**Projekt PE-Parser**

Datum Begin: 27.03.2023

Datum Ende : 07.04.2023

**Index**

1. Deckblatt & Index
2. Einleitung
   1. Projektumfeld
   2. Projektziel
   3. Projektbegründung
   4. Projektschnittstellen
   5. Projektabgrenzung
3. Projektplanung
   1. Projektphasen
   2. Abweichung vom Projektantrag
   3. Ressourcenplanung
   4. Entwicklungsprozess
4. Analysephase
   1. Ist-Anal
   2. Wirtschaftlichkeitsanalyse
   3. Make or Buy-Entscheidung
   4. Projektkosten

**Einleitung**

Projektumfeld:

* Betrieb: Berufsschule
* Mitarbeiterzahl: 2
* Auftraggeber: Hr. Rath (Cracker)

Projektziel:

* Thema: Ist die Erstellung eines sowohl schlanken als auch leistungsfähigem und dennoch flexiblem Parsers für „Portable Executable“ welcher als Produkt auf den Markt zu bringen ist.
* Was soll erreicht werden?

Projektbegründung

* Warum ist das Projekt sinnvoll (z.B. Kosten- oder Zeitersparnis, weniger Fehler)?
* Was ist die Motivation hinter dem Projekt?

Projektschnittstellen

* Mit welchen anderen Systemen interagiert die Anwendung (technische Schnittstellen)?
* Wer genehmigt das Projekt bzw. stellt Mittel zur Verfügung?
* Wer sind die Benutzer der Anwendung?
* Wem muss das Ergebnis präsentiert werden?

Projektabgrenzung

* Was ist explizit nicht Teil des Projekts (insb. bei Teilprojekten)?

INDEX:

1. Name
2. Projektziel

1 Name: Pe-Parser

2 Projektziel:

Am Ende dieses Projektes soll eine voll funktionsfähige Library\* entstehen, welche in der Lage ist ausführbare Dateien im Portable Executable Format zu validieren und zu masern.

Der Pe-Parser soll als dynamische/statische Library (.dll/.obj) und als c++ Source code (.h/.cpp) verwendbar sein.

Python Funktionsverwaltung mit GUI.

<3

3 Projektbegründung:

Der Pe-Parser soll bei „source only“ verwendung möglichst schlanken Assembly erzeugen, mit Hilfe von compiletime Metaprogramming\*\*. Der Parser soll homogen, strukturiert, erweiterbar, sein. Dies soll mit hilfe von Compiletime-wrappern\*\*, welche bereits bestehende (native) Windows API (sets) wrappen, bewerkstelligt werden.

Es soll ein Python zur darstellung und Fehlerbehandlung verwendet werden.

\*\* Compiletime-wrapper -> sind einfache/ komplexe “Hüllen“-Konstrukte, welche während der Kompilier-zeit aufgelöst werden (Meta-programming), und im späteren Assembly code nicht mehr existieren.

4 Technische Schnitstellen:

Es sollen native Windows API Sets verwendet werden.

5 Ziel Nutzer-gruppe:

Zu den Ziel-gruppen gehören Anwendungs-Entwickler.

6 Projektabgrenzung:

Dieses Projekt bietet nur die Möglichkeit zur Analyse aber nicht zum crypten.