# Teilnehmer/innen des Teams:

|  |  |
| --- | --- |
| Klasse:  BI 20a | Team:  GrAf |

# Anforderungsdefinition (Meilenstein A)

|  |  |
| --- | --- |
| „GitHub-GUI“ | |
| **Fachlicher Inhalt:**  (Allgemeine Beschreibung) | Nutzen: Unsere Applikation "Github-GUI" soll es durchschnittlichen Internet Nutzern ermöglichen, ihre Projekte ganz einfach, schnell und ohne den Nutzen von einem CMD auf GitHub raufzuladen und zu aktualisieren. Unser Programm Inspiration von GitHub Desktop nehmen, währen wir unsere eigenen Ideen einfügen und ausführen. Man sollte das Programm ganz einfach und ungehindert nutzen können ohne sich mit den lästigen Git befehlen auseinandersetzen zu müssen. So kann man sich sehr viel Zeit sparen, denn man muss Git nicht mehr lernen.**Automation:**  Das Programm automatisiert Git-Verwaltung durch GUI-Interaktionen.  **Details:**   * **Der Nutzer kann zum Ort des Repositories wechseln, bzw. bestimmten wo ein Repository geklont wird** * **Nach dem Klonen befindet mich gleich im Repository** * **Es können Commits mit Kommentar erstellt werden** * **In einer Listbox können Angaben zum Working Tree (Git status) sowie eine Liste der letzten Commits (Git log) betrachtet werden.** * **Es können Pull-Requests erstellt werden** * **In einem separaten Fenster können alle Branches betrachtet werden, der derzeitige Branch gewechselt und neue Branches erstellt werden**   **GitHub Repository: https://github.com/Sraosha47/GitHub-GUI**  **Erkenntnisse aus der Machbarkeitsabklärung in Windows Powershell:**  Test ob und wie der Output von Git-Befehlen ausgegeben wird.  **Ein Bild, das Text enthält.  Automatisch generierte Beschreibung** |
| **MUSS**  **Kriterien:**  (Konkrete Features, die umzusetzen sind) | **Folgende Features sollen implementiert werden, um einen produktiven Ablauf sicherzustellen:**   * Pfadangabe und Auswahl * Klonen * Commit * Git Pull * Zweites Fenster Branches * Erstellen und wechseln Branches * Merging |

|  |  |
| --- | --- |
| **KANN**  **Kriterien:**  (Konkrete Features, die optional sind) | **Folgende Features können zusätzlich implementiert werden: (Varianten, Kreativität)**   * Ausgabe Git status * Ausgabe Git log * Pull-Requests erstellen * Erstellen eines neuen Repositories * Automatisches Anmelden bei erstem Commit * Git fetch * Git reset (Daten zurücksetzen, solange noch nicht gepusht wurde) * Git revert (entfernen eines Commits) |

## Planung Meilensteine (LB1 / LB2)

|  |  |  |  |
| --- | --- | --- | --- |
| *MS* | *Tätigkeit / Abgabe* | *Soll-Datum* | *Ist-Datum* |
| A | ProjektstartTeam Bildung, Kollaborationsplattform, GitHub Repos \*, LehrerzugangWahl / Ausarbeitung der Anforderungsdefinition Abnahme Anforderungsdefinition durch Lehrperson | 28.09.21 | 28.09.21 |
| B | Teamaufgabe 1:Abgabe: Lösungsdesign  (Funktionsmodell / GUI / PAP / Storyboard) | 28.09.21 | 05.10.21 |
| B2 | Teamaufgabe 2: (Nur LB2)Abgabe: Testvorschrift und Testfälle | 05.11.21 | 09.11.21 |
| C | Einzelaufgabe 2 (LB1) / 3 (LB2):Abgabe Programmcode und DokumentationFachgespräch Projektabnahme | 09.11.21 | 09.11.21 |
| C2 | Einzelaufgabe 4: (Nur LB2)Abgabe: Ausgefüllter Systemtest | 09.11.21 | 09.11.21 |

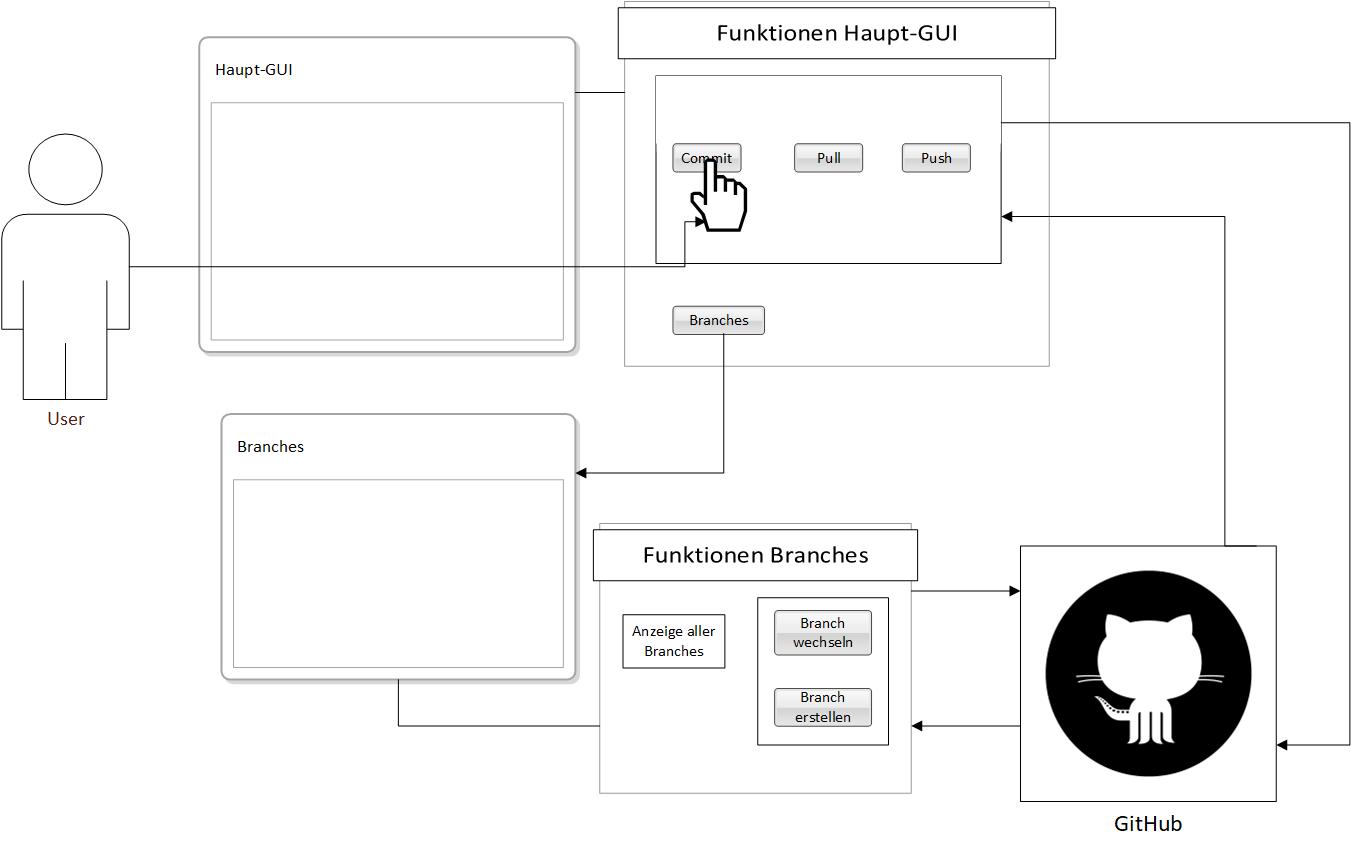
\*) Öffentliche GitHub-URLs im Ablageordner auf dem BSCW ablegen! (pro Team)

Namenskonvention URL: **M122\_Klasse\_Thema\_Name\_Name**

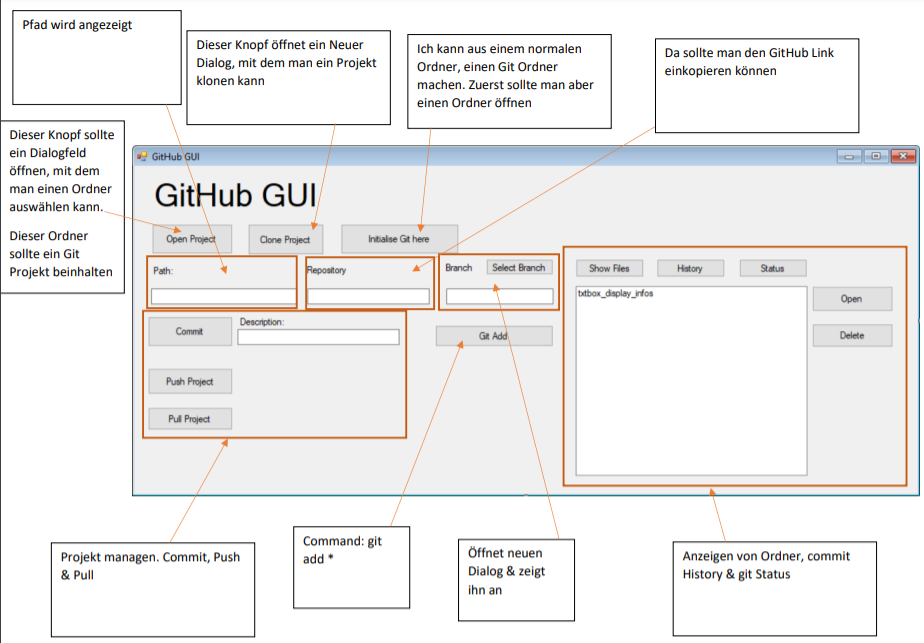
# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

## Schematische Darstellung der Funktionalität, sog. Funktionsmodell

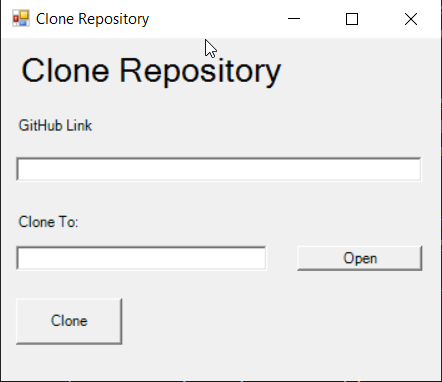


## Graphische Benutzer Schnittstelle (GUI) zur Konfiguration des Ablaufs



## Ablauf der Automation

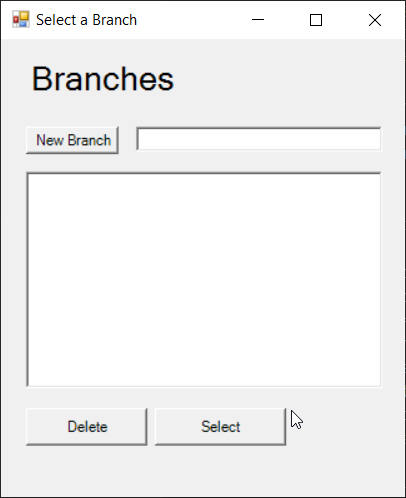
Storyboard



Man hat die Option ein Bestehenden Repository von GitHub Zu klonen. Der Link kommt in die TextBox weiter oben.

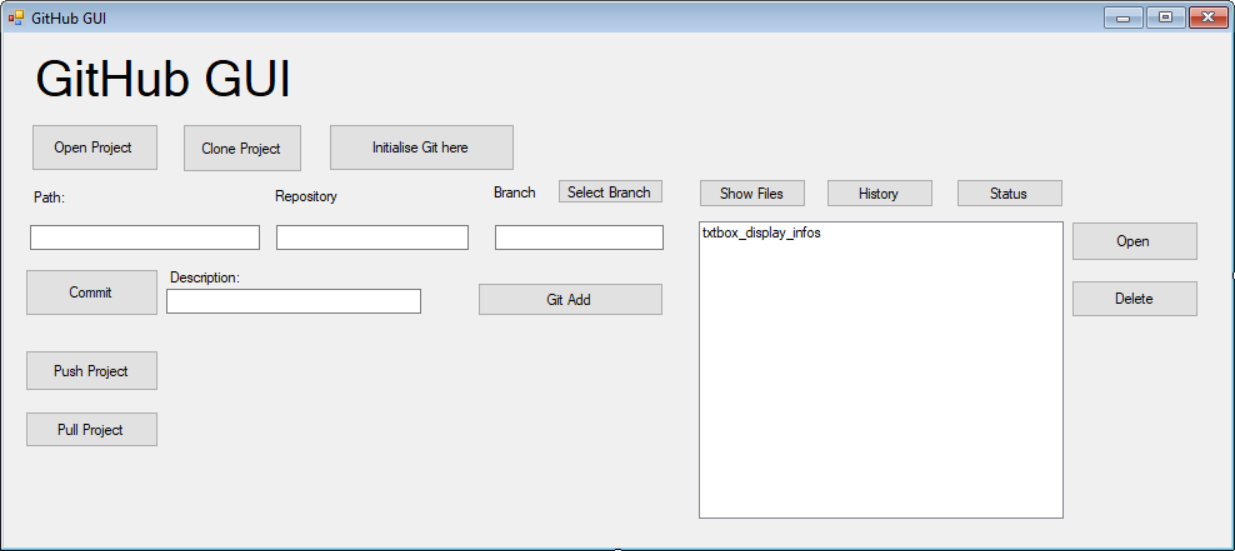
Jetzt kann man den Zielpfad auswählen.

Das Repository wird jetzt in den Zielordner geklont.

Die verschiedene Branche werden ausgewählt werden können.

In «New Branch» werden neue Branches erstellt werden können. In der TextBox Links, wird der Name reingeschrieben

Auch kann man Branches Löschen und sie auswählen.



Im Haupt Fenster wird das Repository verwaltet.

In «Commit» kann man sein Repository commiten.

Push / Pull Project wird das Projekt Aktualisert / Raufgeladen

Weitere Informationen zu den Funktionen kann man weiter oben sehen.

# Testvorschrift (LB2 Meilenstein B2: Teamaufgabe 2)

Siehe GitHub Repository (Link auf BSCW)

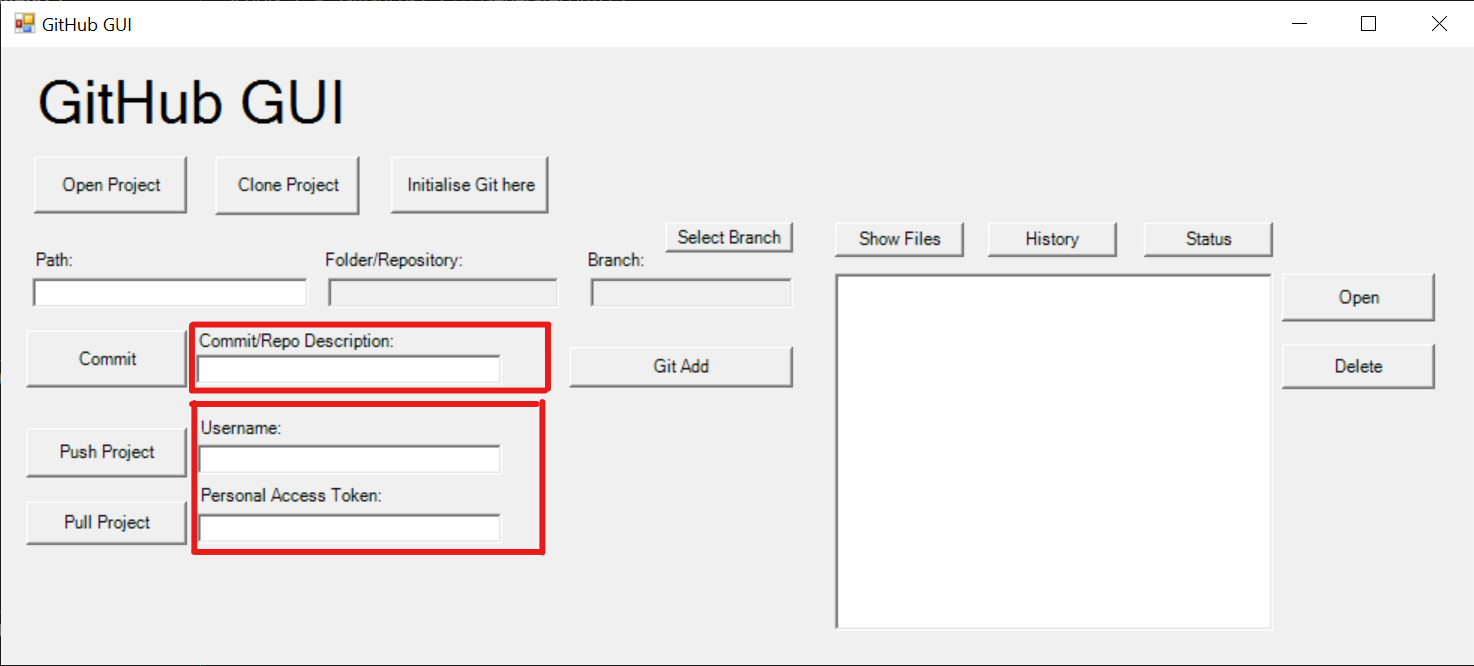
# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

# Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

Namenskonvention URL: **M122\_BI20a\_GitGui\_Afkhami\_Greil**

## Umfang / Abgrenzung / Änderungen gegenüber Design

### Änderungen

GitGui.ps1:  


* Das Label der txtbox\_dexcription wurde geändert, da es nun auch für die Beschreibung eines neuen initialisierten Gits verwendet wird.
* Labels und Textboxen für GitHub-Autorisierung wurden hinzugefügt. Momentan werden diese nur für die Initialisierung eines neuen Gits verwendet.

GitGui\_Branches:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Select-Button wurde in Switch umbenannt und mit dem Delete-Button vertauscht.
* Neuer Button Version Control, der alle Commits anzeigt, zu denen ebenfalls gewechselt werden kann.
* Neuer Button Refresh. Momentan redundant. Wird bei nächster Version entfernt.

Git\_Clone\_Repository:

Keine Änderungen

## Funktionalität der Implementation.

Zusätzlich zu der Inline-Dokumentation sind hier folgende Funktionen / Eventhandler detailliert beschrieben:

...

Ausführliche Beschreibung der internen Funktionen (Eventhandler wie z.B: $btn\_ok\_Click), der Parameter und der Rückgabewerte

**Struktogramm** (BET / SYS) oder **UML Aktivitätsdiagramm** (API)

# Betriebsdokumentation (Meilenstein C: individuelle Aufgabe 3)

Für Administrator und Benutzer wird folgende Anleitung ausgeliefert ...

## Installationsanleitung für Administratoren

Das Programm ist folgendermassen zu installieren und konfiguriert ...

## Bedienungsanleitung für Benutzer

Das Programm ist folgendermassen zu bedienen ...