Berufsakademie Sachsen

Staatliche Studienakademie Leipzig

**Pflichtenheft QuixSync**

Projektarbeit

für Softwareprojekt im fünften Semester

in der Studienrichtung Informatik

Eingereicht von: Quentin Weber, 5000924

Philipp Ludwig, 5000964

Chris Sembritzki, 5000993

Seminargruppe: CS16-1

Inhaltsverzeichnis

[I Abkürzungsverzeichnis 4](#_Toc527541609)

[1 Anforderungen 5](#_Toc527541610)

[1.1 Pflichtanforderungen 5](#_Toc527541611)

[1.2 Wunschanforderungen 5](#_Toc527541612)

[1.3 künftige Anforderungen 5](#_Toc527541613)

[2 Produkteinsatz 6](#_Toc527541614)

[2.1 Anwendungsbereiche 6](#_Toc527541615)

[2.2 Zielgruppen 6](#_Toc527541616)

[2.3 Betriebsbedingung 6](#_Toc527541617)

[3 Produktübersicht 7](#_Toc527541618)

[3.1 Kurze Produktübersicht 7](#_Toc527541619)

[3.2 UML-Diagramme 7](#_Toc527541620)

[3.2.1 Ablauf einer kompletten Synchronisation 7](#_Toc527541621)

[3.2.2 Indizierung 8](#_Toc527541622)

[3.2.3 Vergleich 9](#_Toc527541623)

[3.2.4 Synchronisierung 10](#_Toc527541624)

[3.3 ER Diagramme 11](#_Toc527541625)

[4 Produktfunktionen 12](#_Toc527541626)

[4.1 Analysieren 12](#_Toc527541627)

[4.2 Vergleichen 12](#_Toc527541628)

[4.3 Synchronisierung 12](#_Toc527541629)

[4.4 FTP-Verbindungen 12](#_Toc527541630)

[(4.5 Daemon Betrieb) 12](#_Toc527541631)

[5 Benutzeroberfläche 13](#_Toc527541632)

# 1 Auftrag

## 1.1 Beteiligte Personen

Auftragnehmer:

Projektleiter:

Quentin Weber

- Quentin.Weber@cs16-1.ba-leipzig.de

Projektteam:

Philipp Ludwig

- Philipp.Ludwig@cs16-1.ba-leipzig.de

Chris Sembritzki

- Chris-Thomas.Sembritzki@cs16-1.ba-leipzig.de

Auftraggeber:

Christian Heller

- Christian.Heller@ba-leipzig.de

## 1.2 Auftragsbeschreibung

Das Ziel des Projektes ist das Erstellen eines Tools zur Synchronisation von Verzeichnissen und Dateien. Dies soll lokal auf einem Rechner oder über das Netzwerk erfolgen. Dem Nutzer wird für diesen Zweck eine grafische Oberfläche zur Verfügung gestellt.

## 1.3 Meilensteine

Lastenheft/Thema: Freitag, den 05.10.2018 um 12:00 Uhr

Pflichtenheft: Freitag, den 19.10.2018 um 12:00 Uhr

Softwareprototyp: Freitag, den 07.12.2018 um 12:00 Uhr

Präsentation: Freitag, den 14.12.2018 um 08:00 Uhr

## 1.4 Dokumentation

Die Dokumentation und Versionierung erfolgt über GitHub (<https://github.com/BuckUbel/QuixSync>).

# 2 Anforderungen

Die Anforderungen werden in Form von User Stories dargestellt. Diese beschreiben einen spezifischen Anwendungsfall, welcher von der Anwendung ausgelöst werden soll.

## 2.1 Pflichtanforderungen

Als Anwender möchte ich einen konkreten Ordner mit einem anderen vergleichen. Als Ergebnis dieses Vergleiches sollen mir die Unterschiede aufgelistet werden.

Als Anwender möchte ich einen bestimmten Ordner an verschiedenen Stellen innerhalb des Dateisystems ablegen. Diese verschiedenen Ordner sollen den selben Informationsgehalt und Struktur besitzen.

Als Anwender möchte ich eine graphische Benutzeroberfläche für die Konfiguration und Bedienung des Programmes verwenden..

## 2.2 Wunschanforderungen

Als Anwender möchte ich nicht nur lokale, also auf dem aktuellen Rechner erreichbare, Ordner synchronisieren. Ein Abgleich zweier Ordner über FTP wäre wünschenswert.

Als Anwender brauche ich einen automatisierten Abgleich der zu synchronisierenden Ordner. Das Zeitintervall dafür kann selbst gewählt werden. Dies sorgt dafür, dass ein manuell angestoßener Abgleich nicht mehr zwingend durchgeführt werden muss, um die beiden Ordner auf dem aktuellen Stand zu halten.

## 2.3 künftige Anforderungen

Die nachfolgenden User Stories beschreiben Anforderungen, die in der Zukunft der Anwendung hinzugefügt werden könnten.

Als Anwender möchte ich eine auf UNIX-Systemen optimierte Dateisynchronisation durchführen.

## 2.4 Machbarkeit

Der angegebene Auftrag ist mit der Programmiersprache Java umsetzbar. Der Betrieb der zu entwickelnden Software ist hardwareseitig nicht sehr Ressourcenbeanspruchend und sollte damit auf jeder Konfiguration betrieben werden können.

Als reine Entwicklungszeit bis zum Einreichen des ersten Prototyps beim Auftraggeber werden xx Stunden veranschlagt.

# 3 Produkteinsatz

## 3.1 Anwendungsbereiche

Dieses Programm hat die Funktion zwei Verzeichnisse zu analysieren, zu vergleichen und schlussendlich zu synchronisieren. Die Synchronisation eines Ordners kann auch als Backup für eben jenes Verzeichnis dienen. Dies kann über im Dateisystem eingebundene Ordner erfolgen.

## 3.2 Zielgruppen

Dieses Programm ist besonders für Personen interessant, die häufig auf unterschiedlichen Rechnern arbeiten müssen und den aktuellen Stand eines Ordners auf allen Geräten zur Verfügung gestellt haben wollen.

## 3.3 Betriebsbedingung

Der Benutzer muss ein Desktop-Betriebssystem mit der aktuellen Version von Java 8 verwenden. Allerdings wird im Rahmen dieses Projektes die Synchronisierung ausschließlich auf Windows 10 getestet.

Soll eine FTP-Verbindung erstellt werden, muss eine Netzwerkverbindung zu dem entsprechenden FTP-Server realisiert werden können.

# 4 Produktübersicht

## 4.1 Kurze Produktübersicht

QuixSync wird über eine grafische Oberfläche bedient. Diese soll dem Nutzer folgende Funktionen zur Verfügung stellen:

- Auswählen zweier Verzeichnisse (Quellverzeichnis, Zielverzeichnis)

- Erstellen einer Indexdatei für ein Verzeichnis

- Vereinfachtes Präsentieren einer Indexdatei

- Erstellen einer Vergleichsdatei zwischen zwei Indexdateien

- Vereinfachtes Präsentieren einer Vergleichsdatei

- Starten einer weichen Synchronisation aufgrund einer Vergleichsdatei

- Starten einer harten Synchronisation aufgrund einer Vergleichsdatei

- Starten einer kompletten Synchronisation

(Erstellen und Vergleich von Indexdateien mit inbegriffen)

- Setzen von Nutzer-spezifischen Einstellungen

## 4.2 UML-Diagramme

### 4.2.1 Ablauf einer kompletten Synchronisation

3 Schritte:

Indizierung 🡪 Vergleichen 🡪 Synchronisieren

(Bedenken, das Diagramm muss im Hochformat sein)

### 4.2.2 Indizierung

Parameter: Verzeichnis

🡪 Inhalt wird aufgelistet

🡪 zwei Arrays werden erstellt (Ordner, Dateien)

🡪 Informationen zu der Datei werden gespeichert:

(siehe 3.3.1)

🡪 Informationen zum Ordner werden gesammelt:

(siehe 3.3.1)

🡪 rekursive Inhaltauflistung der Ordner

🡪 Speicherung der Arrays mit weiteren Informationen in einer Datei

🡪 Anzahl der Dateien

🡪

### 4.2.3 Vergleich

Parameter: zwei Indizierungsdateien, isHardSync

🡪 Schleife der Elemente (Ordner/Dateien) der ersten Ebene werden von dem Quellindex aufgerufen

🡪 Suche der Datei mit dem Erstelldatum in dem Zielindex

🡪 gefunden?

ja 🡪 Flag „isCompared“ wird auf true gesetzt

🡪 ist der Name, die Dateigröße oder das „zuletzt geändert“ verschieden?

ja 🡪 Element wird im ReturnCopyArray gespeichert

nein 🡪 Datei wird ignoriert

nein 🡪 Element wird ReturnCopyArray gespeichert

🡪 Ordner aus dem ReturnCopyArray, werden auf dieselbe Weise rekursiv verglichen

🡪 isHardSync?

ja 🡪 Schleife der Elemente der ersten Ebene werden von dem Zielindex

aufgerufen

🡪 „isCompared“ == true?

ja 🡪 nichts geschieht

nein 🡪 Element wird im ReturnDeleteArray gespeichert

nein 🡪 nichts geschieht

🡪 ReturnCopyArray und ReturnDeleteArray werden in einer Datei gespeichert

### 4.2.4 Synchronisierung

Parameter: Vergleichsdatei

🡪 Array der fehlenden Daten(ReturnCopyArray), wird nach und nach

durchgegangen

🡪 Daten werden kopiert

🡪 Array der zu löschenden Daten (ReturnDeleteArray), wird nach und nach durchgegangen

🡪 Daten werden gelöscht

## 4.3 ER Diagramme

4.3.1 Speicherelement

Datei enthält Eigenschaften:

- Erstelldatum (Timestamp)

- zuletzt geändert (Timestamp)

- absoluter Pfad (String)

- Speichergröße (File Size (siehe 3.3.2 ))

Spezialisierungen:

- Datei

- Ordner

Abhängigkeiten:

Datei hat einen übergeordneten Ordner (String)

4.3.2 File Size

Eigenschaften:

- Größe (float)

- Vorzeichen (char)

# 5 Produktfunktionen

## 5.1 Analysieren

## 5.2 Vergleichen

## 5.3 Synchronisierung

## 5.4 FTP-Verbindungen

## (5.5 Daemon Betrieb)

# 6 Benutzeroberfläche