch dann den Ordner geöffnet und kann nun dessen Inhalt sehen sowie auch einen anderen Pfad.

Ich hatte vor das letzte Änderungsdatum ebenfalls einzublenden, jedoch ist dies mit meiner technischen Umsetzung unmöglich, da ich immer jedes Projekt bei Programmstart pulle und somit das Änderungsdatum als aktuelles Datum erscheint.

Unter den Files kann man, falls ein README.md vorhanden ist, das Readme des Projektes sehen. Es erscheint jedoch nicht in Rohform, sondern «gerendert». Anstatt «#» sieht man einen grossen Titel.



Abbildung 8: Explorer

## Commits

Dies ist die Seite, auf der man die Commits, welche man in diesem Projekt getätigt hat, sehen kann.

Man ist immer noch in einem Ordner (man erkennt es am Pfad) und mit einem Klick auf «Explorer» in der Werkzeug-Leiste kommt man zurück zum Ordner, in welchem man zuvor war.



Ebenfalls in der Werkzeug-Leiste ist der «Code»-Knopf, welcher dasselbe macht wie jener im Explorer. Was nicht im Explorer ist, ist eine Anzeige der Anzahl der gesamten Commits in diesem Projekt.

Unter der Werkzeug-Leiste sind die Commits sortiert aufgelistet. Die neusten Commits sind zuoberst und dementsprechend die ältesten zuunterst. Man kann verschiedene Informationen aus den Commits herauslesen. Die Message, welche als «Titel» vom Commit dargestellt wird. Darunter ist der ganze Hash, um in der Konsole mehr Informationen zu bekommen. Rechts ist die Uhrzeit sowie das Datum des Commits.

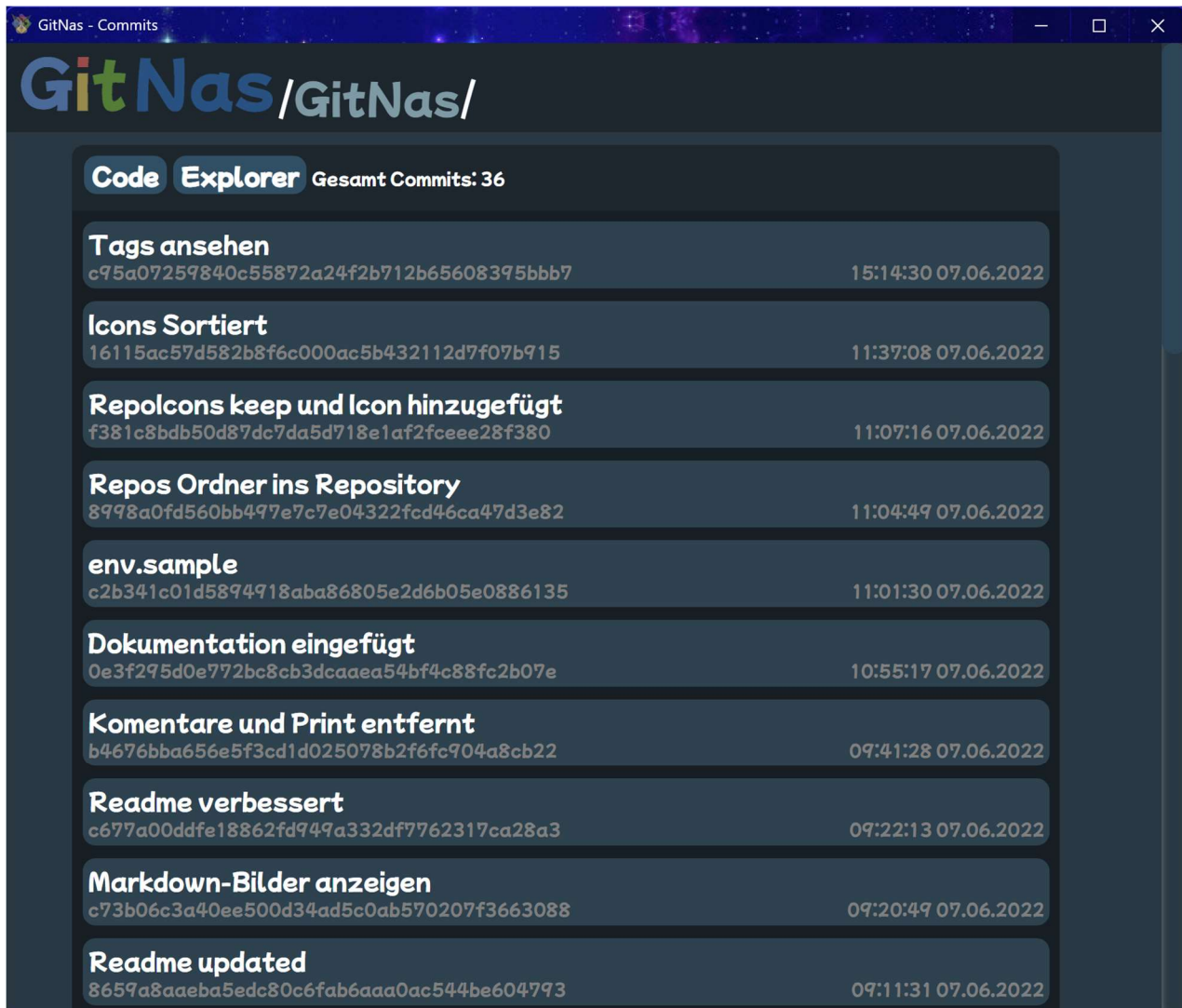


Abbildung 9: Commits

## Repository-Einstellungen

In den Repository-Einstellungen kann man dasselbe machen, wie beim Erstellen, jedoch nicht den Namen ändern.

Man sieht den Namen des Repositories im Pfad damit man sichergehen kann, dass es das Richtige ist.

Die Beschreibung ist aufgeführt mit der Beschreibung, die man zuvor hatte.

Das Icon kann man mit einem Klick auf «Select neu auswählen». Wenn man eines ausgewählt hat, erscheint eine Vorschau unter dem Knopf wie es am Ende aussieht. Wenn man jedoch auf «neues Icon» klickt, wird, wie beim Erstellen, ein Bild mit einer zufälligen Farbe generiert und ebenfalls in der Vorschau eingefügt.

Wenn man zufrieden mit den Änderungen ist, kann man auf «Update» klicken und die Änderungen werden übernommen.

Falls man die Änderungen nicht speichern möchte, kann man auf «Verwerfen» klicken und es wird die Seite geöffnet, welche zuletzt offen war.

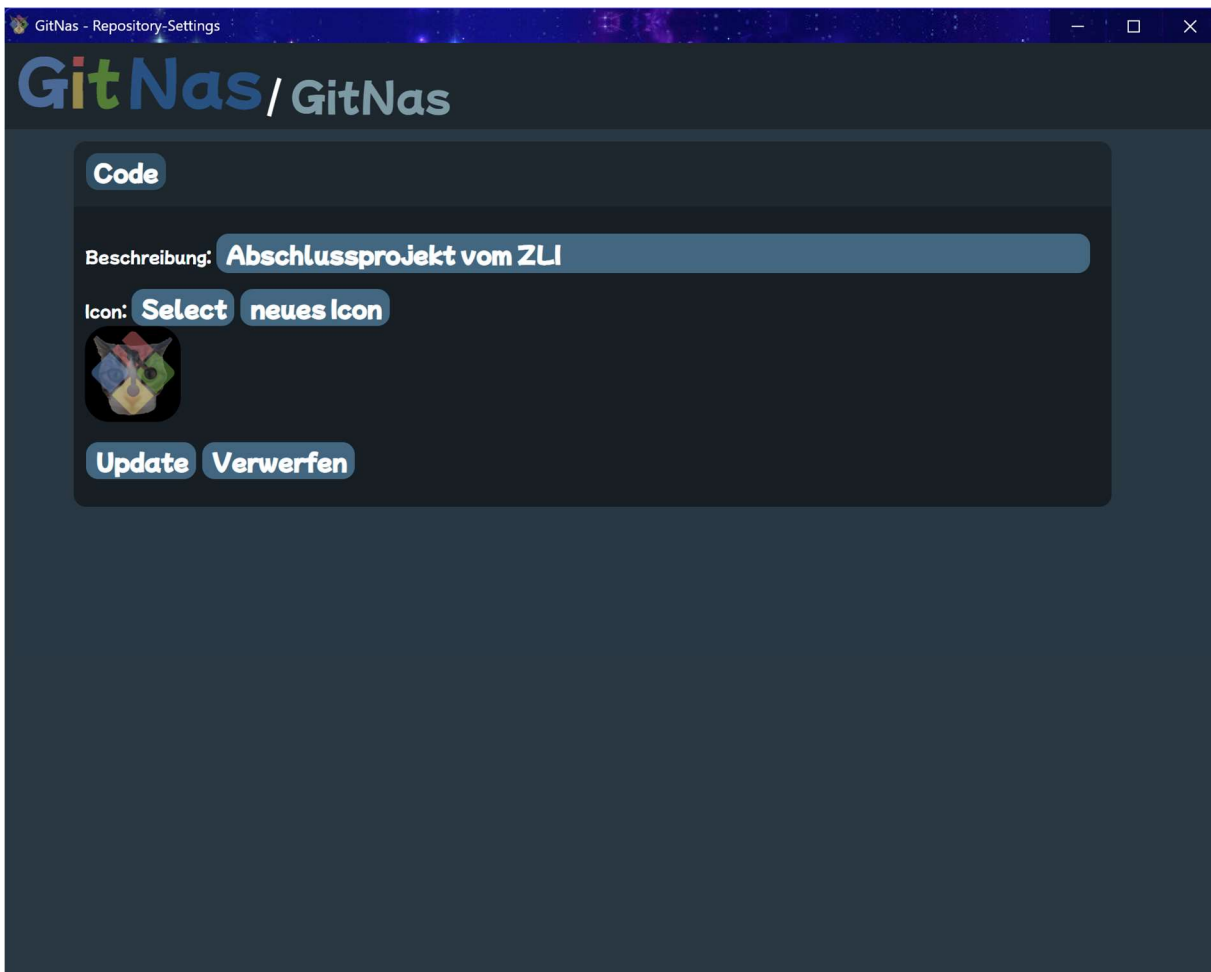


Abbildung 10: Einstellungen

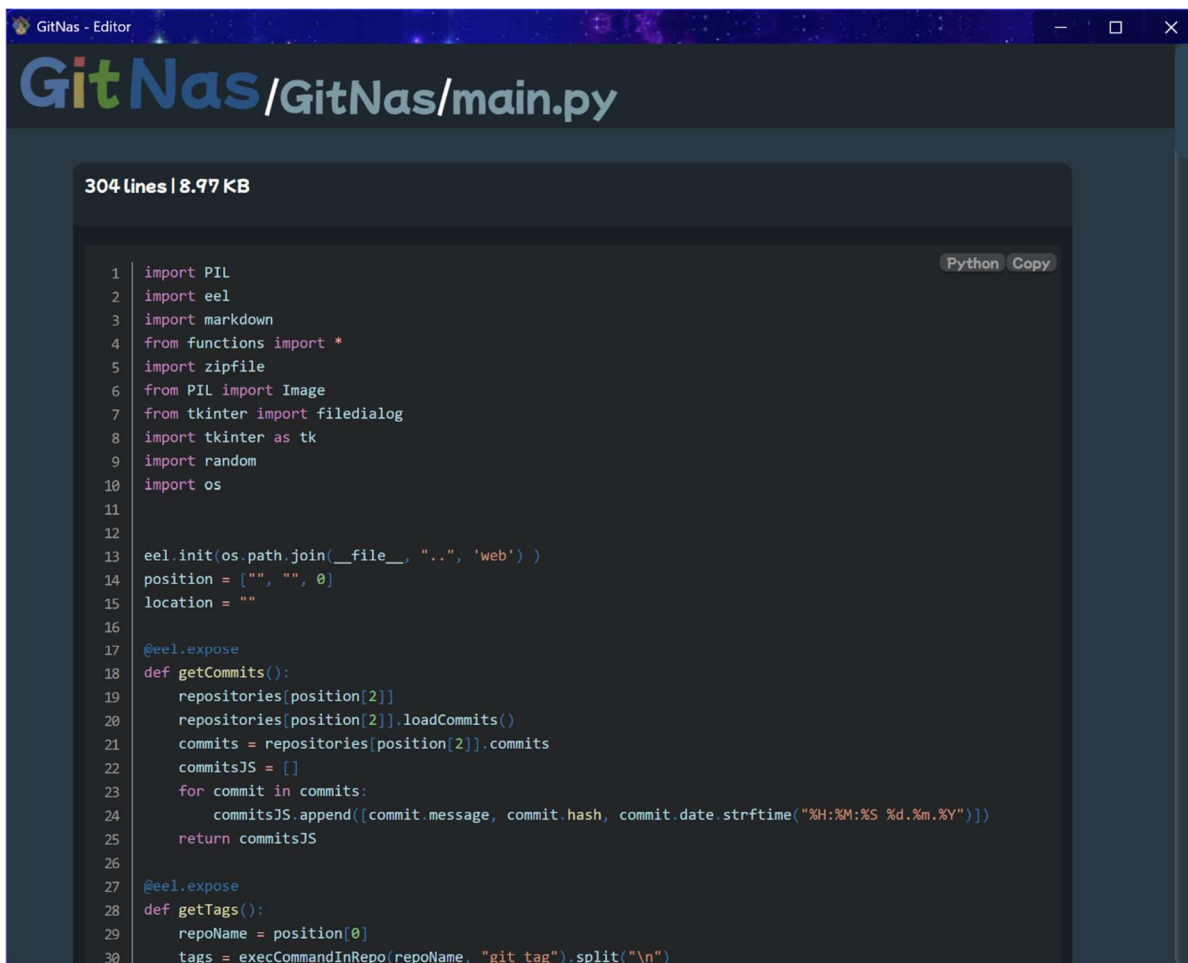
## Editor

Im Editor wird ein File angezeigt, welches man in dem Explorer geöffnet hat. Wenn es ein File ist, welches Text beinhaltet, wird es angezeigt mit einer Syntax-Highlight, welche mir von Prism.js bereitgestellt wird und ich nur ein wenig die Farben geändert habe, damit sie zu meinem Theme passen.

Auch diese Seite hat eine Werkzeug-Leiste mit je nachdem einem oder zwei Elementen. Wenn es ein Text-File ist, sind es zwei Elemente. Nämlich die Anzahl Linien und die Grösse des Files in Byte, Kilobyte, Megabyte oder Gigabyte.



Abgesehen von einem Syntax-Highlight hat der Text eine Funktion um ihn zu kopieren, zudem sieht man die Zeilennummer. Man kann ebenfalls den Typen des Files sehen.



```

1  import PIL
2  import eel
3  import markdown
4  from functions import *
5  import zipfile
6  from PIL import Image
7  from tkinter import filedialog
8  import tkinter as tk
9  import random
10 import os
11
12
13 eel.init(os.path.join(__file__, '..', 'web'))
14 position = ["", "", 0]
15 location = ""
16
17 @eel.expose
18 def getCommits():
19     repositories[position[2]]
20     repositories[position[2]].loadCommits()
21     commits = repositories[position[2]].commits
22     commitsJS = []
23     for commit in commits:
24         commitsJS.append([commit.message, commit.hash, commit.date.strftime("%H:%M:%S %d.%m.%Y")])
25     return commitsJS
26
27 @eel.expose
28 def getTags():
29     repoName = position[0]
30     tags = execCommandInRepo(repoName, "git tag").split("\n")

```

Abbildung 11: Editor

## Tags

Auf dieser Seite kann man die Tags sehen, welche man in diesem Repository erstellt hat.

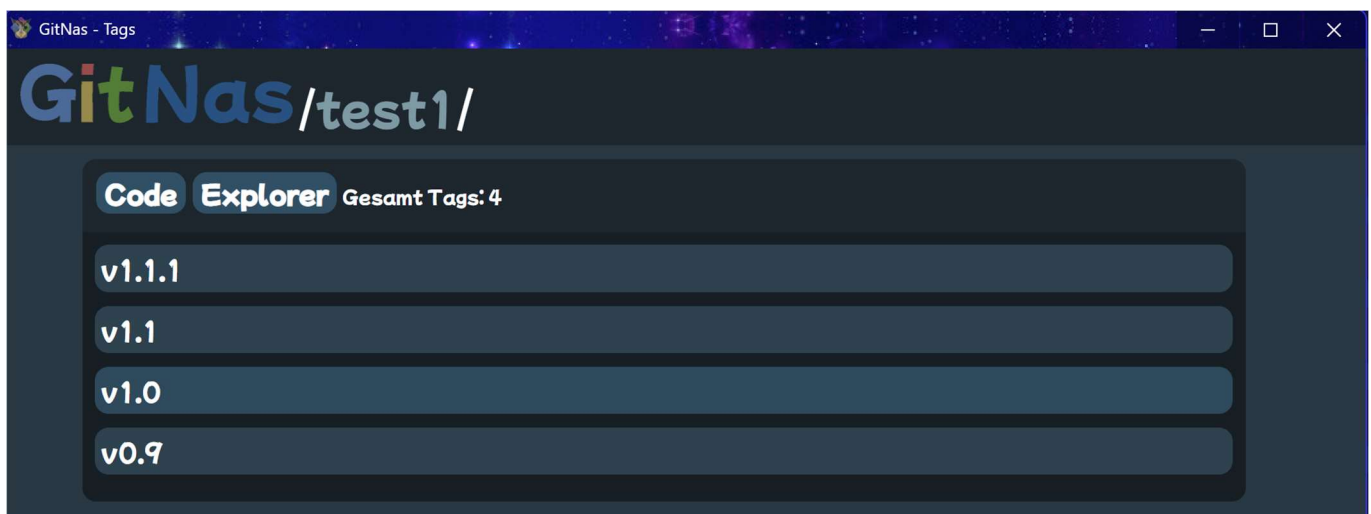


Abbildung 12: Tags

## Arbeitspakete-Überprüfung

Arbeitspaket	Status	Kommentar
Ordnerstruktur aufsetzen	Erledigt	
Klasse erstellen	Erledigt	Aufgrund der Performance habe ich nicht alle Klassen wie geplant erstellt
HTML und CSS	Erledigt	
Funktionen schreiben	Erledigt	
Repositories ansehen	Erledigt	
Beschreibung bearbeiten	Erledigt	
JS für Repository	Erledigt	
Explorer anzeigen	Erledigt	
File-Icons anzeigen	Erledigt	
Pfad erstellen	Erledigt	
Editor anzeigen	Erledigt	
Pfad und Sonderzeichen	Erledigt	
Daten über das File	Erledigt	Letztes Änderungsdatum ist nicht möglich
Neues Repository erstellen	Erledigt	
Leeres Repository Übersicht	Erledigt	
Commits anzeigen	Erledigt	
Branches anzeigen	Erledigt	
Branches wechseln	Erledigt	Probleme mit den Commits
Repository als Zip	Erledigt	
Repository-Icon	Erledigt	Hat länger gedauert als erwartet
Repository Suchen	Erledigt	Im Nachhinein zu Arbeitspakete hinzugefügt
Readme anzeigen	Erledigt	Im Nachhinein zu Arbeitspakete hinzugefügt
About einbetten	Nicht erledigt	Nicht gemacht, weil es Überflüssig ist
Dokumentation einbetten	Erledigt	Sehr schnell abgeschlossen
Tags	Erledigt	
Releas	Nicht erledigt	Ist kein normales Git-Feature

## Reflexion

Mir ist der Start erstaunlich leichtgefallen. Ich habe gedacht, dass ich am Anfang Mühe habe es im Verlauf des Projektes immer leichter wird, aber genau das Gegenteil ist eingetreten.

Alles in allem hat es mir Spass gemacht und ich konnte auch viel dabei lernen. Was mich aber noch glücklicher über das Resultat macht ist, dass ich es auch nutzen werde für persönliche Projekte und es nicht gemacht habe.

Ich habe mit bei der Planung zu wenig vorgenommen. Ich konnte noch ein paar zusätzliche Features einfügen wie zum Beispiel die Suche. Die Suche war in den «User-Cases» aber ich habe gedacht, dass ich zu wenig Zeit hätte, um das noch zu machen. Zudem irrte ich mich in der Schwierigkeit dieser Such-Funktion. Dies umzusetzen war deutlich einfacher als erwartet und konnte so schnell umgesetzt werden.

Jedoch habe ich auch einige Features weggelassen. Unter anderem die About-Seite. Diese habe ich aber übersprungen, da es ein Programm für mich ist und somit eine About-Seite über mich und das Programm überflüssig, ist da ich weiss, was ich gemacht habe und wer ich bin.

## Testprotokoll

Bezeichnung	Repository-Übersicht
Beschreibung	Alle Repositories werden angezeigt Icon wird angezeigt Beschreibung wird angezeigt Suche funktioniert Knopf, um ein neues Repository zu erstellen funktioniert Link zur Dokumentation funktioniert Knopf zum Bearbeiten eines Repositories funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	10:43 08.06.2022
Ergebnis	Alles ausser der Dokumentation funktioniert. Dies aufgrund dessen, dass ich die Dokumentation noch nicht fertig habe.

Bezeichnung	Explorer
Beschreibung	Link-Knöpfe funktionieren Pfad funktioniert Ordner/Files werden sortiert angezeigt Files haben die richtigen Icons Readme wird angezeigt Download als Zip funktioniert Code-Knopf funktioniert Branchwechsel funktioniert File/Ordner öffnen funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	08:49 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Editor
Beschreibung	Pfad funktioniert File-Informationen stimmen (Linien/Grösse/Type) Zeilennummern werden angezeigt Syntax-Highlight funktioniert Kopieren funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	09:18 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Neues Repository
Beschreibung	Eingabeüberprüfung funktioniert Icon auswählen Icon automatisch generieren Erstellung funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	09:30 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Repository-Einstellungen
Beschreibung	Icon auswählen funktioniert Icon generieren funktioniert Icon-Vorschau funktioniert Verwerfen funktioniert Updaten funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	10:03 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Leeres Repository
Beschreibung	Code-Knopf funktioniert Pfad funktioniert Commands werden angepasst Kopieren funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	10:26 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Commits
Beschreibung	Knöpfe funktionieren Gesamt-Anzahl stimmt Pfad funktioniert Alle Commits werden angezeigt Sortierung nach Zeit funktioniert Commit-Hash und Zeit werden angezeigt
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	10:43 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Tags
Beschreibung	Knöpfe funktionieren Gesamt-Anzahl stimmt Pfad funktioniert Alle Tags werden angezeigt Sortierung nach Version funktioniert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	10:57 13.06.2022
Ergebnis	Alles funktioniert

Bezeichnung	Allgemein
Beschreibung	Repositories werden bei Start aktualisiert Backup wird automatisch durchgeführt Icons werden aktualisiert
Testschritte	Alles nacheinander testen
Erwartetes Ergebnis	Alles funktioniert
Testdatum	11:08 13.06.2022
Ergebnis	Alles funktioniert

## Quellen

Settings-SVG: <https://img.icons8.com/material-sharp/344/settings.png>

Icons: <https://marketplace.visualstudio.com/items?itemName=PKief.material-icon-theme>

Syntax-Highlight: <https://prismjs.com/>

Hilfe bei Problemen: <https://stackoverflow.com/>

Python-Packages: <https://pypi.org/>