Zeitplan

Tsomo Taf

January 2024

Zusammenfassung Projektarbeit

SDLC	Istzustand	Idealzustand	Sollzustand
Requirements	-**Endbenutzer**: Kfz-Mechaniker von AW4.0 -**Projektpartner:** AW4null -**Projektmanager & Entwickler**: Herr Bökelmann Mitglieder des Vereins open Skunkforce e.VProjektmanager	**User:** Mechaniker/Käufer von Gebrauchtwagen **Projektpartner:** AW4null **DevTeam:** Mitglieder des Vereins open Skunkforce e.VProjektmanager **Product Owner (PO):** Herr Bökelmann **Betriebspersonal:** **DevOpsteam** **Scrum Master: x**	**User:** Mechaniker/Käufer von Gebrauchtwagen **Projektpartner:** AW4null **DevTeam:** Mitglieder des Vereins open Skunkforce e.VProjektmanager **Product Owner (PO)/ Scrum Master:** Herr Bökelmann **Betriebspersonal:** **DevOps Engineer**
Planning	Information nicht zugänglich	**Vor-Commit-Phase** Besprechungen zur Sprintplanung und Review. Einführung eines Ticketmanagementsystems **-Ereignisse**: Sprint Planning Sprint-Zyklus Sprint-Review Daily-Scrum Retrospecktive) **-Frameworks als Taskt Manager:** Microsft Team, slack, Trello oder jira **Tools Dokumentation:** Confluence	**Vor-Commit-Phase** Besprechungen zur Sprintplanung und Review. Einführung eines Ticketmanagementsystems **-Ereignisse:** Sprint Planning Sprint-Zyklus Sprint-Review weekly-Scrum Retrospecktive) **-Frameworks als Taskt Manager:** Microsft Team, slack oder jira

Design/Architecture	**keine ausreichenden Informationen** -Monolithische Architektur -Benutzeroberfläche (GUI) -Lokale Datenverarbeitung -Event-Driven Programming -Speicherung von Konfigurationen und Daten lokal in Festplatte -nur Offline-Fähigkeit -API-Design Frontend-Framework: Dear ImGui	**Option 1: Monolithische Architektur Hinzufügen:** Modularisierung Separation of Concerns: -Datenbank-Optimierung Logging und Überwachung Sicherheit **Option 2: Microservices-Architektur** Client-Server-Architektur Microservices-Architektur: Frontend-Framework: Backend-Technologien Datenbank-Management API-Design Skalierbarkeit und Lastverteilung Containerisierung und Orchestrierung Cloud-Plattform	-Monolithische Architektur Benutzeroberfläche (GUI) -Lokale Datenverarbeitung -Event-Driven Programming -Speicherung von Konfigurationen und Daten lokal in Festplatte -nur Offline-Fähigkeit -API-Design -Modularisierung Separation of Concerns: -Datenbank-Optimierung Logging und Überwachung -Sicherheit
Implementation	1.Versionierungsverwaltung des Quellcodes von OmniView Git-Repository (GitHub) 2.Kontinuierliche Integration (CI): Eine CI-Pipeline ist so konfiguriert, dass sie das Quellcode-Repository auf Änderungen überwacht. Bei jedem Push oder Pull Request wird die Pipeline ausgelöst. 3.Die CI-Pipeline umfasst die folgenden Schritte: Kompilieren des Quellcodes.	1. Versionierungsverwaltung des Quellcodes von OmniView Git-Repository (GitHub) 2. Kontinuierliche Integration (CI): Eine CI-Pipeline ist so konfiguriert, dass sie das Quellcode-Repository auf Änderungen überwacht. Bei jedem Push oder Pull Request wird die Pipeline ausgelöst. 3. Die CI-Pipeline umfasst die folgenden Schritte: -Durchführung von Unit-Tests. -Statische Analyse des Codes.	1 und 2 - unverändert 3.Die CI-Pipeline umfasst die folgenden Schritte: -Kompilieren des Quellcodes. -Statische Analyse des Codes. -Generierung von Build-Artefakten. -Benutzung des bedingten ternären Operators, um den build zu vereinfachen **Framework :** VsCode, SonQube

	Generierung von Build-Artefakten. Framework : Vs Code Tools : Github action	-Generierung von Build-Artefakten. **Framework:** VsCode, SonQube, jUnit, Tools: Github action, jenkins, oder GitLab	Tools : Github action
Testing	**keine ausreichenden Informationen**	-Unit-Tests - Integrationstests -Akzeptanztests / Staging / Performance-Tests -Durchführung von Unit-Tests - GUI-Tests (z B. mit Selenium oder Appium) **Framework:** VsCode , SonQube, jUnit,	- Unit-Tests - Integrationstests -Akzeptanztests / Staging / Performance-Tests -Durchführung von Unit-Tests. -GUI-Tests (z. B. mit Selenium oder Appium) **Framework:** VsCode, SonQube, jUnit,
Deployment	-Nach genügend Feature wird ein neues Release produziert. -Die CD-Pipeline (Release.yaml) wird wie folgt ausgeführt: -Nach dem Build (Linux/Windows) -Upload des generierten Artefakts -Erstellung der Release Upload Release Assets -die neuen Tags wird hier gespeichert: {steps.create_release.outputs. upload_url }	-Nachdem dem Code erfolgreich gebaut und getestet wurde -Nach genügend Feature wird ein neues Release produziert. -Kopieren des gebauten Codes auf einen Server oder eine Cloud-Plattform oder Bereitstellung des Endartefakts [partieller] Produktivbetrieb -Freigabe in den normalen Produktivbetrieb Die CD-Pipeline sollte wie folgt ausgeführt werden: **Option 1: Cloud/Webanwendungen** • Anwendung zusammen mit ihren Abhängigkeiten mit Docker Container zu verpacken • Container auf der Cloud-Plattform zu orchestrieren **Option 2: Microservices**	-Nachdem dem Code erfolgreich gebaut und getestet wurde -Nach genügend Feature wird ein neues Release produziert. -Kopieren des gebauten Codes auf einen Server Die CD-Pipeline sollte wie folgt ausgeführt werden: **Option1: Repository Manager** • Erstellung eines Docker images von omniview-software • Veröffentlichung des Artefaktes auf einer Artefaktory Manager-Repository, wie Nexus • Die Kunden bekommen Nexus-Anmeldeinformationen **Option 2: Webseite** •eigenen Website bereitstellen

		• jeden Microservice in einem eigenen Container zu verpacken •Erstellung der Infrastruktur mit Terraform (Multi-Cloud) •Ansible für die Bereitstellungskonfiguration •Kubernetes ermöglicht es Ihnen, diese Container zu orchestrieren **Option 3: Desktop-Anwendungen** •eigenen Website bereitstellen •Online-Download bereitstellen **Tools:** ImGui	•Online-Download bereitstellen **Tools**: ImGui, Nexus
Feedback & Bugreporting	**-Intern:** GitHub Issues wird von den Beitragenden des Projekts fur den Austausch, die Verfolgung des Projektverlaufs und die ¨ Verwaltung von Bugs genutzt. **-Benutzer:** über die Verwaltung von Feedback und Bugreporting bei AW4null, keine Informationen	-Implementierung einer benutzerfreundlichen Schnittstelle: -Teststrategien bei Monitoring und Logging: Beta-Tests und Canary-Tests Monitoring-Lösungen: Prometheus oder Grafana	Implementierung einer benutzerfreundlichen Schnittstelle: -Interne Feedback-Funktionen: Dies kann in Form von Menüoptionen, Dialogfeldern -Online-Feedback-Portale: Online-Feedback-Portal **Tools:** Jira oder Bugzilla