Terminal-Config-Manager
Informatik für Anwendungsentwicklung
CHECK24 Tech Hub und Services GmbH

Adrian Schurz

1. November 2022

1 Projektantrag

1 Projektantrag

Der folgende Projektantrag wurde um die Auflagen, welche in der Terminbestätigung genannt wurden, erweitert. Weiterhin hat sich der Name der Firma, ohne jegliche Änderungen des Arbeitsverhältnisses, in der Zwischenzeit geändert und wurde hier aktualisiert.

Abschlussprüfung im Beruf Fachinformatiker für Anwendungsentwicklung
Winter 2022/23
Antragsformular
Azubi-Nr.: 480513
Name: Adrian Schurz

Ausbildungsbetrieb / Praktikumsbetrieb	CHECK24 Tech Hub und Services GmbH
Projektbezeichnung	terminal-config-manager
Projektbeschreibung	**Motivation**: Bei der Arbeit an Software-Anwendungen, welche eng mit einer Vielzahl anderer solcher Anwendungen über mehrere Umgebungen hinweg interagieren, ergaben sich zwei häufige Anwendungsfälle beim Umgang mit Konfigurationsdateien. 1. Das Einsehen und Kontrollieren von Einträgen innerhalb dieser Dateien. Dabei gibt es eine große Zahl an über verschiedene Verzeichnisse verstreuten Dateien. Innerhalb einer Datei gibt es jeweils eine große Zahl an Einträgen, von denen aber oft nur wenige relevant sind. 2. Das Ändern eines solchen Eintrags, um das Verhalten der zugehörigen Anwendung anzupassen. Sowohl das Kontrollieren als auch das Ändern von beliebigen Einträgen soll schnell und in einer Anwendung möglich sein. **Beispielszenario** In der Continuous-Integration-Pipeline (CI) einer zentralen Anwendung schlagen unvermittelt Akzeptanztests fehl. Um unverzüglich die Ursache feststellen zu können, muss das lokale Setup, das in diesem Moment an die Entwicklung einer anderen Anwendung und für die Ausführung einer anderen Menge an Tests konfiguriert ist, umkonfiguriert werden. **IST-Zustand**: Um die Zielanwendung zu untersuchen sind viele Teilaufgaben zu erledigen, beispielsweise das Aktivieren detaillierter

<table>
<tr><td></td><td>Lognachrichten, das Ändern der auszuführenden Unit-, Integration- und Systemtestsuites, das Ändern von Cache-Verhalten, das anpassen von Ziel-IPs anderer, an der Testausführung beteiligter Anwendungen. Jeder dieser Schritte ist wiederum mit mehreren Einzelschritten verbunden. Meist muss zu diesem Zweck eine passende IDE gestartet, die relevante Konfigurationsdatei ermittelt und zum richtigen Unterordner navigiert werden. Danach muss die Datei geöffnet, ihr Inhalt untersucht und der Zielwert geprüft und gegebenenfalls angepasst werden. In Summe sind sehr viele, repetitive Einzelaufgaben händisch zu erledigen. Hier bietet sich daher Potential zur Verbesserung.

Soll-Zustand:
Es existiert ein Kommandozeilenprogramm, das übersichtlich eine Liste anzeigt. Die Liste enthält pro Zeile einen Beschreibungstext und den aktuellen Wert innerhalb der Zieldatei. Das Programm erlaubt es, per Tastendruck Zeilen auszuwählen und den zugehörigen Wert aus einer Menge möglicher Werte auszuwählen. Wird ein neuer Wert ausgewählt, so wird die dazugehörige Konfigurationsdatei umgeschrieben. Sowohl die Zieldateien als auch die jeweiligen möglichen Werte sind konfigurierbar. Zusätzlich existieren ausführliche Unit- und Akzeptanztests.</td></tr>
<tr><td>Projektumfeld</td><td>Die CHECK24 Vergleichsportal GmbH, CHECK24 Vergleichsportal Reise GmbH und CHECK24 Tech Hub und Services GmbH sind Betreiber von check24.de, einer Website, auf der verschiedene Produkte zum Vergleich angeboten werden. Der Auftraggeber CHECK24 Tech Hub und Services GmbH betreibt auf dieser Online-Plattform eine Vergleichsmöglichkeit von Pauschalreisen.

Die Menge und Komplexität interner Anwendungen, die am Produktionsbetrieb und der Qualitätssicherung beteiligt sind, wächst stetig. Damit einher gehen komplexere Interaktionen und Konfigurationsmöglichkeiten, die häufige Fehlerquellen im Betrieb und während der Entwicklung darstellen. Je einfacher diese Konfiguration möglich ist, desto schneller kann während der Fehlersuche und der Entwicklung gearbeitet werden.</td></tr>
<tr><td>Projektphasen (einschließlich Zeitplanung)</td><td><table><tr><td>Phase</td><td>Dauer (h)</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Konzeption</td><td>10</td></tr>
<tr><td>Wahl des Techstacks</td><td>3</td></tr>
<tr><td>Einrichtung Entwicklungsumgebung</td><td>2</td></tr>
<tr><td>Implementierung</td><td>30</td></tr>
<tr><td>Qualitätssicherung</td><td>15</td></tr>
<tr><td>Dokumentation</td><td>10</td></tr>
<tr><td>Gesamt</td><td>70</td></tr>
</table>
</td></tr>
<tr><td>Dokumentation zur Projektarbeit (nicht selbstständig erstellte Dokumente sind zu kennzeichnen)</td><td>Das Projekt wird ausführlich dokumentiert. Dazu gehören:
Eine ausführliche, aus Quellcodekommentaren generierte Dokumentation der einzelnen Programmmodule im HTML-Format
Hilfetexte, die vom Programm selbst bei Bedarf ausgegeben werden
README-Dateien, welche das Aufsetzen des Projekts, die Kompilierung und Ausführung der Unit- und Akzeptanztests beschreiben
Eine Projektdokumentation im PDF-Format, welches eine IST-Analyse, die Anforderungen an die Software, die Projektziele, die Zeitplanung, den Projektverlauf und das Pflichtenheft beinhaltet.
Die Planung des Projekts
Die Umsetzung des Projekts
Eine Ergebnisdiskussion</td></tr>
<tr><td>Bearbeitungsdauer von</td><td>1.10.2022</td></tr>
<tr><td>Bearbeitungsdauer bis</td><td>14.11.2022</td></tr>
<tr><td>Präsentationsmittel</td><td>Laptop</td></tr>
<tr><td>Overheadprojektor</td><td>vorhanden</td></tr>
<tr><td>Projektionsbildschirm (als Beamer nutzbar)</td><td>vorhanden</td></tr>
<tr><td>Andere Präsentationsmittel (sind vom Prüfling mitzubringen)</td><td>Laptop</td></tr>
</table>

Themenbetreuer	Jessica Parth Falk Döring

Themenbestätigung

Folgendes wurde aus Ihrem Schreiben vom 23.09.2022, der Zustimmung des betrieblichen Auftrages mit Auflagen, übernommen:

Thema bestätigt	
Mit Auflagen bestätigt	X Projektdokumentation umfasst auch: Planung, Umsetzung, Ergebnisdokumentation -> Planung entsprechend anpassen
Grund Ablehnung	

2 Nachweisblatt

Inhaltsverzeichnis

3 Problemstellung

4 Literaturverzeichnis

5 Anlagen

6 Kundendokumentation

6.1 Beschreibung

Terminal-Config-Manager ist ein Linux-Programm mit welchem Textpassagen innerhalb meherer Dateien schnell zwischen einer Reihe vorkonfigurierter Textpassagen umgeschalten werden können.
Der Hauptanwendungsfall ist die effiziente Manipulation von Konfigurationsdateien von Softwareanwendungen, die häufig angepasst werden müssen.

6.2 Installation

Es wurden vorkonfigurierte Pakete für sowohl ArchLinux-basierte als auch Debian-basierte Betriebssysteme bereitgestellt. Alternativ kann das Programm auch manuell installiert werden.

Arch-Linux, via PKGBUILD Datei und pacman

Im Projektverzeichnis unter

```
/distribution/arch/PKGBUILD
```

befindet sich eine Spezifikationsdatei anhand derer das Softwarepaket erstellt und anschließend installiert werden kann:

```
cd distribution/arch
makepkg
pacman -U terminal-config-manager-1.0.0-1-x86_64.pkg.tar.zst
```

Die Deinstallation erfolgt mittels

```
pacman -R terminal-config-manager
```

Debian, via .deb Datei und dpkg bzw. apt

Im Projektverzeichnis unter

```
/distribution/debian/terminal-config-manager.deb
```

befindet sich ein Softwarepaket, das mittels `dpkg` oder `apt` direkt installiert werden kann.

```
cd distribution/debian
dpkg --install ./terminal-config-manager.deb
# apt install ./terminal-config-manager.deb
```

Die Deinstallation erfolgt mittels

```
dpkg --remove terminal-config-manager
# apt remove terminal-config-manager
```

Alternative, ohne Paketmanager

Wenn das Programm nicht vom systemeigenen Paketmanager verwaltet werden soll, dann kann es manuell kompiliert und in einem passenden Verzeichnis abgelegt werden.
Voraussetzung hierfür ist, dass das Programm `stack` auf dem System installiert ist.
Im Projektverzeichnis wird mit

```
stack build --test --copy-bins
```

das Programm kompiliert, die Testsuite ausgeführt und die ausführbare Datei im Projektverzeichnis unter

```
bin/terminal-config-manager
```

abgelegt. Anschließend kann das Programm in ein Verzeichnis kopiert werden, das in die Systempfadliste eingetragen ist, beispielsweise

```
cp bin/terminal-config-manager ~/.local/bin
```

Die Deinstallation erfolgt mittels

```
rm ~/.local/bin/terminal-config-manager
rm <Konfigurationsdateipfad>
```

6.3 Konfiguration

Die Zieldateien und -textpassagen müssen vor Ausführung des Programms über eine Datei im YAML-Format [3] konfiguriert werden.

Abbildung 1: Beispielaufbau der Konfigurationsdatei

```yaml
config_lines_to_manage:
  - title: Beispieltitel 1
    path: /home/alice/zieldatei.conf
    pattern: "'statspush_enabled' => {{value}},"
    targetValue: "true"
    possibleValues:
      - "true"
      - "false"

  - title: Beispieltitel 2
    path: /home/alice/verzeichnis/weitere-zieldatei.txt
    pattern: "SOFTWARE_ENV={{value}}"
    targetValue: production
    possibleValues:
      - testing
      - staging
      - production
      - local

  - ...
```

Verzeichnis Das Programm erwartet, dass sich eine solche Datei in einem der folgenden Verzeichnisse befindet. Die Reihenfolge entspricht der absteigenden Priorität beim Vorhandensein mehrerer Konfigurationsdateien:

1. ./config.yaml
2. ${HOME}/.config/terminal-config-manager/config.yaml **(empfohlen)**
3. ${HOME}/.terminal-config-manager.yaml

Der Dateipfad 1 bezeichnet den Ort der ausführbaren Datei selbst und sollte nur zu Debugging- oder Entwicklungszecken genutzt werden. Die Pfade 2 und 3 sind gängige Ablageorte für nutzerspezifische Konfigurationsdateien unter Linux.

Aufbau In Abb. 1 ist der Aufbau der Konfigurationsdatei illustriert. Das äußere Element

```yaml
config_lines_to_manage:
```

muss vorhanden sein. Innerhalb dessen befindet sich eine Liste von Einträgen. Jeder Eintrag gehört zu genau einer Textpassage, die mithilfe des Programms gezielt verändert werden soll.
Jeder Eintrag hat folgende Unterelemente:

- ```
  title: Beispieltitel 1
  ```

  Der Titel kann frei gewählt werden. Idealerweise wird dafür eine sehr kurze Beschreibung des von der Ziel-Textpassage gesteuerten Anwendungsverhaltens benutzt.

- ```
  path: /home/alice/zieldatei.conf
  ```

  Hier wird der absolute Dateisystempfad der Zieldatei angegeben innerhalb derer Textpassagen verändert werden sollen.

- ```
  pattern: "'statspush_enabled' => {{value}},"
  ```

  Dieses Feld enthält ein Zielmuster Mithilfe dieses Musters wird vom Programm der genaue Ort der

6.4 Benutzung

6.5 Problembehandlung

Fehler, keine Config -¿ die Datei fehlt an einem der üblichen Zielorte. -¿ Beispielconfig nehmen Fehler, Config parsing -¿ Die erste gefundene Configdatei ist falsch formatiert und/oder unvollständig -¿ Format prüfen mittels Schema oder mit Beispiel-Config vergleichen oder anhand der Parsing-Fehlermeldung den Fehler beheben.

Literatur

[1] *Jinja2 Project Homepage*. URL: `https://jinja.palletsprojects.com/en/2.11.x`.

[2] *Wikipedia, Template-Engine*. URL: `https://de.wikipedia.org/wiki/Template-Engine`.

[3] *YAML Project Homepage*. URL: `https://yaml.org`.
  ```

Glossar

Platzhalter Ein vordefinierter Text: das englische Wort `value` umgeben von doppelten geschweiften Klammern: `{{value}}` Das Format des Platzhalters ist an die die Template-Engine [2] Jinja2 [1] angelehnt. Er ist Teil des Zielmusters das zusätzlich Text vor und nach der Ziel-Textpassage enthält. Er markiert den Ort der Ziel-Textpassage relativ zum Zielmuster innerhalb der Zieldatei. 1, 5

Textpassage Ein Stück Text aus einer Textdatei. Dabei wird sich meist auf eine in der Konfigurationsdatei spezifizierte Zieldatei des Programms bezogen. 1, 4, 5

Zielmuster Ein Element innerhalb der Konfigurationsdatei. Es bezeichnet ein Stück Text das einen speziellen Platzhalter, den Text `{{value}}`, enhält. Es wird vom Programm dazu genutzt den genauen Ort der Ziel-Textpassage innerhalb der Zieldatei zu identifizieren. Das Zielmuster sollte 1, 4, 5