# Teilnehmer/innen des Teams:

|  |  |
| --- | --- |
| Klasse:  Ap17c | Team:  Diogo Caraça  Silvio Merz |

# Anforderungsdefinition (Meilenstein A)

|  |  |
| --- | --- |
| PC-Optimierer | |
| **Fachlicher Inhalt:**  (Allgemeine Beschreibung) | Nutzen: Mit dem Programm sollte die Performance des Computers gesteigert werden.**Automation:**  Das Programm sollte per Knopfdruck automatisch einige Scripts ausführen. Z.B:   * Speicher optimieren * Leistung optimieren   **Details:**   * Konfiguration: Auswählen was bereinigt werden muss * Integration: Startbare Applikation im Betriebssystem * Administration: Nicht nötig * Sicherheitsaspekte: Programm ist Passwort geschützt     **Erkenntnisse aus der Machbarkeitsabklärung in Windows Powershell:**  Wir haben überprüft ob man Ordner leeren kann, Tasks killen kann und ob man Applikationen vom Autostart aktivieren bzw. deaktivieren kann. |
| **MUSS**  **Kriterien:**  (Konkrete Features, die umzusetzen sind) | **Folgende Features sollen implementiert werden, um einen produktiven Ablauf sicherzustellen:**   * Die Scripts werden über ein GUI per Knopfdruck oder Formulare ausgeführt   + Papierkorb wird geleert   + Vorbestimmte Ordner leeren   + Temporäre Files werden gelöscht   + Autostart von Apps aktivieren/deaktivieren   + Tasks killen |

|  |  |
| --- | --- |
| **KANN**  **Kriterien:**  (Konkrete Features, die optional sind) | **Folgende Features können zusätzlich implementiert werden: (Varianten, Kreativität)**   * User kann zu leerende Ordner selber auswählen * Windows visuelle Effekte können optimiert werden |

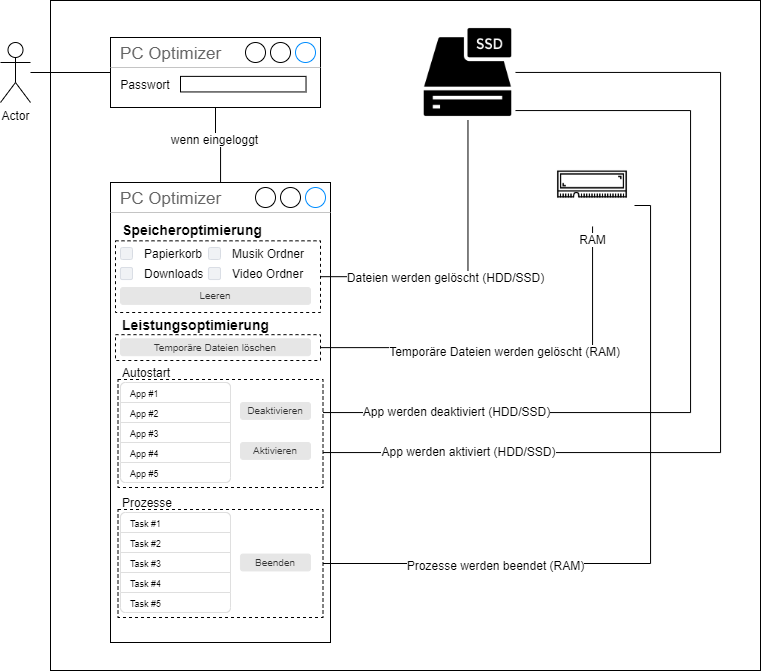
## Planung Meilensteine (LB1 / LB2)

|  |  |  |  |
| --- | --- | --- | --- |
| *MS* | *Tätigkeit / Abgabe* | *Soll-Datum* | *Ist-Datum* |
| A | ProjektstartTeam Bildung, Kollaborationsplattform, GitHub Repos \*, LehrerzugangWahl / Ausarbeitung der Anforderungsdefinition Abnahme Anforderungsdefinition durch Lehrperson | 13.03.2020 |  |
| B | Teamaufgabe 1:Abgabe: Lösungsdesign  (Funktionsmodell / GUI / PAP / Storyboard) | 20.03.2020 |  |
| B2 | Teamaufgabe 2: (Nur LB2)Abgabe: Testvorschrift und Testfälle | 27.03.2020 |  |
| C | Einzelaufgabe 2 (LB1) / 3 (LB2):Abgabe Programmcode und DokumentationFachgespräch Projektabnahme | 03.04.2020 |  |
| C2 | Einzelaufgabe 4: (Nur LB2)Abgabe: Ausgefüllter Systemtest | 03.04.2020 |  |

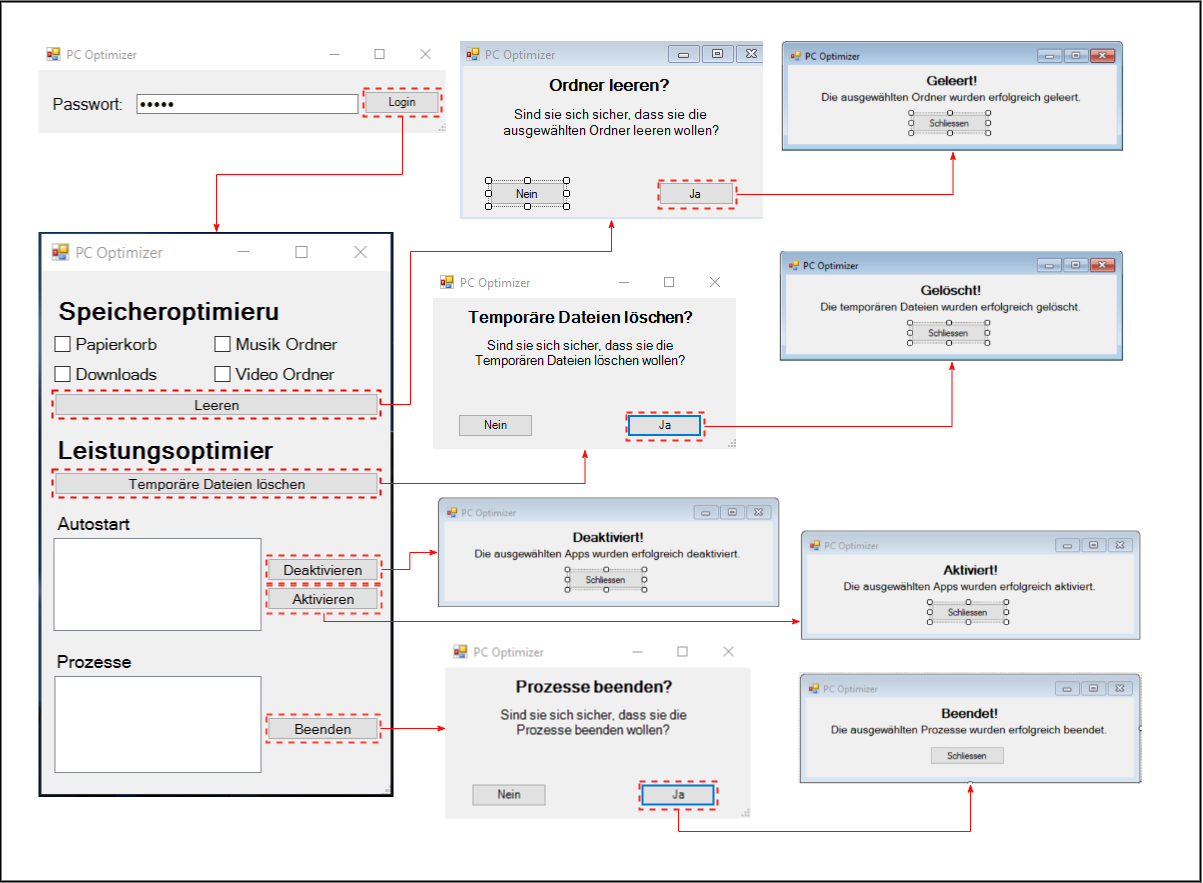
# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

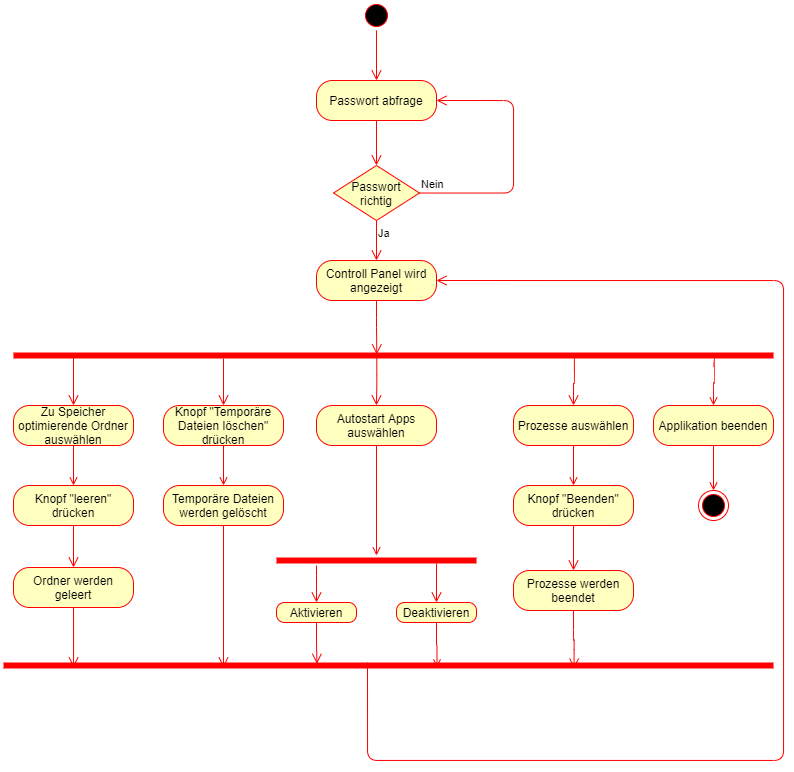
## Schematische Darstellung der Funktionalität, sog. Funktionsmodell



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



## Ablauf der Automation



# Testvorschrift (LB2 Meilenstein B2: Teamaufgabe 2)

Testbeschrieb und vorbereitetes Testprotokoll siehe Dokument   
***M122\_LB2\_Testvorschrift\_MS-B2\_Caraça\_Merz.docx***

# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

Ausgefülltes Testprotokoll siehe Dokument   
***Von Diogo Caraça: M122\_LB2\_Testvorschrift\_MS-C2\_Caraça.docx***

# Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

Die erstellten Projekt-WPS-Scripts sind hier abgelegt und für Entwickler dokumentiert:

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

Namenskonvention URL: <https://github.com/SilvioMerz/M122_Ap17c_Diogo_Silvio/tree/silvio>

## Umfang / Abgrenzung / Änderungen gegenüber Design

Aufgrund unten beschriebener Umstände sind Anpassungen des ursprünglichen Lösungsdesigns gemacht worden:

* „Aktualisieren“ Button wurde beim Abteil „Prozesse“ hinzugefügt
  + Wurde benötigt um die Prozesse auf den neusten Stand aktualisieren zu können
* Neue Messagebox „Nichts ausgewählt“ hinzugefügt
  + Hinweis für den User, dass er keine Auswahl getroffen hat
* Maximier Button entfern
  + Das Fenster sollte in seiner Grösse gleich bleiben
* Messageboxes Designänderung
  + Das Design wurde auf eine Klassische Messagebox angepasst

## Funktionalität der Implementation.

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

„Delete Files“ Funktion welche von „Leeren“ und „Temporäre Dateien löschen“ aufgerufen wird um die Ordner zu leeren:

function DeleteFiles($toEmpty)

{

#Get objects from folders

Get-ChildItem -Path $toEmpty -Recurse -Force |

ForEach {

try {

#Delete object

$\_ | rm -Recurse -Force -ErrorAction Stop

} catch {

#Warning if object is used

Write-Warning "Could not remove Item $($\_.TargetObject)"

}

}

}

„Change Autostart“ Funktion welche von „Deaktivieren“ und „Aktivieren“ verwendet wird:

function changeAutostart($byteCode)

{

#Check- if no apps were selected

if ($listAutostart.SelectedIndices.Count -eq 0) {

& $messageBoxNoSelection

} else {

#Get selected app name

$appname = $listAutostart.SelectedItems.Text

#Change autostart value of selected app

Set-ItemProperty -Path HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run -Name $appname -Value ($byteCode)

[System.Windows.MessageBox]::Show('Die ausgewählte App wurde erfolgreich deaktiviert','PC Optimizer')

}

}

“Messagebox Selection” Funktion welche von „Leeren“, „Temporäre Dateien löschen“ verwendet wird, damit ich nicht jedes Mal die Optionen der Messagebox neu angeben muss:

function MessageBoxSelection($text)

{

#Message box with defined title and options and only variable text

[System.Windows.MessageBox]::Show($text,'PC Optimizer','YesNo','Question')

}

„Leeren“ onClick Event:

##################################################

#"Leeren" button is pressed:

##################################################

$emptyDirectives=

{

#Check if no checkbox was selected

if (!$music.Checked -and !$video.Checked -and !$downloads.Checked -and !$paperBin.Checked) {

& $messageBoxNoSelection

}

else {

#If at least one box was selected action confirmation required

$msgBoxSelection = MessageBoxSelection("Ordner leeren?")

switch ($msgBoxSelection) {

#User pressed Yes:

'Yes' {

$foldersToEmpty = @()

#Takes the path of the selected folders

if ($music.Checked) {

$foldersToEmpty += "$env:USERPROFILE\Music"

}

if ($video.Checked) {

$foldersToEmpty += "$env:USERPROFILE\Videos"

}

if ($downloads.Checked) {

$foldersToEmpty += "$env:USERPROFILE\Downloads"

}

if ($paperBin.Checked) {

$foldersToEmpty += "C:\`$Recycle.Bin\" + ([System.Security.Principal.WindowsIdentity]::GetCurrent()).User.Value

}

DeleteFiles($foldersToEmpty)

[System.Windows.MessageBox]::Show('Die ausgewählten Ordner wurden erfolgreich geleert','PC Optimizer')

}

#User pressed No:

#Nothing happens

}

}

}

##################################################

„Temporäre Dateien löschen“ onClick Event:

##################################################

#"Temporäre Dateien löschen" button is pressed:

##################################################

$deleteTempFiles=

{

#Ask for confirmation messagebox

$msgBoxSelection = MessageBoxSelection("Temporäre Dateien löschen?")

switch ($msgBoxSelection) {

#User pressed Yes:

'Yes' {

#Temp files paths

$tempfolders = @(“C:\Windows\Temp\\*”, “C:\Windows\Prefetch\\*”, “C:\Users\\*\Appdata\Local\Temp\\*”)

DeleteFiles($tempfolders)

[System.Windows.MessageBox]::Show('Die temporären Dateien wurden erfolgreich gelöscht','PC Optimizer')

}

#User pressed No:

#Nothing happens

}

}

##################################################

„Deaktivieren“ onClick Event:

##################################################

#"Deaktivieren" button is pressed:

##################################################

$deactivateApp=

{

#Bytecode to deactivate autostart

changeAutostart([byte[]](0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00))

}

##################################################

„Aktivieren“ onClick Event:

##################################################

#"Aktivieren" button is pressed:

##################################################

$activateApp=

{

#Bytecode to activate autostart

changeAutostart([byte[]](0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00))

}

##################################################

„Beenden“ onClick Event:

##################################################

#"Beenden" button is pressed::

##################################################

$quitProcess=

{

#Check- if no processes were selected

if ($listProcesses.SelectedIndices.Count -eq 0) {

& $messageBoxNoSelection

}

else {

#Get selected processes

$selectedProcesses = @($listProcesses.SelectedIndices)

#Get ids of selected processes

$columnIndex = ($listProcesses.Columns | Where {$\_.Text -eq "Id"}).Index

#Stop selected tasks

$selectedProcesses | ForEach-Object {

#Stop task by Id

Stop-Process -Id ($listProcesses.Items[$\_].SubItems[$columnIndex]).Text -Confirm:$false -Force

}

#Update listview

Start-Sleep -s 0.55

&$updateProcesses

[System.Windows.MessageBox]::Show('Die ausgewählten Prozesse wurden erfolgreich beendet','PC Optimizer')

}

}

##################################################

„Aktualisieren“ onClick Event:

##################################################

#"Aktualisieren" button is pressed:

##################################################

$updateProcesses=

{

#Clear listview

$listProcesses.Items.Clear()

$listProcesses.Columns.Clear()

#Get processes

$processes = Get-Process | Select Id,ProcessName | Sort-Object ProcessName

$processProperties = $processes[0].psObject.Properties

#Create Listview header

$processProperties | ForEach-Object {

$listProcesses.Columns.Add("$($\_.Name)")

}

#Show processes

ForEach ($process in $processes){

$processListViewItem = New-Object System.Windows.Forms.ListViewItem($process.Id)

$process.psObject.Properties | Where {$\_.Name -ne "Id"} | ForEach-Object {

$columnName = $\_.Name

$processListViewItem.SubItems.Add("$($process.$columnName)")

}

$listProcesses.Items.Add($processListViewItem)

}

$listProcesses.Columns[0].Width = 40

$listProcesses.Columns[1].Width = 175

}

„Login“ onClick Event:

$loginBtn.add\_Click({

if ($passwortInput.Text -ceq "PC-Optimizer\_Admin") {

#Close login form

$loginF.Hide()

$loginF.Close() | Out-Null

#Show PC Optimizer

GenerateForm

}

else {

[System.Windows.MessageBox]::Show('Passwort ist falsch!','PC Optimizer','OK','Warning')

}

})

# 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 ...

* Ein „Windows PowerShell“ Fenster öffnen
* Mit dem Befehl „Set-Executionpolicy unrestricted” die Executenpolicy veringern um Custom Script ausführen zu können
* Befehl mit „Ja, alle“ bestätigen
* Das Script PCOptimizer.ps1 herunterladen
* Das Script als Admin ausführen

## Bedienungsanleitung für Benutzer

Das Programm ist folgendermassen zu bedienen ...

* Das Passwort „PC-Optimizer\_Admin“ eingeben
  + Speicheroptimierung
    - Zu leerende Ordner auswählen
    - „Leeren“ Button betätigen
    - Mit „Ja“ die Aktion bestätigen
  + Temporäre Dateien löschen
    - „Temporäre Dateien löschen“ Button betätigen
    - Mit „Ja“ die Aktion bestätigen
  + Autostart
    - Deaktivieren
      * Zu deaktivierende App auswählen
      * „Deaktivieren“ Button betätigen
    - Aktivieren
      * Zu aktivierende App auswählen
      * „Aktivieren“ Button betätigen
  + Prozesse
    - Zuerst die aktuelle Liste laden durch betätigen des „Aktualisieren“ Buttons
    - Zu beendende Prozesse selektieren
    - „Beenden“ Button betätigen
    - Mit „Ja“ die Aktion bestätigen