Projektplan

Silvan Adrian
Fabian Binna

1 Änderungshistorie

Datum	Version	Änderung	Autor
17.09.15	1.00	Erstellung des Dokuments	Gruppe
18.09.15	1.01	Organisationsstruktur + Einführung + Projektübersicht	Silvan Adrian
18.09.15	1.02	Qualitätsmassnahmen	Fabian Binna
20.09.15	1.03	Text verbessert und zusätzliche Infos eingefügt	Silvan Adrian
25.09.15	1.04	Verbesserungen	Silvan Adrian
25.09.15	1.05	Sprint Planungsübersicht eingefügt	Silvan Adrian
26.09.15	1.06	Testing erweitert/verbessert + Logging	Fabian Binna
02.10.15	1.07	Verbesserungen gemäss neusten Erkenntnissen	Silvan Adrian
03.10.15	1.08	Testing angepasst + IDE definiert	Fabian Binna
04.10.15	1.09	Beschreibungen von Sprintplanung verbessert	Silvan Adrian
14.10.15	1.10	Verbesserungen Projektplanung + Features	Silvan Adrian
19.10.15	1.11	Verbesserungen Projektplanung + Testing	Silvan Adrian
25.10.15	1.12	Metriken verbessert + Testing	Silvan Adrian
26.10.15	1.13	Testing Anpassungen	Fabian Binna

Inhaltsverzeichnis

2 Einführung

2.1 Zweck

Dieses Dokument beschreibt die Planung der SA SDDC.

2.2 Gültigkeitsbereich

Dieses Dokument ist während des ganzen Projekts gültig und wird laufend aktualisiert.

2.3 Referenzen

2.3.1 APIs:

http://developer.openstack.org/
http://libvirt.org/
https://jclouds.apache.org/

2.3.2 SCRUM Guide:

http://www.scrumguides.org

3 Projektübersicht

Es soll eine generische API erstellt oder eine bestehende API verwendet werden, um über ein Dashboard direkt Services zu abonnieren, dabei können bei der API Vorlagen hinterlegt werden, welche mehrere Servicemodule beinhalten (Compute,Storage,Network). Die API soll dabei möglichst modular aufgebaut sein, damit Neuerungen einfach auf den neusten Stand gebracht werden können.

3.1 Zweck und Ziel

Gelerntes aus verschiedenen Modulen anwenden, sich mit der gestellten Aufgabenstellung auseinandersetzen und diese umsetzen.

3.2 Primäre Features

- Generische API
- Workflow zum abonnieren/kündigen
- Unterstützung von mindestens 1 Compute Anbieter
- Unterstützung von mindestens 1 Storage Anbieter
- Unterstützung von mindestens 1 Network Anbieter
- Admin Dashboard um Services verwalten zu können

- Spartanisches User-Dashboard

3.3 Erweiterte Features

- Admin-Dashboard um Funktionen erweitern
- Unterstützung für mehr Compute Anbieter
- Unterstützung für mehr Storage Anbieter
- Unterstützung für mehr Network Anbieter

3.4 Lieferumfang

- Source-Code
- Projektdokumentation
- Generische API Dokumentation
- Restful API Dokumentation

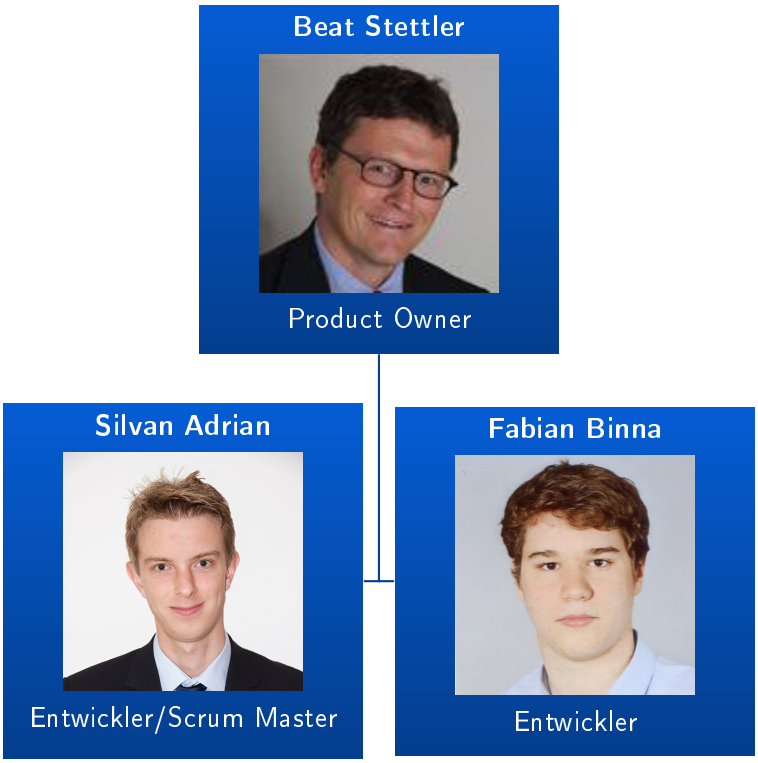
3.5 Annahmen und Einschränkungen

Die Applikation setzte sich aus einem Admin-Dashboard und einer generischen API/Restful API zusammen, die sollten dabei verteilt betrieben werden können. Das User-Dashboard würde lediglich zum Testen der API verwendet. Die generische API soll dabei so viele Anbieter wie möglich abdecken.

4 Projektorganisation

Wir setzen beim Project auf Scrum, wodurch lediglich die Projektrollen Product Owner, Entwickler und Scrum-Master zur Verfügung stehen.

4.1 Organisationsstruktur



4.2 Externe Schnittstellen

Projektbetreuer:
Beat Stettler (Beat.Stettler@ins.hsr.ch)
Urs Baumann (Urs.Baumann@ins.hsr.ch)

5 Managment Abläufe

5.1 Kostenvoranschlag

Gemäss Vorgaben wird mit 480 Stunden gesamt gerechnet.
Gesamtaufwand: 480 Stunden
Aufwand pro Woche: ca.: 34 Stunden
Aufwand pro Woche pro Teammitglied: ca.: 17 Stunden

5.2 Planung

Name	Beschreibung	Beginn	Ende
Analyse Anforderung Design	Analyse der vorhandenen API's + Anforderungen erstellen + Design Grundlagen erstellen	14.09.15	19.10.15
Sprint 1	Grundgerüst generische API, um Storage, Compute und Network abonnieren/kündigen zu können + Workflow abonnieren/löschen	19.10.15	02.11.15
Sprint 2	Restful Interface für die API, um Services über HTTP abonnieren zu können	02.11.15	16.11.15
Sprint 3	Admin Dashboard, um Services/Servicemodule verwalten und ihnen ein Konfigurationsdatei hinterlegen zu können	16.11.15	30.11.15
Sprint 4	User-Dashboard (sehr spartanisch) um Services abonnieren/kündigen zu können + letzter Release der Software	30.11.15	11.12.15
Abschluss	Studiengang Arbeit zur Abgabe vorbereiten + Präsentation vorbereiten	11.12.15	18.12.15

5.3 Besprechungen

Daily Meetings (ca. 10 - 15min) werden während der Pause oder über Skype stattfinden, um sich gegenseitig über Probleme, abgeschlossene Arbeitspakete informieren zu können. Nach jedem Abschluss eines Sprints wird ein Sprintreview geplant und durchgeführt, um zu prüfen ob alles gemäss Plan erledigt wurde + die aktuellste Version der Software released.

6 Risikomanagement

6.1 Risiken

Risiken werden im Dokument Initiales_Risikomanagement.pdf beschrieben

6.2 Umgang mit Risiken

Für die Risiken werden Reserven eingeplant. Die Reserven werden direkt in die einzelnen Tickets einberechnet. Falls Risiken eintreffen werden diese sofort an einem der Daily-Meetings kommuniziert und mögliche Lösungen kurz evaluiert.

7 Arbeitspakete

Die Arbeitspakete werden in Open Project erstellt und gepflegt. Lesender Zugriff ist anonym möglich, schreibender nur als Benutzer.
Link zur Open Project Instanz : http://sddc.silvn.com

8 Infrastruktur

8.1 Entwicklungsinfrastruktur

Name	Hardware	Betriebssystem	IDE
Silvan Adrian	MacBook Pro	OSX 10.10.5	Eclipse Mars
Fabian Binna	Lenovo T430s	Windows 10	Eclipse Mars

8.2 Tools/Software

- **BuildServer:** Travis-CI
- **Versionsmanagement:** GIT
- **Notifications:** Slack
- **Wireframing:** Pencil
- **Test Coverage:** Cobertura
- **Bugs Vorbeugung:** Findbugs
- **Metriken:** SonarQube

8.3 Kommunikationsmittel

- E-Mail
- Skype
- Open Project
- GitHub
- Whatsapp
- Slack

9 Qualitätsmassnahmen

9.1 Dokumentation

Die Dokumentation befindet sich auf einem privaten GitHub Repository (SDDC_Doku). Die Texte werden in LaTeXgeschrieben, aber auch als PDF auf das Repository gestellt.

9.2 Projektmanagement

Für das Projektmanagement wird OpenProject verwendet.

http://sddc.silvn.com

9.3 Metriken

Für Metriken wird SonarQube verwendet.

http://sonar.silvn.com

9.4 Entwicklung

9.4.1 IDE

Eclipse Mars

9.4.2 Unit Testing / Test-Driven Development

Die Unit Tests kommen in einen separaten Ordner "Test". Es wird eine möglichst hohe Code Coverage angestrebt. Die Code Coverage wird mit einem Tool sichergestellt.

Die Klassen werden mit Hilfe von Test-Driven Development implementiert.

9.4.3 Code Review

Nach jedem Sprint oder bei Abschluss grosser Arbeitspakete wird ein Code Review durchgeführt.

Die Review Protokolle werden auf dem GitHub Repository SDDC-Doku abgelegt.

9.4.4 Logging

Für das Logging wird log4j verwendet.

TRACE: Ausführliches Debugging
DEBUG: allgemeines Debugging

INFO: allgemeine Informationen
WARN: Auftreten einer unerwarteten Situation
ERROR: Fehler
FATAL: Kritischer Fehler

9.4.5 Metrikanalyse

Die Metriken werden nach jedem Sprint protokolliert und Analysiert. Die Metriken helfen während einem Code-Review auf problematische Code Stücke hinzuweisen. Sie können auch beim Debugen hilfreich sein.

9.5 Testen

9.6 Modultest

Die Modultests werden, wie schon beschrieben ,während der Entwicklung mit Test-Driven Development erstellt. Zu jedem Modul werden Tests geschrieben, um eine möglichst hohe Code Coverage zu erreichen. Die Tests werden nochmals auf dem Build Server ausgeführt um "it works on my machine" abzudecken.

Produkte:

- JUnit Tester
- Travis
- SonarQube Analyse

9.7 Integrationstest

Bei den Integrationstests kommt eine Kombination aus automatisierten Tests, sowie eine Analyse des Ablaufs zum Einsatz. Die Tests werden an verschiedenen Schichten angesetzt (Presentation und Business-Layer). Die Unteren Schichten, die für den Integrationstest nicht relevant sind werden gemockt. Nachdem die automatische Tests durchgelaufen sind, werden die Log-Daten ausgewertet und mit den Sequenzdiagrammen abgeglichen. Es ist wichtig, das die Komponenten korrekt miteinander kommunizieren.

Produkte:

- JUnit Tester
- Log-Daten

9.8 Systemtest

Die Systemtests richten sich nach den User Stories und deren Akzeptanzkriterien. Die Kriterien werden von einem Tester nach Protokollangaben getestet und dokumentiert. Auch hier können Log-Daten eine Hilfe für die Analyse sein.

Systemtests werden nach jedem Sprint durchgeführt um die geplante Funktionalität zu testen und so die neuste Version der Software releasen zu können.

Produkte:

- Protokoll
- Log-Daten

9.9 Abnahmetest

Der Abnahmetest wird vom Kunden selbst durchgeführt.