Dokumentation

Bilfinger Wartungscenter

Florian Hansen

30. März 2020

1 Einleitung

Das *Bilfinger Wartungscenter* ist eine Web-Anwendung, die eine Verwaltung von Wartungsarbeiten ermöglicht. Hierfür stehen dem Benutzer verschiedene Funktionen zur Verfügung, um Wartungen einzusehen, zu bearbeiten und Berichte erstellen zu lassen. Das Ziel ist es, dem Benutzer einen guten Überblick über anstehende und bereits getätigte Wartungsarbeiten zu geben und ihm damit einen Großteil der Verwaltungsarbeit abzunehmen.

2 Anforderungen

In diesem Kapitel sollen die Anforderungen an das zu entwickelnde System analysiert und definiert werden. Hierbei wird zwischen allgemeinen, funktionalen und nicht-funktionalen Anforderungen unterschieden.

2.1 Allgemeine Anforderungen

1. Software-Architektur

1.1. Die Anwendung soll mehrere Microservices erhalten, die komplett unabhängig voneinander arbeiten.
1.2. Die Microservices sollen mithilfe von NodeJS / Express entwickelt werden.
1.3. Die Microservices sollen mithilfe von Schnittstellen (APIs) kommunizieren.
1.4. Die APIs sollen im RESTful-Design implementiert werden.
1.5. Die Clientanwendung soll mithilfe von JavaScript / React entwickelt werden.
1.6. Die Clientanwendung greift mithilfe der definierten RESTful-Schnittstellen auf die verschiedenen Microservices zu.

2. Systemanforderungen

2.1. Die Andwendung soll plattformunabhängig sein.
2.2. NodeJS soll als serverseitige Sprache verwendet werden, d.h., dass dies auf dem ausführenden System installiert werden muss.
2.3. Auf dem ausführenden System (serverseitig) soll Docker installiert werden, um Microservices zu installieren bzw. zu laden.

2.2 Funktionale Anforderungen

1. Benutzerverwaltung

1.1. Benutzer sollen sich anmelden können.
1.2. Benutzer sollen über eine Rolle verfügen.
1.3. Mögliche Rollen sollen sein: Verwalter, Wartungskraft.
1.4. Benutzer sollen einen Benutzernamen, einen Vornamen, einen Nachnamen, eine eindeutige Identifikationsnummer (ID) und eine E-Mail-Adresse besitzen.
1.5. Benutzer sollen nur manuell registriert werden können.
1.6. Benutzer sollen ihre Daten, mit Ausnahme der ID, anpassen können.

2. Microservice: Authentifizierung

2.1. Es soll möglich sein, sich mithilfe seiner Benutzerkennung und seines Passworts anzumelden.
2.2. Nach einer erfolgreichen Anmeldung soll es möglich sein, sich mithilfe eines Tokens zu authentifizieren.
2.3. Die Tokens sollen als Json Web Token (JWT)[1] implementiert werden.
2.4. Ein Token soll nur für eine bestimmte Zeit verwendet werden können.
2.5. Nach Ablauf der Lebenszeit eines Tokens, muss sich der Benutzer erneut im System anmelden und einen neuen anfordern, um sich authentifizieren zu können.

3. Microservice: Wartungen

3.1. Dieser Dienst soll nur authentifizierten Benutzern zur Verfügung stehen.
3.2. Benutzer sollen Wartungen abrufen können. Dabei werden ihnen nur diejenigen zur Verfügung gestellt, die ihnen zugeteilt wurden.
3.3. Nur Benutzer der Rolle *Verwalter* sollen neue Wartungen erstellen können.
3.4. Es soll als Verwalter möglich sein, Wartungen zu bearbeiten und zu verändern.
3.5. Es soll als Verwalter möglich sein, Wartungen Wartungskräften zuzuornden.
3.6. Es soll als Verwalter möglich sein, Kunden anzulegen.

[1] https://jwt.io/

3.7. Es soll als Verwalter möglich sein, Kunden zu bearbeiten.
3.8. Es soll als Verwalter möglich sein, Kunden zu löschen.
3.9. Kunden sollen von Verwaltern Wartungen zugewiesen werden können.
3.10. Ein Kunde besitzt einen Namen, eine Ansprechsperson (Vor- und Nachname, Telefonnummer, E-Mail, ...) und eine Anschrift (Ort, PLZ, Straße)

4. Benutzeroberfläche: Anmeldung

4.1. Wenn der Benutzer nicht angemeldet ist, soll immer die Anmeldung aufgerufen werden.
4.2. Der Benutzer soll seinen Benutzernamen und Kennwort eingeben können.
4.3. Mithilfe eines Steuerelements soll der Anmeldeversuch durchgeführt werden können.
4.4. Beim Scheitern des Anmeldeversuchs soll dementsprechendes Feedback gegeben werden.
4.5. Beim erfolgreichen Einloggen soll der angemeldete Benutzer solange angemeldet bleiben, bis er sich wieder abmeldet.

5. Benutzeroberfläche: Dashboard

5.1. Nur angemeldete Benutzer sollen diesen Teil der Anwendung sehen können.
5.2. Das Dashboard soll dem Benutzer die Anzahl der anstehenden Wartungen in dem aktuellen Jahr anzeigen.
5.3. Das Dashboard soll dem Benutzer die Anzahl der erledigten Wartungen in dem aktuellen Jahr anzeigen.
5.4. Das Dashboard soll dem Benutzer die nächste anstehende Wartung anzeigen.
5.5. Der Benutzer soll mithilfe eines Steuerelements schnell auf die Übersichtsseite der dementsprechenden Wartung gelangen können.
5.6. Es soll eine Tabelle erstellt werden, die dem Benutzer die nächsten fünf anstehenden Wartungen anzeigt. Dabei soll der Kunde, der Ort, die Anlage und der Zeitpunkt des Termins angezeigt werden.
5.7. Durch einen Klick auf einen Tabelleneintrag soll der Benutzer zur dementsprechenden Übersichtsseite der Wartung gelangen können.

6. Benutzeroberfläche: Wartungen

6.1. Nur angemeldete Benutzer sollen diesen Teil der Anwendung sehen können.
6.2. Eine Tabelle soll dem Benutzer alle Wartungen anzeigen, die ihm zugeordnet wurden.
6.3. Der Benutzer soll die Einträge der Tabelle filtern können, z.B., um bereits abgeschlossene Arbeiten auszublenden.
6.4. Der Benutzer soll eine Wartung als abgeschlossen markieren können.
6.5. Benutzer mit der Rolle *Verwalter* sollen neue Wartungen erstellen und anderen Benutzern zuweisen können.
6.6. Wartungen sollen lediglich Kundendaten mit einem (wiederkehrenden) Zeitpunkt verbinden, um Wartungsarbeiten zu definieren.
6.7. Beim Erstellen von Wartungen soll die Wartungsperiode und das Ende des Wartungsvertrages angegeben werden können.
6.8. Verwalter sollen Wartungen löschen können.
6.9. Verwalter sollen Wartungen bearbeiten können.

7. Benutzeroberfläche: Kunden

7.1. Nur angemeldete Verwalter sollen diesen Teil der Anwendung sehen können.
7.2. Verwalter sollen neue Kunden hinzufügen können.
7.3. Kunden besitzen einen Namen, eine Anschrift und eine Ansprechsperson inkl. Kontaktinformationen.
7.4. Verwalter sollen Kunden löschen können.
7.5. Verwalter sollen Kundendaten bearbeiten können.

2.3 Nicht-funktionale Anforderungen

1. Sicherheit

1.1. Server- und Client-Anwendungen sollen die Integrität, Vertraulichkeit und Verfügbarkeit der Informationen gewährleisten.

2. Performance

2.1. Die Anwendung soll mit ausreichender Performance laufen.

2.2. Algorithmen sind so zu wählen, dass diese eine möglichst geringe Komplexität besitzen.

3. Usability

3.1. Die Anwendung soll mithilfe des Design-Systems *Material Design*[2] umgesetzt werden.

[2] https://material.io/design/