Abschlussprüfung Winter 2016

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Webapplikation zur Verwaltung der Zugriffsrechte der Benutzeraccounts externer Services

Webbasiertes Tool zur Unterstützung der Entwickler

Abgabetermin: Nürnberg, den 15.12.2016

Prüfungsbewerber:
Farah Schüller
Straße 123
1337 Stadt

Ausbildungsbetrieb:
SUSE Linux GmbH
Maxfeldstraße 5
90409 Nürnberg

Inhaltsverzeichnis

Abkürzungsverzeichnis

1 Einleitung

1.1 Projektumfeld

Die SUSE Linux GmbH wurde 1992 gegründet und ist seit 2014 eine Tochtergesellschaft der Micro Focus AG. Weltweit beschäftigt das Unternehmen etwa 750 Mitarbeiter, welche auf die Standorte Nürnberg, Prag, Peking und Provo (USA) verteilt sind. Der Hauptsitz der Firma befindet sich in Nürnberg, ebenso wie der Hauptteil der Softwareentwicklung.

Das Kerngeschäft der SUSE Linux GmbH umfasst die Entwicklung einer Linux-basierten Distribution. Für Privatkunden werden hierzu die Distributionen der openSUSE-Familie, die größtenteils von der Community entwickelt werden, bereitgestellt, für Geschäftskunden die Produkte der SUSE Linux Enterprise-Familie.

Die Entwicklung erfolgt nach dem Open-Source-Prinzip. Durchgeführt wurde das Projekt im Team SUSE IT, die neben der Infrastruktur auch die Entwicklungsumgebung, die sogenannten Engineering Services, bereitstellen.

1.2 Projektbeschreibung

Zur Entwicklung der Open-Source-Software setzt die SUSE Linux GmbH in Teilen der Entwicklung auf externe Tools und Services. Dies wird besonders im Bereich der Codeentwicklung und -pflege deutlich, da neben den firmeneigenen Entwicklern auch Partner sowie Open-Source-Community mitarbeiten. Hierbei kann es zu der Diskrepanz kommen, dass Entwickler neben ihrem Firmenaccount auch private Accounts zur Entwicklung benutzen, wodurch eine genaue Zuordnung der Accounts und ein Überblick über Zugriffsberechtigungen nahezu unmöglich wird.

Zur Lösung dieser Problematik soll eine Applikation entwickelt werden, welche es ermöglicht eine Zuordnung der firmeninternen Accounts auf mögliche Accounts externer Services durchzuführen. Das Ergebnis soll mittels einer API abrufbar sein. Dabei soll die Applikation in ihrer Grundfunktion dem Gedanken der Self-Service-Technologies folgen, damit Latenzen bei Änderungen der Accountinformationen minimiert und administratives Personal entlastet wird. Um dies möglichst plattformunabhängig zu gestalten, erfolgt die Realisierung als Webapplikation.

Da externe Services eine Zulassung durch den Betriebsrat und die IT-Sicherheit benötigen, beschränkt sich die erste Version darauf, die Accountinformationen der Applikationen Github (verzeichnisorganisierte Versionsverwaltung von Quellcode) und Trello (Tool zur Steuerung von agilen Software-Projekten) einzubinden.

1.3 Ist-Analyse

Bei den verwendeten externen Services Github und Trello können Unternehmen zur Abgrenzung und Verwaltung ihrer firmeninternen Ressourcen und Informationen Organisationen anlegen, zu denen

Accounts von Mitarbeitern hinzugefügt werden und ihnen besondere Zugriffsrechte einzuräumen. So haben beispielsweise nur Mitarbeiter Schreibrechte auf den Quellcodeverzeichnissen bei Github und Mitglieder der Open-Source-Community sowie Partner können in der Regel nur Vorschläge, sogenannte Pull Requests, bei dem jeweiligen Projekt einreichen.

Der Abgleich der internen Mitarbeiter- mit den Userlisten innerhalb der angelegten Firmenorganisationen erfolgt bisher manuell. Dies bedeutet, dass beispielsweise für jeden neuen Mitarbeiter persönlich oder schriftlich angefragt werden muss welches User-Alias er bei den jeweiligen Services führt, um dessen Account dann zur Organisation hinzuzufügen und ihm Zugriff zu ermöglichen. Ist dieser Aufwand bereits groß, erhöht er sich bei Mitarbeiteraustritten, wenn es gegebenenfalls auf Grund fehlender Kontaktmöglichkeit sehr mühsam wird eine Zuordnung nachzuvollziehen.

Bei Organisationen mit mehreren hundert Mitgliedern entsteht dadurch ein großer zeitlicher und personeller Aufwand, welcher bisher von einigen wenigen Administratoren gestemmt wird. Durch die ungenaue Beschaffung der benötigten Informationen ist das bisherige Konzept sehr fehleranfällig und ineffizient durchzuführen.

1.4 Soll-Konzept

Um Personal zu entlasten und den Prozess wesentlich zu vereinfachen, soll eine zentrale Plattform geschaffen werden, die eine gesammelte Zuordnung der Mitarbeiter zu ihren externen Nutzerkonten zum Abgleich bereitstellt. Die Datensätze sollen von den jeweiligen Mitarbeitern selbst verwaltet und gepflegt werden. Dadurch bleibt alleine der Abgleich mit den Nutzerlisten der bei den externen Services angelegten Organisationen und ein eventuelles Hinzufügen oder Entfernen des Mitarbeiters aus der Organisation übrig.

Die Plattform soll als Webapplikation gestaltet werden, was eine plattformunabhängige Nutzung ermöglicht. Zusätzlich soll die Implementierung modular und leicht erweiterbar sein, um zukünftig geplante Funktionen einfach hinzufügen und warten zu können. Ebenso soll die Applikation über Tests verfügen, welche jene für andere Entwickler in Zukunft nachvollziehbar macht.

1.5 Projektschnittstellen

Die Applikation nutzt mehrere APIs um z.B. die Betriebszugehörigkeit oder verschiedene Informationen aus den Datenbanken der externen Services abzufragen.

Mitarbeiterspezifische Informationen, wie Name, Standort oder Vorgesetzter, werden aus dem firmeneigenen eDirectory abgefragt. Zur Abfrage der Organisationszugehörigkeit werden die externen Schnittstellen der genannten Tools Github und Trello verwendet.

Die Schnittstelle der Applikation, welche die gesammelten Informationen über den Mitarbeiter bereitstellt, wird in dem Administrationstool etsync konsumiert, welches für die Administratoren der externen Organisationen bei Github und Trello entwickelt wurde.

2 Projektplanung

2.1 Projektphasen

2.2 Ressourcenplanung

2.2.1 Personalplanung

Das für die Entwicklung benötigte Personal besteht aus einem Auszubildenden, der dafür von seiner regulären Mitarbeit im derzeitigen Team freigestellt wird.

2.2.2 Kostenplanung

Die Personalkosten belaufen sich auf ein Auszubildendengehalt, wodurch lediglich Stromkosten für die zur Projekterstellung eingesetzte Hardware anfallen. Durch die ausschließliche Nutzung einer Entwicklungsumgebung und Werkzeugen aus dem Open-Source-Bereich fallen keine zusätzlichen Kosten wie Software-Lizenzen an.

2.3 Entwicklungsprozess

Der Entwicklungsprozess erfolgt test-driven, was bedeutet dass vor dem Schreiben des eigentichen Quellcodes einer Funktionalität zunächst der dazugehörige Test entwickelt wird. Diese Vorgehensweise ermöglicht einen abstrakteren und zielorientierteren Blick auf die eigentlichen geforderten Funktionen der Applikation, in dem als erstes eine Überlegung über das Ergebnis einer Funktion und daraufhin die eigentliche Implementierung des jeweiligen Algorithmus erfolgt, welcher zu dem gewünschten Ergebnis führt.

2.4 Architekturdesign

Zur Umsetzung der geforderten Modularität der Applikation bietet sich das sogenannte *Model-View-Controller-Schema* (MVC) an. Dieses ist ein gängiges Schema beim Aufbau von Webapplikationen und besteht aus drei Teilen:

- das *Model*, welches grob die einzelnen Tabellen einer Datenbank repräsentiert. Es enthält in der Regel zusätzliche Logik und Regeln, die zur Verwaltung und Zusammensetzung von Attributen eines Datenbankobjektes notwendig sind.
- der *View*, welcher eine Ausgabe der gewünschten Daten bereitstellt. Im Kontext einer Webapplikation ist dies die eigentlich dargestellte Webseite in HTML/CSS.

- der *Controller*, welcher die Schnittstelle zwischen Model und View darstellt. Im Controller werden Eingaben verarbeitet und Befehle an das Model weitergegeben, welche die angeforderten Informationen an den Controller zurückgibt, der diese wiederum an den entsprechenden View verteilt.

Diese Aufteilung ermöglicht ein strukturierte und übersichtliche Entwicklung, da Funktionslogik, Datenbankoperationen und Ausgabelogik klar getrennt und jeweils namentlich zugehörig gekennzeichnet sind.

TODO: enter figure of MVC schematics

2.4.1 Wahl der Programmiersprache

2.4.2 Wahl des Frameworks

2.5 Benutzeroberfläche

- Twitter Bootstrap
- functional drop-in solution as there's no designer available

2.6 Datenmodell

- insert database model chart here?
- single table inheritance
- tool model for every kind of tool available

2.7 Rechteverteilung

- general access for regular employee: maintenance of record
- admin access for list of all records and API

2.8 Schnittstellen

- API for etsync

2.9 Paketierung

- RPM
- all dependencies included