# 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 |

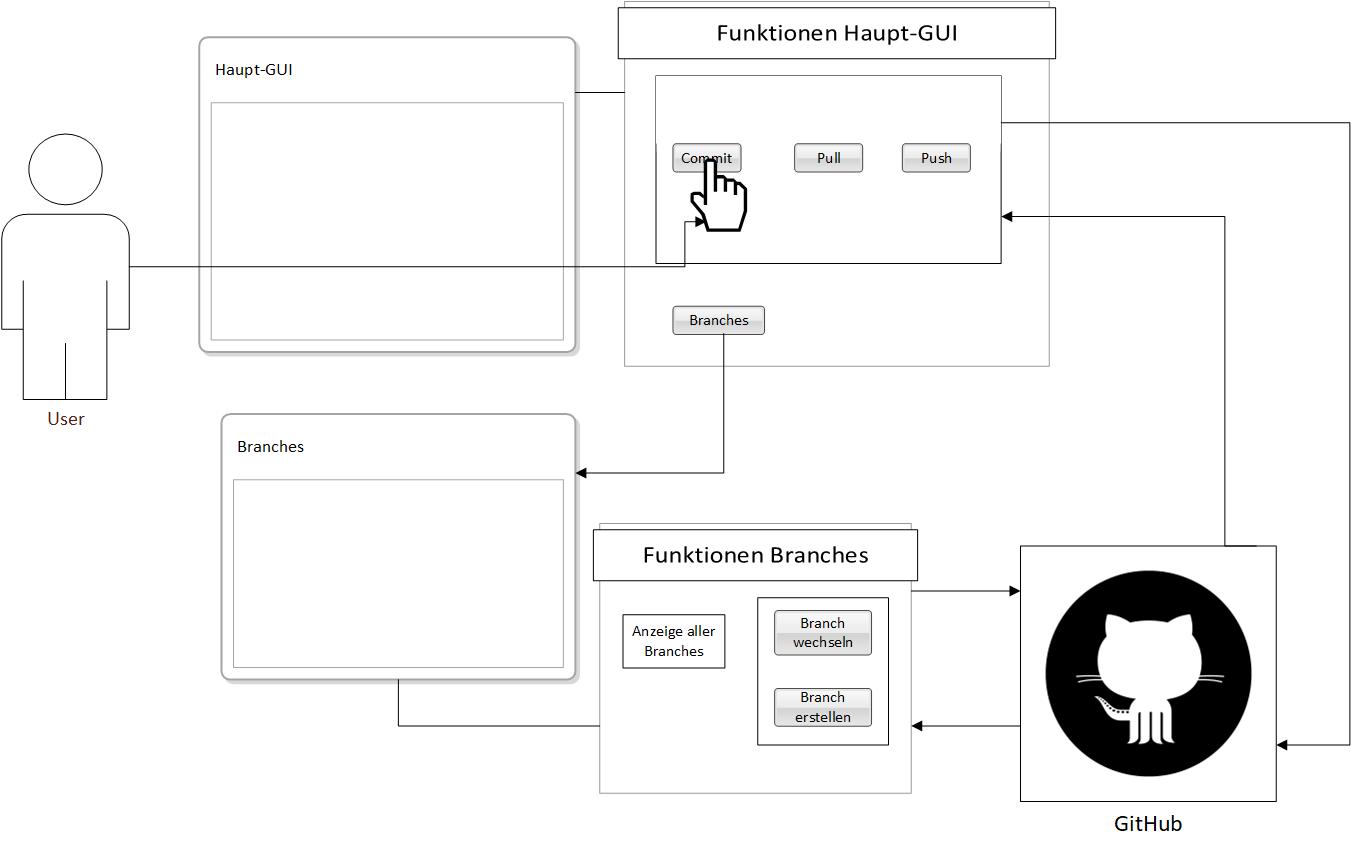
\*) Öffentliche GitHub-URLs im Ablageordner auf dem BSCW ablegen! (pro Team)

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

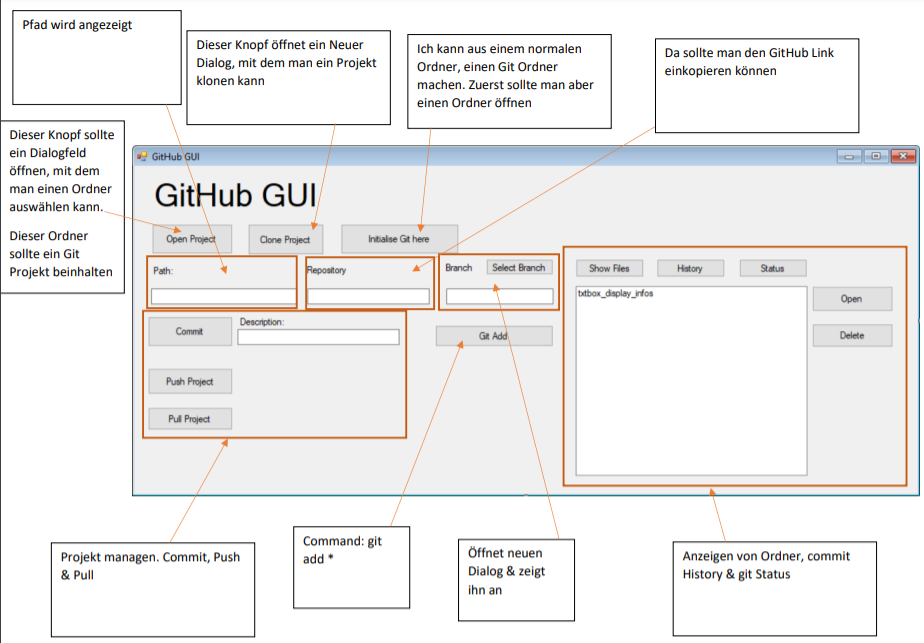
# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

## Schematische Darstellung der Funktionalität, sog. Funktionsmodell

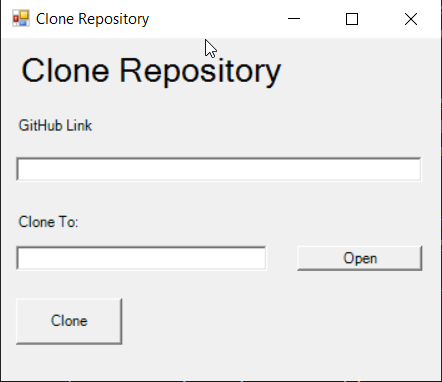


## Graphische Benutzer Schnittstelle (GUI) zur Konfiguration des Ablaufs



## Ablauf der Automation

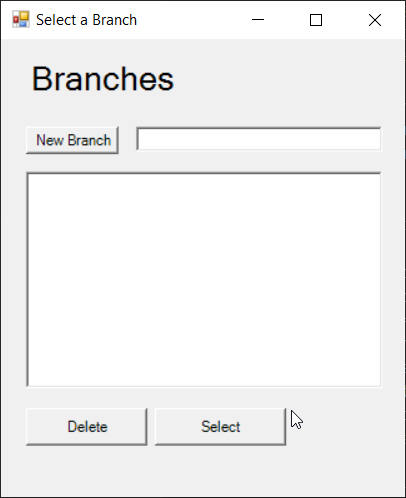
Storyboard



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

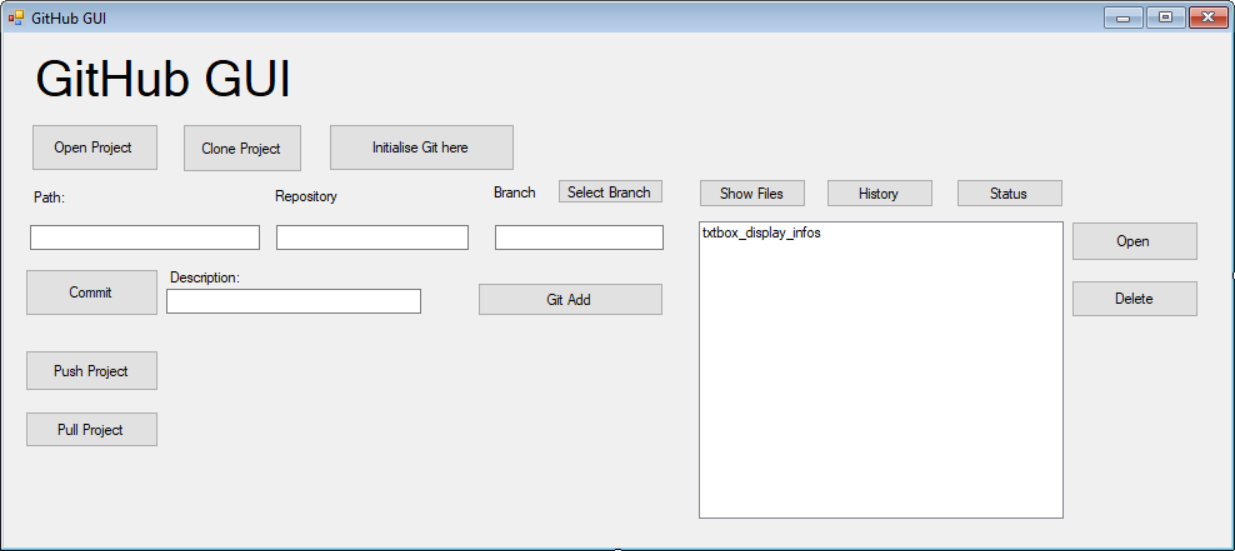
Jetzt kann man den Zielpfad auswählen.

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

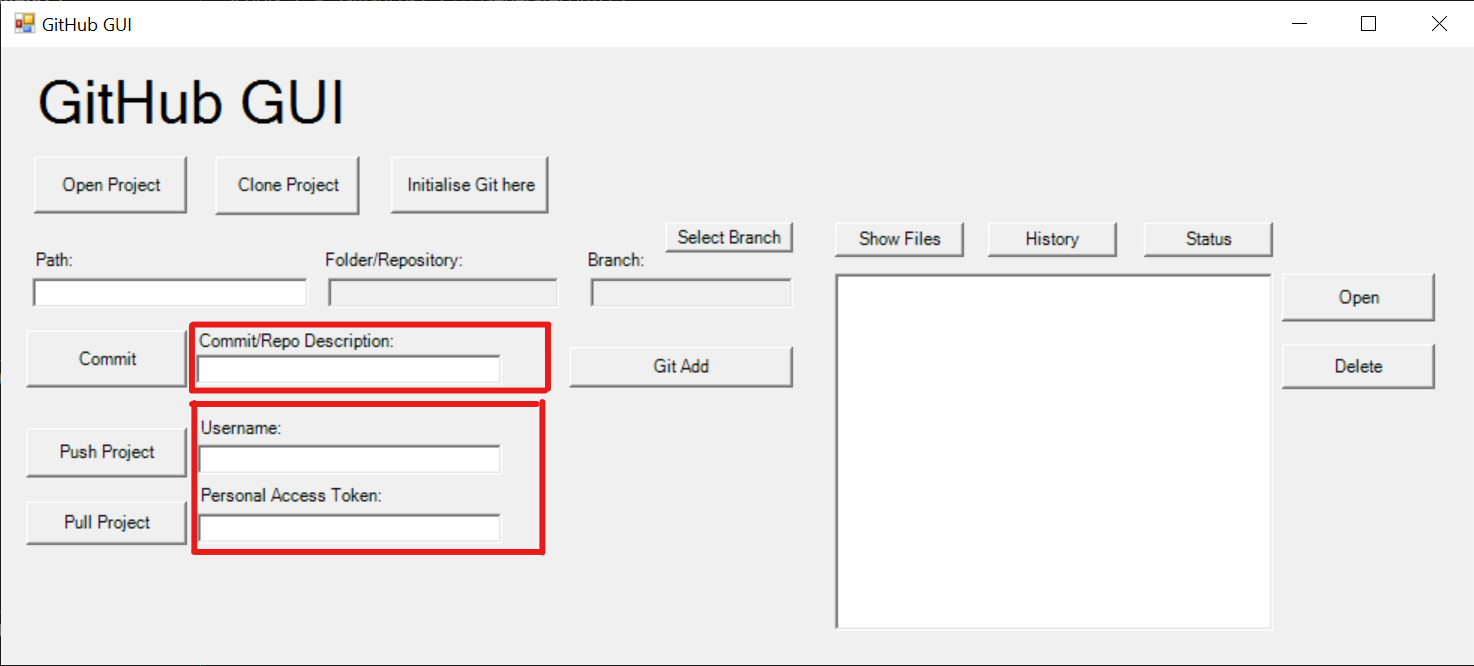
# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

# 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**

Erstellt als erstes ein GitHub repository mit dem Namen des ausgewählten Ordners als Repository-Namen.

Hierfür wird Authorisierungs-Header mit dem GitHub-Username und Access Token aus den dafür vorgesehenen Textboxen erstellt. Danach wird ein Body erstellt, welcher die verschiedenen Attribute des Repos definiert. Unter anderem wird ein ReadMe erstellt, dessen Inhalt in der Textbox Commit/Repo Description selber definiert werden kann. Header und Body werden dann an ein dafür vorgesehenes API von GitHub gesendet, welches dann das neue Repository erstellt.

Im zweiten Teil wird im ausgewählten Ordner ein git initalisiert. Dieses wird dann mit dem neu erstellten Repo auf GitHub mit git remote add, git pull –set-upstream und git push verbunden und synchronisiert. Der neue Branch «main» wird in der Textbox Branch angezeigt. Als letztes wird Show-GitFiles ausgeführt, welches jetzt zusätzlich zu den bisherigen Dateien, das neue ReadMe in der Listbox anzeigen sollte.

**$txtbox\_path\_KeyDown**

Erlaubt es dem Benutzer einen Pfad direkt in der dafür vorgesehenen Textbox einzufügen und mit Enter zu diesem Pfad zu wechseln. Zeigt darauf an wie der Ordner, bzw. das Repo heisst, wie der Branch heisst, bzw. dass es kein Repository ist und zeigt in der Listbox die Dateien des ausgewählten Ordners an.

### **GitGui\_Branches.ps1**

**Funktionen:**

**New-WinObject**

Gleich wie bei GitGui.ps1, nur auf das hiesige Fenster bezogen.

**Add-BranchList**

Löscht alle Objekte aus der Listbox. Führt git branch -a aus und gibt dessen Output an die Variable $branch\_list. Deren Inhalt wird dann in der Listbox angezeigt.

**Event Handlers:**

**$btn\_new\_branch\_clicked**

1. Erstellt und wechselt zu neuem Branch
2. Führt ersten Push mit --set-upstream aus um mit Remote Repo auf GitHub zu verbinden.
3. Zeigt den Namen des neuen Branch in der Textbox von GitGui.ps1 an
4. Löscht Inhalt der Textbox new\_branch\_name
5. Aktualisiert die Listbox Branches mit Add-BranchList
6. Aktualisiert die Listbox von GitGui.ps1 mit Show-GitFiles

**$btn\_delete\_branch\_clicked**

1. Fügt den Inhalt des ausgewählten Objekts der Listbox der Variable $branch zu
2. Löscht den Branch lokal mit git branch -d $branch.trim()
3. Löscht den Branch auf dem remote Repository mit git push origin -d $branch.trim()
4. Aktualisiert die Listbox mit Add-BranchList

**$btn\_switch\_branch\_clicked**

Die alternative Method mit der Textbox anzugeben zu welchem Branch man wechseln möchte, muss überarbeitet werden. Es muss überprüft werden ob der angegebene Branch überhaupt existiert. Alternative könnte man einstellen, dass der Branch, falls nicht existent, gleich erschaffen wird.

1. Trimmt den Text des ausgewählten Objekts und weist ihn der Viariable $branch\_select zu.
2. Splitted diesen Text bei allen Leerzeichen und weist das erste Objekt des erstellten Arrays der Variable zu. (Ermöglicht es zu bestimmten Commits zu wechseln)
3. Wenn $branch\_select nicht der jetzige Branch ist und ein Objekt in der Listbox ausgewählt wurde:
4. Wechselt zu ausgewähltem Branch mit git checkout $branch\_slct
5. Wenn kein Objekt in der Listbox ausgewählt ist und die Textbox für einen neuen Branch nicht gleich Null ist:
6. Wechselt zu Branch der in der Textbox new\_branch\_name angegeben ist.

**$btn\_version\_ctrl\_branch\_clicked**

Zeigt die bisherigen Commits mit ihren Hashs in der Listbox auf.

**$btn\_refresh\_branch\_clicked**

Zeigt alle Branches auf. Wird benötigt um von den Commits zu den Branches zurückzukehren.

### **Git\_Clone\_Repository.ps1**

**Funktionen:**

**New-WinObject**

Gleich wie bei GitGui.ps1, nur auf das hiesige Fenster bezogen.

**New-CloneOrder**

1. Prüft ob offener Pfad in GitGui bereits ein Repository
2. Falls nicht: Wechselt zu diesem Pfad
3. Fügt Pfad in $txtbox\_cloning\_destination ein

**New-Clone**

1. Erstellt Variable $repo\_url und weist ihr den Text der Textbox github\_link zu
2. Splitted die $repo\_url anhand von „/“ und weist das erstellte Array der Variablen $repo\_name zu.
3. Wählt das letzte Element des Arrays aus, entfernt den Anhang „.git“ und weist es $repo\_name zu.
4. Erstellt Variable $new\_path der der derzeitige Pfad + „\“ + $repo\_name zugewiesen wird.
5. Das GitHub Repo wird geklont [git clone $repo\_url]
6. Der Pfad wird mit [Set-Location $new\_path] ins neue Repository geändert.
7. Die Textboxen von GitGui werden aktualisiert
8. Ein upstream zum GitHub Repo wird aufgesetzt
9. Beide Textboxen des Git\_Clone\_Repository.ps1 werden geleert.

**Set-AllPaths**

1. Wechselt zu Pfad der in der Textbox cloning\_destination angegeben ist.
2. Aktualisiert den angezeigten Pfad im GitGui.
3. Aktualisiert die Listbox des GitGuis so dass aktuelle Dateien/Ordnerstruktur angezeigt wird.

**Get-LinkError**

Erstellt Fenster mit der Nachricht «Please insert a link to a GitHub repository» und lässt es aufpoppen.

**Get-DestError**

Erstellt Fenster mit der Nachricht «Selected directory already is a Repository. Please choose a different location.» und lässt es aufpoppen.

**Event Handler:**

**$btn\_clone\_clicked**

1. Prüft ob eingegebener Text in txtbox\_github\_link mit der Form «https://github.com/.+/.+» übereinstimmt.
2. Falls nicht: Get-LinkError
3. Sonst: falls $txtbox\_cloning\_destination nicht leer -> New-Clone

**$btn\_select\_path\_clicked**

1. Öffnet den FolderBrowser.
2. Falls der ausgewählte Pfad Dateien beinhaltet wird geprüft ob es sich um ein Repository handelt.
3. Falls ja: Get-DestError
4. Sonst:
   1. Der Pfad in der cloning\_destination Textbox wird angepasst
   2. Set-AllPath

**$txtbox\_github\_link\_KeyDown**

1. Prüft ob eingegebener Text in txtbox\_github\_link mit der Form «https://github.com/.+/.+» übereinstimmt.
2. Falls nicht: Get-LinkError
3. Sonst: falls $txtbox\_cloning\_destination nicht leer -> New-Clone

**$txtbox\_cloning\_destination\_KeyDown**

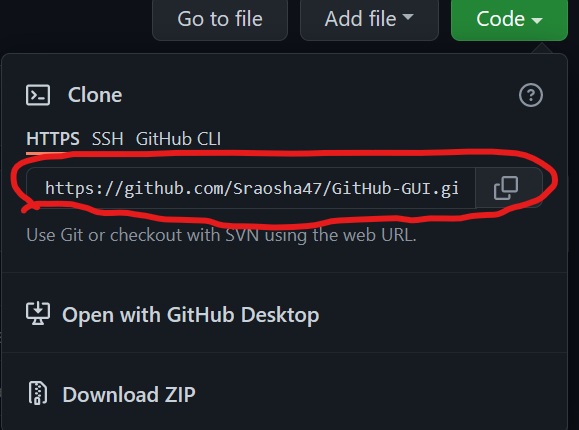
1. Prüft ob Textbox leer ist.
2. Wenn nicht leer und Enter gedrückt wird:
   1. Falls ausgewählter Ort bereits ein Repository -> Get-DestError und Textbox leeren
   2. Sonst: Set-AllPaths

# 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 ...