Traffic Police

# 1 Einleitung

* 1. Projektumfeld

Ich arbeite bei der Netzwerkfirma Witcher-Netz. Wir sind ein kleiner Internet-Anbieter und streben für ständigen Wachstum. Die Anwender haben sich jedoch in der letzten Zeit über lange Warteizeiten beim Zugriff auf Web-Seiten beschwert. Die Netzwerkadministratoren haben festgestellt, dass das Problem bei der Namensauflösung auftritt.

* 1. Projektziel

Das Ziel des Projektes ist, eine bequeme Möglichkeit bei der DNS-Analyse durch eine einfach bedienbare Anwendung zu bieten. Die Anwendererfahrung ist die wichtigste Rückmeldung und das Problem der langen Wartezeiten muss dringend behoben werden. Meine Anwendung ermöglicht den Administratoren, die Probleme zu erkennen, analysieren und sie dadurch schneller lösen.

* 1. Projektbegründung

Dieses Projekt ist für unser guten Rumpf außerordentlich wichtig und wenn Kunden aufgrund der jetzigen Leistungsstörungen den Vertrag kündigen, verliert das Unternehmen sowohl Geld als auch potenzielle, neue Kunden.

* 1. Projektschnittstellen

Die Anwendung überprüft DNS-Anfragen und Antworten, die bei einer Netzwerkschnittstelle geschehen. Die Administratoren können dadurch analysieren, welche DNS-Anfragen auf einem System gestellt werden und welche Antworten darauf zurückgesendet werden. Zugunsten der Ausführbarkeit der Anwendung muss Winpcap auf dem System installiert werden, da es die Abhängigkeit des PcapDotNet Pakets ist.

# Projektplanung

* 1. Projektphasen

Für die Umsetzung standen 70 Stunden zur Verfügung, die in verschiedenen Teilen unterteilt werden können. Die einzelnen Phasen und deren benötigten Zeit lassen sich in der Tabelle darstellen.

|  |  |
| --- | --- |
| **Projektphase** | **Geplante Zeit** |
| Analysephase | 7 Stunden |
| Entwurfsphase | 10 Stunden |
| Implementierungsphase | 40 Stunden |
| Abnahme und Einführung | 1 Stunde |
| Erstellen der Dokumentation | 10 Stunden |
| **Gesamt** | **70 Stunden** |

Tabelle 1: Zeitplanung

* 1. Ressourcenplanung

Die Projektkosten wurden möglichst gering gehalten. Für die Entwicklung der Anwendung wurde Visual Studio 2019 Community Edition verwendet, welche eine performante, kostenlose Anwendung ist.

* 1. Entwicklungsprozess, Planungsmethode

Bevor der Entwicklungsprozess angefangen werden konnte, habe ich mich für das Wasserfall Vorgehensmodell entschieden und dementsprechend die zu implementierende Anwendung geplant und die Voraussetzungen analysiert. Da ich im Voraus wusste, was genau erreicht werden musste, haben sich die Stärke vom Wasserfallmodell gezeigt.

# Analysephase

* 1. Ist-Analyse

Der Internet-Anbieter Witcher-Netz muss sich mit einem Leistungsproblem auseinandersetzen: Kunden haben sich über die langen Ladezeiten und unerreichbare Webseiten beschwert. Die Netzwerkadministratoren streben sich, um die Wurzel der Probleme zu finden und die Situation zu analysieren. Ich möchte ihre Arbeit erleichtern, indem ich eine Anwendung zu Verfügung stelle, welche die DNS-Anfragen- und Antworten analysiert, auf IPv4 beschränkt. Damit wird die Entdeckung der Probleme leichter und schneller.

* 1. Wirtschaftlichkeitsanalyse

Aufgrund der Probleme der jetzigen Situation verliert das Unternehmen Kunden und das negative Phänomen mindert den Rumpf, welche das Unternehmen daran hindert, neue Kunden zu gewinnen. Dies bedeutet einen Verlust von mehrere hunderte von Euros monatlich.

* 1. Make or Buy-Entscheidung

Da die Anwendung sehr spezifisch ist, lohnt es sich vielmehr, eine eigene Implementierung durchzuführen, als ein fertiges Produkt zu kaufen. Somit spart das Unternehmen weitere kosten.

* 1. Projektkosten

Die Projektkosten sind aus der Arbeit des Praktikanten (ich), aus den Aufwendungen der Ressourcen (Strom) und aus der Beratung eines Entwicklers entstanden. Der Stundensatz eines Auszubildenden beträgt in der Regel 10€, der eines Mitarbeiters 25€.

Die gesamten Personal- und Projektkosten werden in der zweiten Tabelle dargestellt.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vorgang** | **Ressource** | **Zeit** | **Gesamt** |
| Entwicklung | 1 x Auszubildender | 70 Std. | 700,00€ |
| Fachgespräch | 1 x Mitarbeiter | 3 Std. | 75,00€ |
| Code-Analyse | 1 x Mitarbeiter | 2 Std. | 50,00€ |
| Abnahme | 1 x Mitarbeiter | 1 Std. | 25,00€ |
| Ressourcen | Strom | 76 Std. | 7,00€ |
| **Gesamtkosten: 857 €** | | | |

# Entwurfsphase

* 1. Zielplattform

Ich habe für die Fertigstellung der Anwendung die Programmiersprache C# gewählt. Die Zeitoptimierung entspricht der Auswahl der Benutzeroberfläche: die Anwendung wird durch die Konsole bedient. Windows 10 wird als Betriebssystem verwendet und es wird keine separate Datenbank benötigt, da die Ergebnisse werden in Form von einer Text-Datei gespeichert.

* 1. Architekturdesign

Es wird ein NuGet-Paket für die Erleichterung der Entwicklung verwendet: PcapDotNet. Das Paket ermöglicht die Protokollierung und Überprüfung des Netzwerkverkehrs, aber es hat die Abhängigkeit von einem äußeren Paket: Winpcap. Es werden keine weiteren Frameworks oder Architekturen benötigt.

* 1. Entwurf der Benutzeroberfläche

Die Benutzeroberfläche, wie oben erwähnt, ist die Konsole. Der Benutzer bedient die Anwendung durch Konsoleneingaben nach klarer Einleitung und Instruktionen.

Die Meldungen haben passende Farben: Fehler – rot, Warnung – gelb, Erfolg – grün und generelles Info – weiß.

Die einzelnen Netzwerkereignisse werden in Echtzeit in der Konsole ausgegeben.

* 1. Datenmodell

Die protokollierten Entitäten sind Netzwerkevents. Die Netzwerkevent Klasse ist eine abstrakte Klasse, woraus zwei abgeleitete Klassen stammen: Query und Response. Dadurch wird unterschieden, ob die gerade aufgetretene Netzwerkevent eine Anfrage oder eine Antwort ist.

Die Netzwerkevents werden einer Event-Repository Klasse zugefügt und in einer Liste von Netzwerkevents gespeichert.

Die Überwachung selbst findet in der „Police“ Klasse statt, welche für die Auswahl des Adapters, für die Öffnung des Adapters und für die Auswertung der Netzwerkpakete verantwortlich ist.

Die IOHelper statische Klasse ist für die Auswahl des Speicherorts verantwortlich. Eine Ausnahme wird ausgelöst und behandelt, falls der Speicherort nicht erreichbar ist oder der Dateiname und die Dateiendung ungültig sind.

Die MessageHelper statische Klasse ist für die Rückmeldungen und Informationsausgaben verantwortlich. Die PrintMessage Methode nimmt einen Text und eine Meldungsbezeichnung. Die Meldungstyp bestimmt die Farbe der Ausgabe.

* 1. Geschäftslogik

Nach dem Starten der Anwendung wird der User nach einem Speicherort gefragt. Dieser Ort kann wie folgt aussehen:

*„C:\\DNS\_Protocol.txt“*

Diese Adresse ist der Standard-Speicherort, wenn der User einfach Enter drückt und nichts angibt. Eine Voraussetzung ist jedoch, dass die Anwendung mit Administratorenrechte gestartet werden muss, um auf den Laufwerk C:\\ schreiben zu dürfen. Wenn eine ungültige Location angegeben wird, bekommt der User die passende Fehlermeldung und die Analyse kann solange nicht durchgeführt werden, bis einen gültigen Speicherort mit „.txt“ Endung nicht eingetragen wird.

Der User wählt danach einen Netzwerkadapter aus und dadurch wird die Beobachtung des Netzwerkverkehrs gestartet. Die einzelnen Ereignisse werden in der Konsole ausgegeben aber auch gleichzeitig in den instanziierten Objekten intern gespeichert.

Wenn der Beobachtung abgebrochen wird, wird die Methode „SaveMonitoringResultsToTextFile“ Methode der EventRepository Klasse ausgeführt und das Resultat in der Text-Datei gespeichert. Falls ein Fehler auftritt, wird die passende Fehlermeldung ausgegeben und die Anwendung muss neugestartet werden.