# 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 |

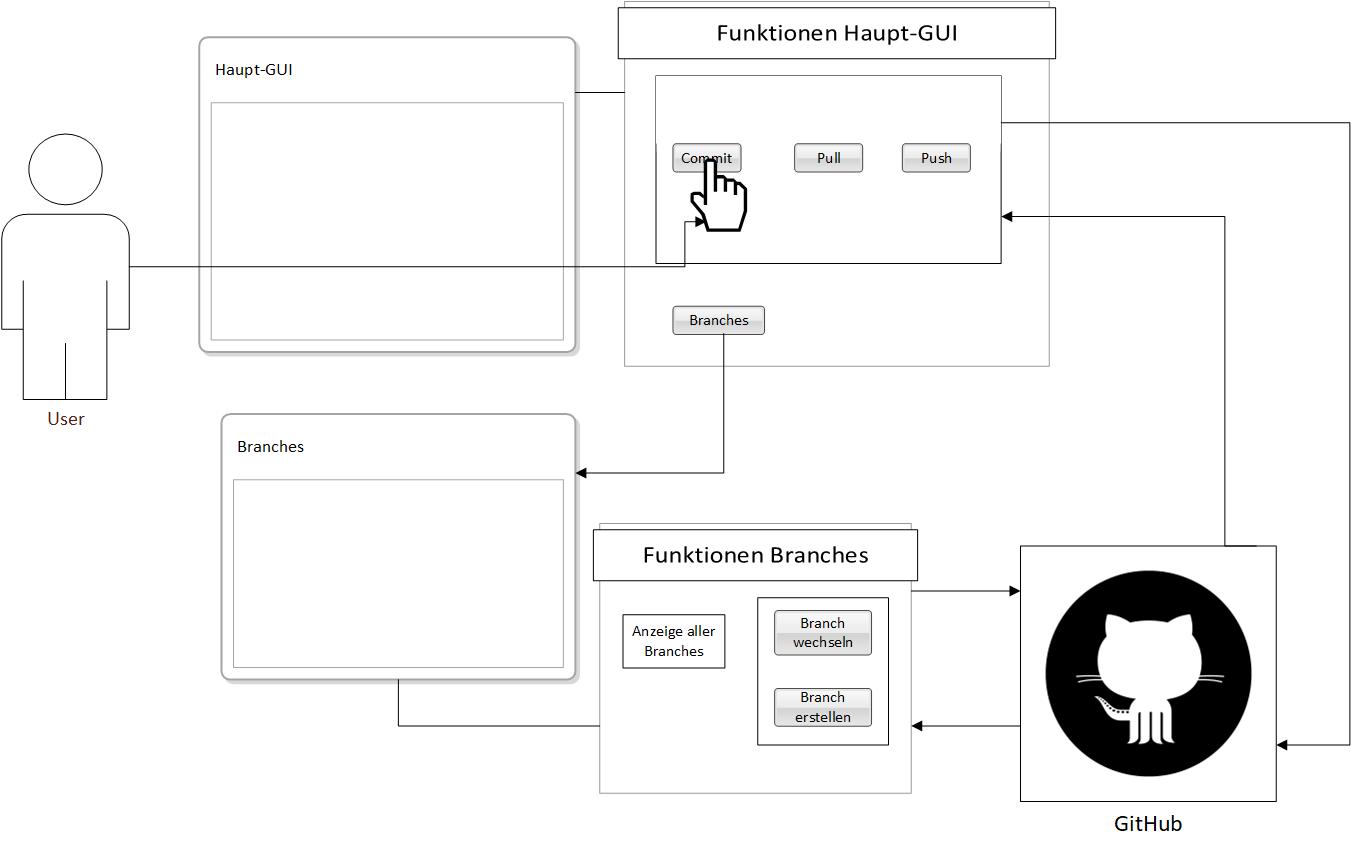
\*) Öffentliche GitHub-URLs im Ablageordner auf dem BSCW ablegen! (pro Team)

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

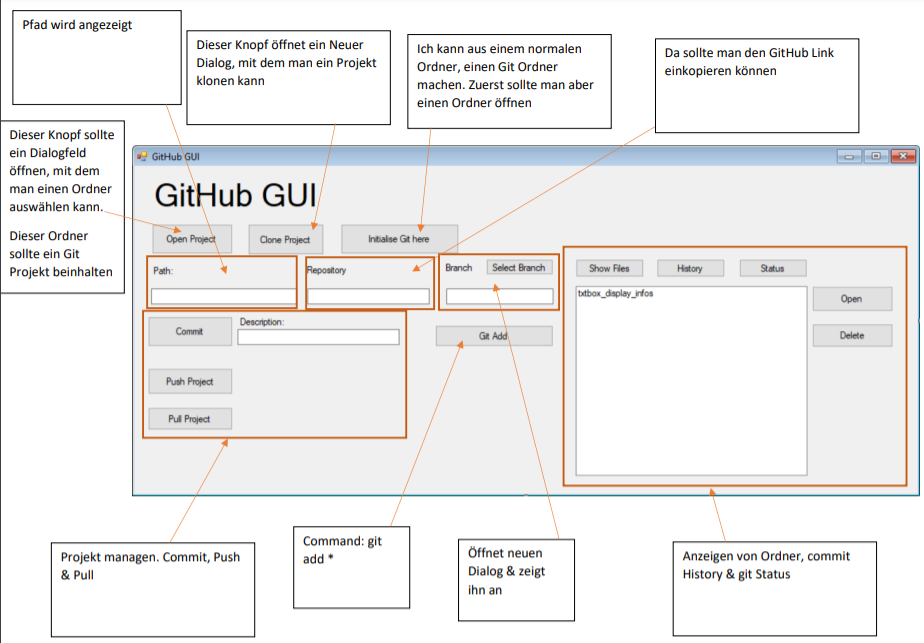
# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

## Schematische Darstellung der Funktionalität, sog. Funktionsmodell

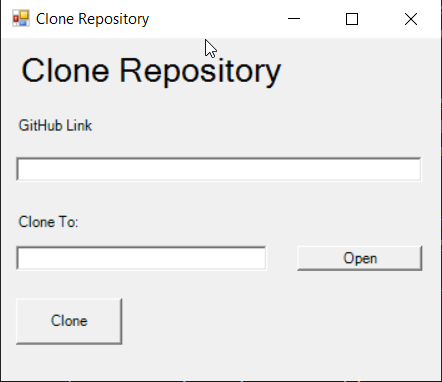


## Graphische Benutzer Schnittstelle (GUI) zur Konfiguration des Ablaufs



## Ablauf der Automation

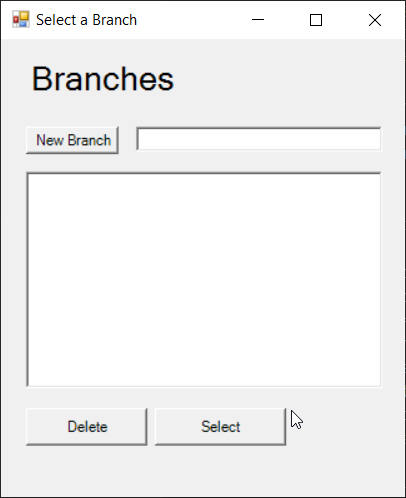
Storyboard



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

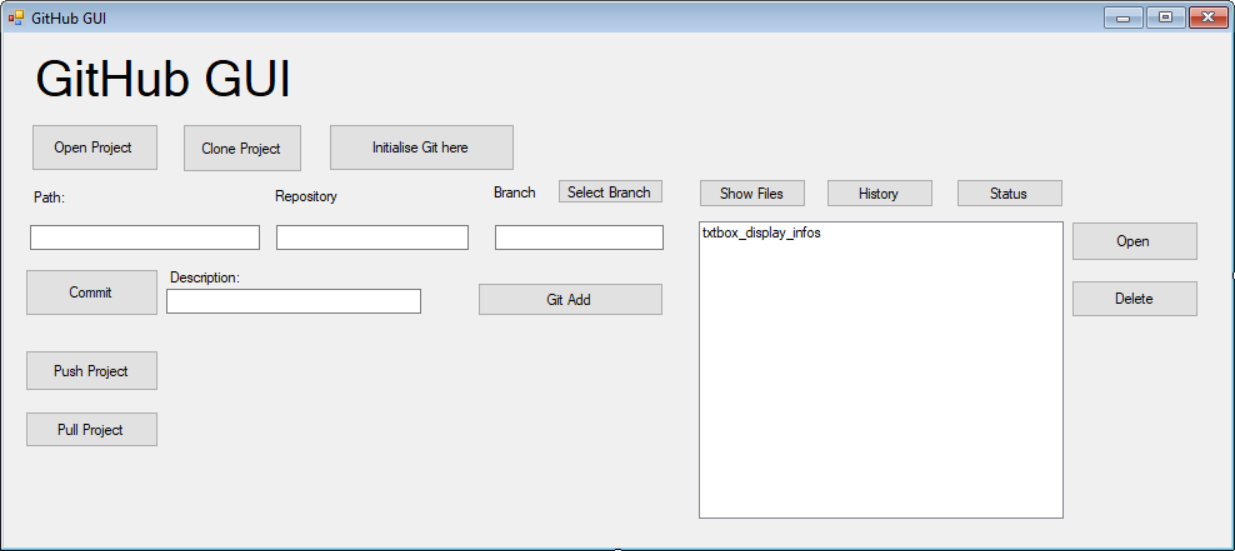
Jetzt kann man den Zielpfad auswählen.

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 BSCW

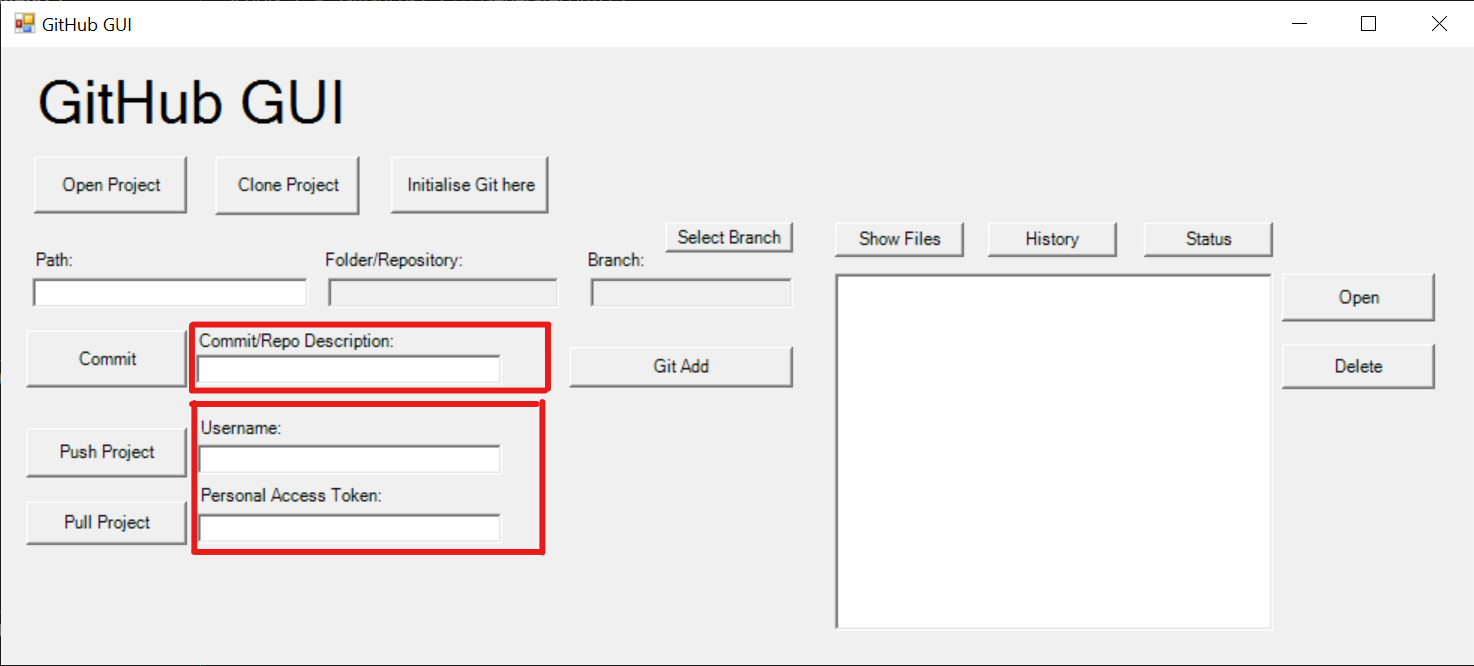
# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

# Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

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

## 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 am GUI

## Funktionalität der Implementation.

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

### GitGui.ps1

**Variablen:**

**[bool]$global:files\_loaded = $false**

Dient dazu anzugeben, ob die Objekte in der Listbox von GitGui.ps1 Daten des ausgwählten Ornders sind.

**Funktionen:**

**global:Get-GitStat**

Leert die Listbox des GitGui, speichert die Information von git status in ein Array namens $status und fügt die Items des Arrays der Listbox hinzu.

Setzt die variable $global:files\_loaded zu $false.

**global:Show-GitFiles**

Löscht alle Objekte der Listbox und fügt alle Dateien des ausgewählten Ordners und dessen Unterordner der Listbox zu. Falls der Ordner keine Dateien enthält zeigt die Listbox «No files in this location» an.

**New-WinObject**

Funktion zum leichteren erstellen von Windows Forms Objekten. Da die letzte Zeile alle erstellten Objekte dem bereits erstellten Fenster hinzufügt ist ihr Scope lokal und ich musste sie jedem Skript einzeln hinzufügen. Ich wette es gibt hierfür aber ebenfalls eine Lösung.

**Event Handler:**

**$btn\_branchGUI\_clicked & $btn\_cloneGUI \_clicked**

Öffnen die anderen Skripts.

**$btn\_open\_project\_clicked**

1. Öffnet Folder Browser
2. Setzt den Pfad zum ausgewählten Objekt(Set-Location)
3. Zeigt den Pfad, Namen des Repositorys und den Branch in der entsprechenden Textbox an. Falls es sich nicht um ein Repository handelt, wird es in der Textbox Branch angezeigt «Not a repository»
4. Führt Show-GitFiles aus

**$btn\_show\_files\_clicked**

Führt Show-GitFiles aus

**$btn\_open\_file\_clicked**

Falls $files\_loaded = $true: öffnet ausgewählte Datei in der Listbox

**$btn\_remove\_file\_clicked**

Falls $files\_loaded = $true: löscht ausgewählte Datei in der Listboxund führt Show-GitFiles aus.

**$btn\_show\_history\_clicked**

Leert die Listbox und füllt sie mit Informationen zu allen bisherigen Commits (git log) und setzt $files\_loaded zu $false.

**$btn\_show\_status\_clicked**

Führt Get-GitStat aus.

**$btn\_pull\_clicked**

Führt git fetch, git pull und Get-GitStat aus.

**$btn\_git\_add\_clicked**

Führt git add \* und Get-GitStat aus

**$btn\_commit\_clicked**

1. Leert Listbox
2. Führt git commit -m aus mit Text der Textbox „Commit/Repo Description“ aus
3. Fügt Ausgabe von git commit der Listbox zu

**$btn\_push\_clicked**

Führt Git Push und Get-GitStat aus.

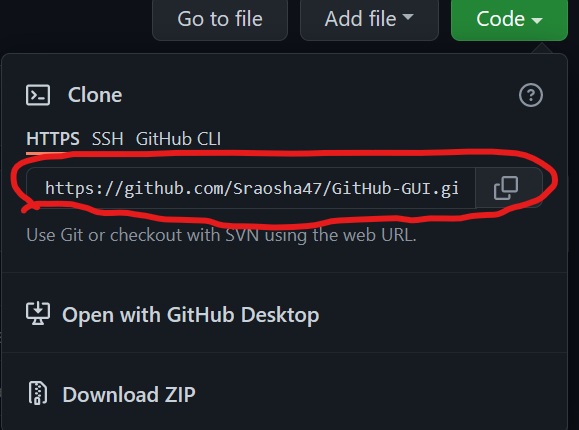
**$btn\_git\_initialise\_clicked**

# Betriebsdokumentation (Meilenstein C: individuelle Aufgabe 3)

Für Administrator und Benutzer wird folgende Anleitung ausgeliefert ...

## Installationsanleitung für Administratoren

1. Neuste Version von git herunterladen und installieren:  
   (<https://git-scm.com/downloads>)
2. Repository GitGui (<https://github.com/Sraosha47/GitHub-GUI/tree/afkhami>) forken.  
   Ein Bild, das Text, Gerät, Anzeige, Messanzeige enthält.

   Automatisch generierte Beschreibung
3. Zu eigenem Fork wechseln.
4. Auf Code klicken und Link kopieren.  
   
5. Im Terminal an gewünschten Ort wechseln (Set-Location [Pfad])
6. git clone [kopierter Link] und im neu erstellten Ordner den .git-Ordner löschen.

Optional, falls Sie Ihre geklonten Dateien über GitHub verwalten möchten:

1. Set-Location [Pfad des geklonten Repos]
2. git add \*
3. git commit -m «Set upstream to remote »
4. git remote add origin [kopierter Link]
5. git push --set-upstream origin main

## Bedienungsanleitung für Benutzer

Das Programm ist folgendermassen zu bedienen ...