**Dokumentation für die WetterApp**

*Eine Web-Anwendung die Informationen über das Wetter zur Verfügung stellt*

Entwickler:

Hadis Nasibibour

Wessam Fahed

Christian Stein

Kilian Ketelhohn

****

# Einleitung

## 1.1 Projektbeschreibung

## 1.2 Projektziel

## 1.3 Projektumfeld

# 2. Projektplanung

## 2.1 Entwicklungsprozess

# 3. Analysephase

## 3.1 Anwendungsfälle

## 3.2Lastenheft

# 4. Entwurfsphase

## 4.1 Auswahl des Frameworks

## 4.2 Entwurf der Benutzeroberfläche

## 4.3 Entwurf des „Back-end“

## 4.4 Datenbank

## 4.5 Pflichtenheft

# 5. Implementierungsphase

## 5.1 Implementierung der Benutzeroberfläche

## 5.2 Implementierung des Backend

# 6. Fazit

## 6.1 Soll-/Ist-Vergleich

## 6.2 Lessons Learned

## 6.3 Ausblick

**Anhang**

1. Use-Case-Diagramm
2. Lastenheft (Auszug)
3. Wireframe
4. Pflichtenheft (Auszug)

# Einleitung

Die folgende Projektdokumentation schildert den Ablauf des zweiten Berufsschulprojektes, welches die Autoren im Rahmen ihrer Ausbildung zum Fachinformatiker mit Fachrichtung Anwendungsentwicklung durchgeführt haben.

## 1.1 Projektbeschreibung

Es soll eine Wetter-App entwickelt werden, die als Web-Anwendung aufgerufen werden kann. In der Web-Anwendung soll es möglich sein, aktuelle Wetterinformationen für verschiedene Orte einzusehen.

## 1.2 Projektziel

Ziel des Projektes ist es, innerhalb von zwei Schulwochen, die Wetter-App mit allen Anforderungen gemäß der Qualitätsstandards umzusetzen, welche aus der ISO/IEC 9126-1:2001 hervorgehen.

## 1.3 Projektumfeld

Erarbeitet wurde das Projekt an der Berufsschule für Technik in Schwerin. Dabei hat sich die Projektgruppe aus den folgenden Schülern zusammengestellt: Hadis Nasibipour, Wessam Fahed, Christian Stein und Kilian Ketelhohn.

## Projektplanung

In der Projektplanung soll die notwendige Zeit und die benötigten Ressourcen sowie ein Ablauf der Durchführung des Projektes geplant werden.

## 2.1 Entwicklungsprozess

Bevor mit der Realisierung des Projektes begonnen werden konnte, mussten sich die Autoren für einen geeigneten Entwicklungsprozess entscheiden. Dieser definiert die Vorgehensweise, nach der die Umsetzung erfolgen soll. Im Zuge des Projektes entschieden sich die Autoren für das Wasserfallmodell.

Beim Wasserfallmodell geht es darum, die einzelnen Entwicklungsschritte in einer linearen Abfolge zu organisieren. Im ersten Schritt werden die Anforderungen des zu entwickelten Systems erfasst und dokumentiert. Basierend auf den Anforderungen, der ersten Projektphase, wird dann die Funktionalität und Architektur des Systems entworfen.

In der Implementierungsphase werden dann die Entwürfe in Code umgesetzt. Nachdem alle Komponenten entwickelt wurden, werden sie zu einem ausführbaren System zusammengefasst. Zum Schluss wird den Benutzern das System vorgelegt und abgenommen. Daraufhin folgt über die Lebensdauer der Software eine kontinuierliche Weiterentwicklung und Verbesserung des Programmcodes.

# Analysephase

## 3.1 Anwendungsfälle

Um eine grobe Übersicht über die abzudeckenden Anwendungsfälle zu erhalten, wurde im Zuge der Analysephase ein Use-Case-Diagramm erstellt. Hierbei wurden die betroffenen Akteure identifiziert und deren Anforderungen an das Projekt ermittelt. Das Use-Case-Diagramm ist im Anhang a): Use-Case-Diagramm zu finden.

## 3.2 Lastenheft

Am Ende der Analysephase haben die Autoren zusammen ein Lastenheft erstellt. Ein Auszug des Lastenheftes ist im Anhang b): Lastenheft (Auszug) zu finden.

# Entwurfsphase

Als Folge der Analysephase wurde vor der eigentlichen Implementierung des Projektes eine Entwurfsphase durchgeführt. Hierbei wird entworfen, wie das System später aussehen soll und wie dies technisch umzusetzen ist. Am Ende der Entwurfsphase entsteht das Pflichtenheft, welches den Auftraggebern des Projektes vorgelegt wird.

## 4.1 Auswahl des Frameworks

Die Autoren haben sich dafür entschieden das Express JS Framework einzusetzen. Dieses Framework ist eine moderne Lösung für Webanwendungen und bietet alle Funktionalitäten die für die Entwicklung der Web-Anwendung benötigt werden.

## 4.2 Entwurf der Benutzeroberfläche

Für die Benutzeroberfläche ist eine intuitive Navigation sowie eine einfache Nutzung der Anwendung notwendig. Hierzu wurden Mockups entwickelt die diese Anforderungen realisieren sollen. Auf Basis der Mockups hat das Team Wireframes herausgearbeitet. Die fertigen Wireframes sind in Anhang c) zu finden.

## 4.3 Entwurf des „Back-end“

Dieses Kapitel beschreibt den Entwurf des Backends für eine WetterApp, die Wetterdaten von der OpenWeatherMap API bezieht und an die Frontend-Benutzeroberfläche weitergibt. Es werden die Architektur des Backends, die Implementierung der Schnittstelle zur OpenWeatherMap API und die wichtigsten technischen Details und Best Practices geplant.

Das Backend der WetterApp wird als RESTful API entwickelt. Diese Architektur ermöglicht es, verschiedene Endpunkte bereitzustellen, die spezifische Wetterdaten abrufen und an das Frontend weiterleiten. Zu den wichtigsten Komponenten der Backend-Architektur zählt der Webserver, welcher zur Verarbeitung von HTTP-Anfragen und Antworten dient. Des weiteren wird eine API-Schnittstelle benötigt, die zwischen App und Open-Weather-API vermittelt. Außerdem wird eine Datenbank benötigt zum speichern der anfallenden Datenmengen. Nicht zu Vergessen ist auch die Middleware ein wichtiger Bestandteil, welcher eingehende Anfragen, Authentifizierung und Validierung bearbeitet

## 4.4 Datenbank

Die Datenbank für die Wetter-App wurde mit MySQL, einem weit verbreiteten relationalen Datenbankmanagementsystem, implementiert. Die Wahl von MySQL bietet mehrere Vorteile, darunter hohe Leistung, Zuverlässigkeit und eine große Community-Unterstützung. Die Datenbank enthält eine zentrale Tabelle, weatherlogs, in der alle relevanten Wetterdaten gespeichert werden.

Die Tabelle weatherlogs ist das Herzstück der Datenbank und speichert die folgenden Informationen:

**id:** Eine eindeutige, automatisch inkrementierte ID, die als Primärschlüssel dient.

**city:** Der Name der Stadt, für die die Wetterdaten erhoben wurden.

**temperature:** Die gemessene Temperatur in Grad Celsius.

**humidity:** Die relative Luftfeuchtigkeit in Prozent.

**description:** Eine Beschreibung des aktuellen Wetterzustands (z.B. "klarer Himmel").

**windspeed:** Die Windgeschwindigkeit in Metern pro Sekunde.

**timestamp:** Der Zeitstempel, der automatisch das Datum und die Uhrzeit der Datenerfassung speichert.

## 4.5 Pflichtenheft

Anhand der Entwürfe wurde am Ende der Entwurfsphase ein Pflichtenheft erstellt. Hierbei wird die

konkrete Umsetzung der im Abschnitt 3.2 (Lastenheft) ermittelten Anforderungen erfasst. Hiermit

kann am Ende des Projektes überprüft werden, ob alle Anforderungen an die Anwendung abgedeckt

und ob diese auch wie vereinbart umgesetzt wurden. Ein Auszug aus dem Pflichtenheft ist im An-

hang d): Pflichtenheft (Auszug) zu finden.

# Implementierungsphase

Anhand des erstellten Pflichtenheftes konnten die Autoren mit der Implementierung des Projektes beginnen.

## 5.1 Implementierung der Benutzeroberfläche

Die HTML-Datei bildet das Grundgerüst unserer Benutzeroberfläche. Sie definiert die Struktur der verschiedenen Elemente, wie z.B. Header, Menüleiste, Wetterinformationen und Benachrichtigungsfelder. Durch die Verwendung von semantischem HTML gewährleisten wir eine klare und verständliche Struktur, die sowohl von Menschen als auch von Suchmaschinen leicht interpretiert werden kann.

Das CSS-Stylesheet definiert das visuelle Erscheinungsbild unserer Benutzeroberfläche. Es enthält Regeln für die Gestaltung von Text, Farben, Layouts und Animationen. Durch die Verwendung von CSS-Klassen und Selektoren können wir das Aussehen der verschiedenen Elemente steuern und eine konsistente Benutzererfahrung sicherstellen.

Um sicherzustellen, dass unsere Wetter-App auf verschiedenen Bildschirmgrößen und Geräten gut funktioniert, haben wir ein responsives Design implementiert. Dies wird durch die Verwendung von Media Queries und flexiblen Layouts erreicht, die es der Benutzeroberfläche ermöglichen, sich dynamisch an die Bildschirmgröße anzupassen.

Zur Anzeige von Wetterinformationen integrieren wir JavaScript, um Wetterdaten von einer externen API abzurufen und dynamisch in die Benutzeroberfläche einzubinden. Durch die Verwendung von AJAX-Anfragen können wir die Wetterdaten aktualisieren, ohne die gesamte Seite neu zu laden, was zu einer reaktionsschnellen Benutzererfahrung führt.

Insgesamt haben wir eine benutzerfreundliche und ansprechende Benutzeroberfläche für unsere Wetter-App implementiert, die es den Benutzern ermöglicht, einfach auf aktuelle Wetterinformationen zuzugreifen und ihre täglichen Aktivitäten entsprechend zu planen.

## 5.2 Implementierung des Backend

Das Backend bildet das Rückgrat unserer Web-Anwendung und ist verantwortlich für die Verarbeitung von Anfragen, die Datenverarbeitung und die Interaktion mit externen Diensten wie Wetter-APIs. Für unsere Wetter-App haben wir Express.js verwendet, um ein robustes und effizientes Backend zu implementieren.

Express.js ist ein leistungsstarkes Node.js-Framework, das sich ideal für die Entwicklung von Web-Anwendungen eignet. Es bietet eine einfache und flexible Möglichkeit, einen HTTP-Server zu erstellen und Routen für verschiedene Endpunkte zu definieren. Durch die Verwendung von Express.js können wir eine skalierbare und leicht erweiterbare Backend-Architektur aufbauen.

Die Routen bilden die Schnittstelle zwischen dem Client und dem Server und definieren, wie Anfragen verarbeitet werden. Wir definieren Routen für verschiedene Funktionen unserer Wetter-App, wie z.B. das Abrufen von aktuellen Wetterdaten, das Anzeigen von Wettervorhersagen und das Verwalten von Benachrichtigungen.

# Fazit

Zum Abschluss des Projektes ziehen die Autoren ein Fazit über das Gelernte und geben einen Ausblick auf die Zukunft des Projektes.

Express.js bietet Middleware-Funktionen, die wir verwenden können, um zusätzliche Funktionalität hinzuzufügen, wie z.B. Authentifizierung, Fehlerbehandlung und Protokollierung. Wir können eigene Middleware erstellen oder auf vorgefertigte Middleware-Bibliotheken zurückgreifen, um unsere Anwendung zu erweitern.

Um Wetterdaten von externen Quellen abzurufen, integrieren wir externe APIs in unser Backend. Dies ermöglicht es uns, genaue und aktuelle Wetterinformationen bereitzustellen. Durch die Verwendung von Express.js können wir HTTP-Anfragen an externe APIs senden und die erhaltenen Daten verarbeiten und an den Client zurücksenden.

## 6.1 Soll-/Ist-Vergleich

Bei der rückblickenden Betrachtung des Projektes kann festgestellt werden, dass alle vorher definierten Anforderungen gemäß Pflichtenheft erfüllt wurden. Der zu Beginn des Projektes erstellte Projektplan konnte eingehalten werden.

## 6.2 Lessons Learned

Im Zuge des Projektes konnten die Autoren viele Erfahrungen über die Arbeit an einem vollständigen

Projekt sammeln. Hierbei wurde deutlich, dass großer Wert auf die Analyse und den Entwurf eines

Projektes gelegt werden muss, die Kommunikation mit dem Team dann jedoch nicht abbrechen

darf. Es ist von großem Vorteil, stetiges Feedback zu bekommen und sich ändernde Anforderungen

schnell zu identifizieren und umzusetzen. Durch das Projekt wurden auch fachliche Kompetenzen

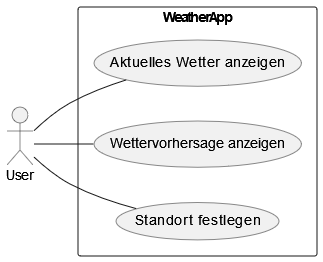
erworben. So konnten die Autoren viele Erkenntnisse zur Erstellung von API-Schnittstellen und Web-Entwicklung erwerben.

## 6.3 Ausblick

Für das Projektteam ist es eine große Erkenntnis, welchen Nutzen man aus einer API-Schnittstelle ziehen kann. Daher ist es denkbar, dass in Zukunft weitere Projekte, welche eine APIs umfassen, entstehen, um den Nutzen dieser in andere Bereiche der Entwicklung einfließen zu lassen.

# Anhang

1. **Use-Case-Diagramm**



1. **Lastenheft (Auszug)**

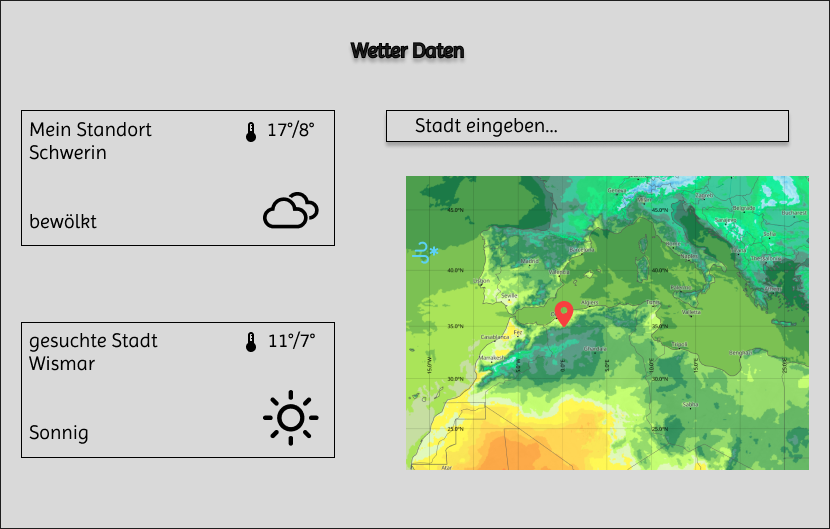
Im folgenden Auszug aus dem Lastenheft werden die Anforderungen definiert, die die zu entwickelnde Anwendung erfüllen muss. Betrachtet wird die Anwendung aus Sicht der im Anhang a): Use-Case-Diagramm ermittelten Akteure.

Anforderungen

Es werden folgende Anforderungen an die Anwendung gestellt:

* Als Benutzer muss ich einen Standort angeben können
* Als Benutzer muss ich das Wetter für den angegebenen Standort einsehen können
* Als Benutzer muss ich eine Wettervorhersage einsehen können für einen Zeitraum von 24 Stunden

1. **Wireframe**



1. **Pflichtenheft (Auszug)**

Das Pflichtenheft beschreibt detailliert die erforderlichen Funktionen, Leistungen, und Qualitäten der Wetter-App. Es dient als Grundlage für die Entwicklung und Implementierung der Anwendung. Die App muss dem Benutzer ermöglichen, das aktuelle Wetter für einen bestimmten Standort abzurufen. Die Anzeige des aktuellen Wetters sollte detaillierte Informationen wie Temperatur und Wetterbedingungen enthalten. Die Wetterdaten müssen zuverlässig und genau sein. Die Vorhersagedaten müssen regelmäßig aktualisiert werden, um Genauigkeit zu gewährleisten. Die App muss dem Benutzer die Möglichkeit geben, seinen aktuellen Standort automatisch zu erkennen. Die Standorterkennung sollte schnell und präzise sein Die Benutzeroberfläche der App muss intuitiv und benutzerfreundlich gestaltet sein, um eine einfache Navigation zu ermöglichen. Die App sollte responsiv sein und auf verschiedenen Bildschirmgrößen und Gerätetypen gut funktionieren. Die Benutzeroberfläche muss ästhetisch ansprechend gestaltet sein. Die App muss zuverlässig sein und eine hohe Verfügbarkeit der Wetterdaten sicherstellen. Die App sollte robust sein und angemessen auf Fehler oder Ausfälle reagieren. Die Datensicherheit und der Datenschutz müssen gewährleistet sein. Die App wird für eine Vielzahl an Browsern entwickelt und sollte auf den neuesten Versionen dieser Plattformen gut funktionieren. Die App muss eine sichere Datenübertragung gewährleisten und Datenschutzrichtlinien einhalten. Die App muss mit verschiedenen Bildschirmgrößen und Auflösungen kompatibel sein. Die Genauigkeit der Wettervorhersagen kann je nach Wetterbedingungen und Datenquellen variieren.