Dokumentation zur Anwendungsentwicklung eines Wetterdaten-Analysetools

Dokumentation Projekt mit Anwendungsentwicklung

Studienjahrgang 2022

Kurs C

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

DUALE HOCHSCHULE BADEN-WÜRTTEMBERG

VILLINGEN-SCHWENNINGEN

Bearbeiter:

N. Diehl, S. Dubiel,

Q. Großhauser, A. Iacona, M. Paffen

Betreuender Dozent:

Prof. Dr. Wolfang Funk

Inhaltsverzeichnis

Inhaltsverzeichnis III
1 Projektdefinition 1
1.1 Projektsteckbrief 1
1.2 Projektziel 1
1.3 Rahmenbedingungen 1
2 Projektorganisation 2
2.1 Stakeholder 2
2.2 Verantwortlichkeiten 2
2.3 Kommunikationsplan 3
3 Projektplanung 3
3.1 Projektmeilensteine 3
3.2 Vorgangsliste 4
3.3 Netzplan 4
3.4 Risikomanagement 5
3.5 Qualitätssicherung 9
4 Projektdurchführung 13
4.1 Funktionale Anforderungen 13
4.2 Nicht-funktionale Anforderungen 16
4.3 Use-Case „Wetterdaten analysieren“ 18
4.4 Handbuch 21
4.4.1 Grafische Benutzeroberfläche (Mockup) 22
4.4.2 Architektur 22
4.4.3 Bedienungsanleitung 25
4.5 Test der Anwendung 29
4.6 4.6 Eingesetzte Werkzeuge 31
5 Projektabschluss 32
5.1 Projektabschlussbericht 32
5.2 Erfahrungsbericht 33
6 Anhang 33
7 Selbstständigkeitserklärung 37

1 Projektdefinition

1.1 Projektsteckbrief

Zum Start des Projektes liefert der Projektsteckbrief eine kompakte Übersicht der Projektdaten und dient allen Beteiligten als zentrale Informationsquelle.

Projektname	Anwendungsentwicklung eines Wetterdaten-Analysetools
Auftragnehmer	Gruppe 1
Projektteam	Nikolas Diehl (ND), Simon Dubiel (SD), Quirin Großhauser (QG), Angelo Iacona (AI), Malte Paffen (MP)
Projektleiter	Malte Paffen
Auftraggeber	Prof. Dr.-Ing. Wolfgang Funk
Projektzeitraum	08.01.2025 – 14.03.2025

1.2 Projektziel

Ziel des Projektes ist die Entwicklung einer Softwareanwendung zur Analyse historischer Wetterdaten auf Basis des Daily Global Historical Climatology Network (GHCN). Die Anwendung soll dabei eine benutzerfreundliche grafische Oberfläche bieten, um Messdaten von Wetterstationen zu durchsuchen, zu analysieren und zu visualisieren.

1.3 Rahmenbedingungen

Die folgenden Rahmenbedingungen definieren die zentralen Vorgaben und Einschränkungen des Projektes und legen somit den Projektrahmen fest.

Datenbasis: GHCN
Anwendung: Lauffähig unter Windows 11 (optional WSL)
Architektur: Client-Server-Modell; Server wird in Containern bereitgestellt
Technologien: Python inkl. Flask (Backend), JavaScript und HTML (Frontend)
Systembereitstellung: Docker Containerisierung

2 Projektorganisation

2.1 Stakeholder

Die Stakeholder des Projektes sind alle Personen oder Gruppen, die ein Interesse am Projekt haben oder direkt betroffen sind. Die folgende Tabelle umfasst alle Stakeholder inklusive ihrer Rolle/Funktion innerhalb des Projektes. Der Grad der Betroffenheit sowie die Art der Kommunikation sind ebenfalls dargestellt.

Name	Funktion	Betroffenheit	Kommunikation
ND, SD, QG, AI, MP	Projektteam	hoch	Tägliche Abstimmung des Teams
Malte Paffen	Projektleitung	sehr hoch	Wöchentliche Updates
Prof. Dr. Wolfgang Funk	Kunde	sehr hoch	Wöchentlicher Statusbericht durch Projektleitung
Julian Gommlich	Interner Consultant	mittel	Bei Bedarf (wöchentlich)

2.2 Verantwortlichkeiten

Die klare Zuweisung von Verantwortlichkeiten stellt sicher, dass Aufgaben und Zuständigkeiten innerhalb des Projektes eindeutig definiert sind, womit die Zusammenarbeit langfristig gestärkt wird. Die folgende Tabelle gibt Auskunft über die Verantwortlichkeiten der einzelnen Projektbeteiligten.

Nikolas Diehl	Backend-Entwicklung und Entwicklungskoordination
Simon Dubiel	Projektkoordination, Projektdokumentation und Testing
Quirin Großhauser	Dokumentation und Testunterstützung
Angelo Iacona	Design, Frontend-Entwicklung und technische Dokumentation
Malte Paffen	Technische Frontend Entwicklung und Projektmanagement

2.3 Kommunikationsplan

Für die erfolgreiche Durchführung eines Projektes mit mehreren Beteiligten ist die Kommunikation innerhalb des Teams sowie mit externen Stakeholdern ein entscheidender Erfolgsfaktor. Im Rahmen des Projektes wurden die folgenden Kommunikationswege gewählt.

Microsoft Teams

- Ablage und Synchronisation aller Dokumente und Unterlagen
- Regelmäßiger Austausch über aktuellen Status und Next Steps
- Visuelle Darstellungen über die Aufgabenverteilung im Team

E-Mail

- Externe Kommunikation mit dem Kunden
- Interne Kommunikation mit dem Consultant
- Weiterleitung relevanter Informationen an das Projektteam

Meetings

- Besprechung wichtiger Fortschritte und Hindernisse
- Beratungsgespräche mit internem Consultant über Optimierungen
- Projektabnahme durch Kunden innerhalb eines Meetings

Ad-hoc Kommunikation

- Kurzfristige und spontane Absprachen

3 Projektplanung

3.1 Projektmeilensteine

Projektmeilensteine markieren wichtige Etappen im Projektverlauf und dienen zur kontinuierlichen Fortschrittskontrolle. Die folgende Tabelle gibt einen Überblick über die zentralen Meilensteine von Beginn bis Ende des Projektes.

08.01.2025	Projektbeginn
Woche 0-2	Projektinitialisierung
Woche 2-4	Basis Setup & Prototyping
Woche 4-6	Funktionale Integration
Woche 6-8	Datenvisualisierung
Woche 8-9	Finalisierung und Deployment
17.03.2025	Projektende

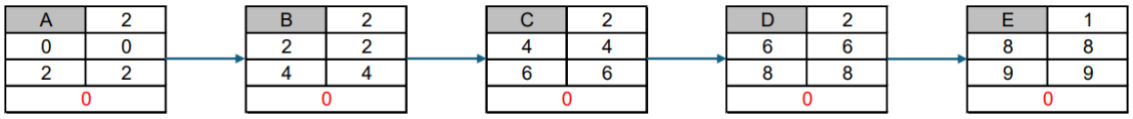
3.2 Vorgangsliste

Die zuvor definierten Meilensteine werden in eine Vorgangsliste überführt, die somit alle notwendigen Aufgaben enthält und für eine strukturierte Planung sorgt. Die folgende Tabelle zeigt die Vorgangsliste, die in Anbetracht des Umfangs des Projektes sehr verständlich und trivial dargestellt ist.

Tätigkeit	Beschreibung	Dauer in Wochen	Vorgänger
A	Projektinitialisierung	2	-
B	Basis Setup & Prototyping	2	A
C	Funktionale Integration	2	B
D	Datenvisualisierung	2	C
E	Finalisierung und Deployment	1	D

3.3 Netzplan

Die Vorgangsliste mit den zentralen Tätigkeiten des Projektes kann in einen Netzplan überführt werden, um den Ablauf der Aktivitäten zu visualisieren. Der folgende Netzplan impliziert, dass aufgrund des engen zeitlichen Projektrahmens, keine Aktivität einen möglichen Puffer erlaubt.



3.4 Risikomanagement

Ein effektives Risikomanagement identifiziert potenzielle Risiken frühzeitig und entwickelt angemessene Maßnahmen, um diesen entgegenzuwirken. Zunächst werden innerhalb einer Risikoübersicht potenzielle Risiken identifiziert, die den Projektverlauf maßgeblich beeinfluss können.

Risikoübersicht

Risikokategorie	Risiko-ID	Beschreibung
Organisation	R1	Verzögerungen bei der Kommunikation mit dem Kunden (MP).
	R2	Missverständnisse oder Unklarheiten bei den Anforderungen.
Team & Ressourcen	R3	Ausfall eines Schlüsselmitglieds (z. B. Krankheit von SD oder ND).
	R4	Überlastung einzelner Teammitglieder aufgrund enger Zeitplanung.
	R5	Koordinationsprobleme im Team (z. B. unklare Verantwortlichkeiten).
Technik (Frontend)	R6	Technische Schwierigkeiten bei der Implementierung der Weltkugel (Three.js).
	R7	Probleme bei der Integration von Backend-Daten ins Frontend.
Technik (Backend)	R8	Verzögerungen bei der Integration der NOAA-API (ND).
	R9	Probleme bei der Datenaufbereitung und Optimierung.
	R10	Unzureichende Testabdeckung im Backend aufgrund des verkürzten Zeitplans.
Technik (CI/CD)	R11	Fehlerhafte CI/CD-Konfiguration oder Docker-Setup.
Dokumentation	R12	Fehlende oder fehlerhafte Dokumentation aufgrund von Zeitdruck.
Deployment	R13	Probleme bei der Veröffentlichung des Docker-Images auf der GitHub Container Registry.

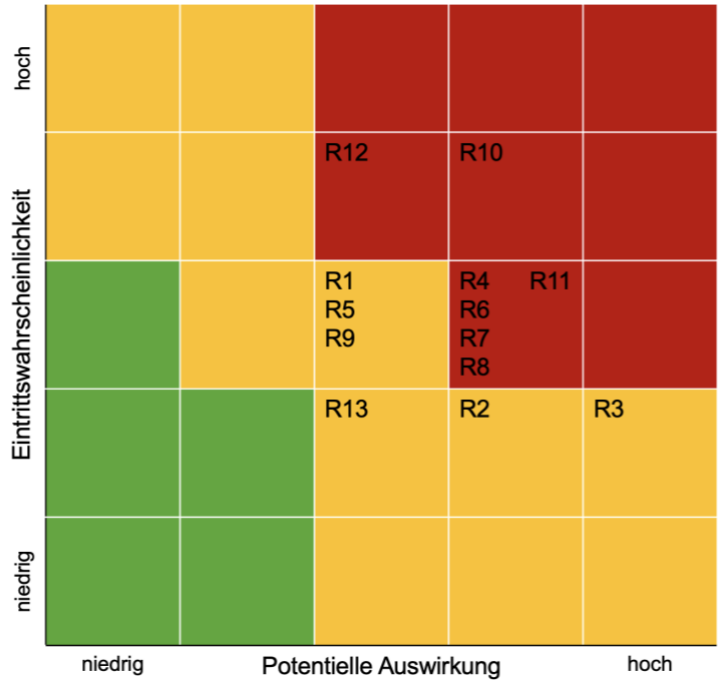
Nach der Identifikation werden die Risiken hinsichtlich ihrer Eintrittswahrscheinlichkeit und potenzieller Auswirkung bewertet. Das Risiko-Level ergibt sich aus der Multiplikation dieser Faktoren. Somit liefert die folgende Risiko-Level-Tabelle die Basis für die Entwicklung von Maßnahmen zur Risikominimierung.

Risiko-Level

ID	Eintrittswahrscheinlichkeit	Potenzielle Auswirkung	Risiko-Level
R1	3	3	9 (Mittel)
R2	2	4	8 (Mittel)
R3	2	5	10 (Hoch)
R4	3	4	12 (Hoch)
R5	3	3	9 (Mittel)
R6	3	4	12 (Hoch)
R7	3	4	12 (Hoch)
R8	3	4	12 (Hoch)
R9	3	3	9 (Mittel)
R10	4	4	16 (Kritisch)
R11	3	4	12 (Hoch)
R12	4	3	12 (Hoch)
R13	2	3	6 (Mittel)

Die Einschätzung über das Risiko-Level der einzelnen Risiken ermöglicht es, die Risiken visuell in einer Risikomatrix darzustellen.

Risikomatrix



Risikomanagementplan

Der Risikomanagementplan sieht Maßnahmen für den Umgang mit den identifizierten Risiken für den Projektverlauf vor. Risikobewältigende Maßnahmen verringern die Eintrittswahrscheinlichkeiten bzw. Auswirkungen oder vermeiden kritische und hohe Risiken. Überwachungsmaßnahmen stellen die regelmäßige Prüfung des Status mittlerer Risiken sicher. Die Beschreibung des Umgangs mit niedrigen Risiken entfällt mangels Risiken mit entsprechendem Risiko-Level. Die Verantwortlichkeiten für einzelne Risiken sind im Risikomanagementplan aufgeführt.

Maßnahmen für kritische und hohe Risiken (Risiko-Level >= 12)

ID	Maßnahme	Ziel	Verantwortlich	Frist
R4	Aufgabenverteilung überprüfen und Zeitpuffer für kritische Aufgaben einplanen.	Überlastung vermeiden	MP	Laufend

R6	Proof-of-Concepts für kritische Three.js-Funktionen frühzeitig umsetzen.	Technische Risiken minimieren	AI	Woche 2
R7	Tests für Schnittstellen zwischen Frontend und Backend planen und frühzeitig durchführen.	Integration sicherstellen	AI, SD, ND	Woche 4
R8	API-Integration priorisieren und mit Mock-Daten arbeiten, um frühzeitig Daten aufzubereiten.	Verzögerungen vermeiden	ND	Woche 3
R10	Teststrategien für kritische Backend-Funktionen priorisieren und regelmäßig prüfen.	Qualität des Backends sicherstellen	SD, ND	Woche 5
R11	Manuelle Tests der CI/CD-Pipeline vor Automatisierung durchführen, um Fehler zu vermeiden.	Fehler in der Pipeline beheben	MP, SD	Woche 6
R12	Zwischenstände der Dokumentation regelmäßig überprüfen und Feedback einholen.	Dokumentationsqualität sicherstellen	QG	Laufend

Maßnahmen für mittlere Risiken (Risiko-Level 6-11)

ID	Maßnahme	Verantwortlich	Frequenz

R1	Regelmäßige Rückmeldungen an das Team über den Status der Kundenkommunikation.	MP	Wöchentlich
R2	Anforderungen schriftlich festhalten und frühzeitig vom Kunden bestätigen lassen.	MP	Vor jedem Sprint
R3	Dokumentation und Übergabepläne erstellen, um Wissensverlust bei Krankheit zu vermeiden.	Alle	Laufend
R5	Regelmäßige Abstimmungen zwischen den Gruppenmitgliedern sichern das rechtzeitige Erkennen von Missverständnissen und Abstimmungsbedarf.	MP bzw. alle	Mindestens wöchentlich
R9	Aufgaben mit geringerer Priorität verschieben, um Zeit für Tests zu gewinnen.	SD, ND	Nach jedem Sprint
R13	Parallelisierung der Veröffentlichung mit Testumgebungen sicherstellen.	MP, SD	Vor Veröffentlichung

Niedrige Risiken (Risiko-Level <= 5)

Risiken mit einem Risiko-Level kleiner oder gleich 5 werden im Projektverlauf zunächst akzeptiert. Die Auswirkungen werden hingenommen. Da keines der 13 ermittelten Risiken ein Risiko-Level kleiner oder gleich 5 hat sind dementsprechend Maßnahmen für niedrige Risiken zu vernachlässigen.

3.5 Qualitätssicherung

Die Qualitätssicherung stellt sicher, dass das Projekt die funktionalen und nicht-funktionalen Anforderungen vollständig erfüllt. Es gewährleistet die besondere Qualität der Dokumentation und Kommunikation im Projektverlauf sowie vor der Übergabe an den Kunden. Das Qualitätsmanagement zielt auf die reibungslose

Übergabe des fertigen Produkts an den Kunden. Dazu stellt es die Qualität der Lieferobjekte sowie die Testabdeckung in geforderter Höhe sicher.

Qualitätsziele

Bereich	Qualitätsziel
Funktionale Anforderungen	Alle definierten funktionalen Anforderungen (ID 0.x) werden vollständig und korrekt implementiert.
Nicht-funktionale Anforderungen	Mindestens 80 % Testabdeckung, Einhaltung von Ressourcenlimits (Docker: 2 CPUs, 1 GB RAM).
Dokumentation	Vollständige, konsistente und DIN-konforme Projektdokumentation (DIN 69901).
Codequalität	Sauber strukturierter, dokumentierter und durch Tests validierter Code.
Deployment	Stabiles CI/CD mit GitHub Actions und erfolgreich veröffentlichte Docker-Images.

Qualitätsmanagementprozesse

1. Qualitätsplanung
    - Ziel: Definition von Qualitätsanforderungen und Standards für das Projekt.
    - Maßnahmen:
        - Festlegung der Anforderungen anhand der funktionalen und nicht-funktionalen Anforderungen (ID 0.x und 1.x).
        - Entwicklung eines Testplans für Unit-, Service- & Integrationstests.
        - Definition von Akzeptanzkriterien für jedes Arbeitspaket.

2. Qualitätssicherung
    - Ziel: Laufende Überprüfung, ob die Qualitätsziele eingehalten werden.
    - Maßnahmen:
        - Code-Reviews durch Teammitglieder (SD und ND überprüfen gegenseitig).
        - Wöchentliche Statusmeetings zur Besprechung von Fortschritten und Qualitätsproblemen (MP moderiert).

- Automatisierte Tests (GitHub Actions) zur Sicherstellung der Funktionalität nach jedem Push.

3. Qualitätskontrolle

- Ziel: Überprüfung der Arbeitsergebnisse, um sicherzustellen, dass sie den Anforderungen entsprechen.
- Maßnahmen:
  - Abnahmeprotokolle für abgeschlossene Arbeitspakete (z.B. API-Endpunkte, Frontend-Visualisierungen).
  - Vergleich der Testergebnisse mit den Akzeptanzkriterien.
  - Dokumentationsprüfung durch QG vor Meilenstein- und Projektabschluss.

Qualitätsprüfmethoden

Methode	Beschreibung	Verantwortlich
Code-Review	Überprüfung des Codes auf Lesbarkeit, Einhaltung der Standards und potenzielle Fehler.	SD, ND
Automatisierte Tests	Durchführung von Unit-, Service- und Integrationstests über GitHub Actions.	SD, ND
Manuelle Tests	Testen der Anwendung im Frontend (Three.js-Weltkugel, Tabellen, Diagramme).	AI
Dokumentationsprüfung	Überprüfung der Vollständigkeit und Konsistenz der Dokumentation anhand einer Checkliste.	QG
Review-Meetings	Gemeinsame Besprechung der Ergebnisse und Qualitätsfragen im Team.	MP, alle

QM-Rollen und -Verantwortlichkeiten

Rolle	Verantwortung
MP	Überwachung der Qualitätssicherung und Moderation von Review-Meetings.
QG	Sicherstellung der Dokumentationsqualität und Konsistenz.

SD & ND	Umsetzung der Backend-Qualität (API, Tests) und gegenseitige Code-Reviews.
AI	Sicherstellung der Frontend-Qualität (Three.js, D3.js) und Tests der Visualisierungen.

Qualitätsprüfplan

Meilenstein	Qualitätsmaßnahme	Verantwortlich	Frist
Projekt-initialisierung	Festlegung der Qualitätsziele und Erstellung des Testplans.	MP, QG	Woche 2
Basis Setup & Prototyping	Überprüfung der grundlegenden API- und Frontend-Funktionen durch automatisierte und manuelle Tests.	SD, ND, AI	Woche 4
Funktionale Integration	Durchführung von End-to-End-Tests und Sicherstellung der Backend-Frontend-Kommunikation.	SD, ND, AI	Woche 6
Daten-visualisierung	Überprüfung der korrekten Darstellung und Performance der Visualisierungen durch Tests und Feedback.	SD, ND	Woche 8
Finalisierung & Deployment	Durchführung einer Dokumentationsprüfung und Abnahme des gesamten Systems.	QG, MP	Woche 9

QM-Werkzeuge und -Hilfsmittel

Werkzeug	Zweck
GitHub Actions	Automatisierung der Tests und CI/CD-Prozesse.
Docker	Sicherstellung einer einheitlichen Laufzeitumgebung.
pytest	Durchführung von Unit- und Integrationstests.
ESLint/Prettier	Code-Qualitätsprüfung für das Frontend.

Überwachung und Bericht

Qualitätsmetriken:

- Testabdeckung (≥80 %).
- Anzahl der offenen und geschlossenen Fehler (Bugs).
- Anzahl der durchgeführten Code-Reviews.

Berichtswesen:

- Wöchentliche Statusberichte durch MP und QG.
- Abschlussbericht zur Einhaltung der Qualitätsziele am Projektende

4 Projektdurchführung

4.1 Funktionale Anforderungen

Die folgenden Anforderungen beschreiben zentrale Funktionen des Systems.

Anforderung 0.1: Eingabe von Koordinaten und Radius

Schlüsselwort	Information
ID	0.1
Art	Funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Nutzer sollen in der Lage sein, geografische Koordinaten (Breite, Länge) und einen Radius einzugeben, um damit eine Eingrenzung der relevanten Wetterstationen vorzunehmen.
Beschreibung und Kontext	Die Anwendung erfasst die eingegebenen Parameter und leitet diese an das Backend weiter, wo eine Filterung der Daten vorgenommen wird.
Eingaben und Quellen	Nutzereingaben (Koordinaten, Radius).
Ausgaben und Ziele	Validierte Weitergabe der Eingaben an das Backend zur Verarbeitung.
Begründung	Diese Eingabeparameter bilden die Grundlage für die Datenverarbeitung und Filterung relevanter Wetterstationen.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

Anforderung 0.2: Filterung von Wetterstationen

Schlüsselwort	Information
ID	0.2
Art	Funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Basierend auf den Eingaben des Nutzers sollen Wetterstationen innerhalb des spezifizierten Radius identifiziert werden.
Beschreibung und Kontext	Das Backend interagiert mit der NOAA-API, um Wetterstationsdaten zu beziehen, und nutzt Algorithmen wie die Haversine-Formel zur Distanzberechnung, um Stationen im spezifizierten Radius zu ermitteln. Die Verfügbarkeit von Wetterdaten im gewünschten Zeitraum ist anhand des sog Inventorys zu ermitteln.
Eingaben und Quellen	Nutzerangaben, NOAA-API („/stations“-Endpunkt).
Ausgaben und Ziele	Eine Liste relevanter Wetterstationen mit ID, Name und Koordinaten.
Begründung	Diese Funktion dient der Identifikation von Wetterstationen, die auf Grundlage geografischer Daten für den Nutzer relevant sind.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

Anforderung 0.3: Auswahl einer Wetterstation

Schlüsselwort	Information
ID	0.3
Art	Funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Nutzende sollen eine Wetterstation aus der zuvor generierten Liste auswählen können.
Beschreibung und Kontext	Die ausgewählte Station wird genutzt, um weitere Datenabfragen durchzuführen und diese im Kontext historischer Wetterdaten aufzubereiten.
Eingaben und Quellen	Nutzerauswahl (Stations-ID).
Ausgaben und Ziele	Weiterleitung der Stations-ID an das Backend.
Begründung	Diese Auswahl erlaubt eine gezielte Datenanalyse für eine spezifische Station.

Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

Anforderung 0.4: Abfrage von historischen Wetterdaten

Schlüsselwort	Information
ID	0.4
Art	Funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Das Backend muss historische Wetterdaten der ausgewählten Wetterstation über die NOAA-API abrufen und diese Daten aggregieren.
Beschreibung und Kontext	Die abgerufenen Rohdaten werden verarbeitet, um Jahresmittelwerte von Temperaturminima und -maxima zu berechnen.
Eingaben und Quellen	Stations-ID, NOAA-API („/data“-Endpunkt).
Ausgaben und Ziele	Strukturierte Daten, die Jahresmittelwerte für die Nutzer visualisieren.
Begründung	Ermöglicht Nutzenden die Analyse von historischen Wetterdaten auf Basis einer spezifischen Station.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

Anforderung 0.5: Darstellung der Wetterdaten

Schlüsselwort	Information
ID	0.5
Art	Funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Historische Wetterdaten sollen sowohl tabellarisch als auch grafisch dargestellt werden.
Beschreibung und Kontext	Die Datenvisualisierung ermöglicht es Nutzenden, Trends und historische Entwicklungen in Temperaturminima und -maxima intuitiv zu erfassen. Es werden Temperaturmaxima sowie -minima im gewählten Zeitraum jährlich sowie je Jahreszeit ausgegeben. Die Jahreszeiten sind für Standorte auf der Nordhalbkugel wie folgt anzunehmen (je +

	sechs Monate für Standorte auf der Südhalbkugel): Winter – November des Vorjahres bis Januar, Frühling – Februar bis April, Sommer – Mai bis Juli, Herbst – August bis Oktober.
Eingaben und Quellen	Daten vom Backend
Ausgaben und Ziele	Diagramme und Tabellen zur Visualisierung
Begründung	Eine visuelle Darstellung fördert das Verständnis und die Analyse der historischen Wettertrends.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

4.2 Nicht-funktionale Anforderungen

Die folgenden Anforderungen legen die Qualitätsmerkmale des Systems fest.

Anforderung 1.1: Testautomatisierung und Testabdeckung

Schlüsselwort	Information
ID	1.1
Art	Nicht-funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Es wird eine Testabdeckung von mindestens 80% gefordert. Zusätzliche entspricht der Code den Anforderungen des sog. Clean Codes.
Beschreibung und Kontext	Durch die Implementierung von Tests wird sichergestellt, dass die Funktionalität der Anwendung stabil bleibt.
Eingaben und Quellen	Quellcode, Testframeworks (z. B. pytest)
Ausgaben und Ziele	Automatisierte Berichte über Testergebnisse
Begründung	Eine hohe Testabdeckung reduziert die Wahrscheinlichkeit von Fehlern und verbessert die Qualität des Codes.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

Anforderung 1.2: Nutzung von Docker zur Containerisierung

Schlüsselwort	Information
ID	1.2
Art	Nicht-funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Die Anwendung wird in Docker Containern zur Verfügung gestellt.
Beschreibung und Kontext	Die Containerisierung stellt sicher, dass die Anwendung konsistent auf verschiedenen Umgebungen läuft und einfach zu deployen ist.
Eingaben und Quellen	Dockerfile, Docker Compose
Ausgaben und Ziele	Lauffähige Docker-Container
Begründung	Ermöglicht eine einfache Bereitstellung und Wartung der Anwendung.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

Anforderung 1.3: Nutzung von GitHub und CI/CD-Pipeline

Schlüsselwort	Information
ID	1.3
Art	Nicht-funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	GitHub wird als Versionskontrollsystem verwendet. Eine automatisierte CI/CD-Pipeline stellt sicher, dass Tests und Builds bei jedem Push ausgeführt werden.
Beschreibung und Kontext	GitHub Actions wird genutzt, um den Entwicklungsprozess zu automatisieren und die Codequalität kontinuierlich zu überprüfen.
Eingaben und Quellen	GitHub, GitHub Actions
Ausgaben und Ziele	Automatisierte Builds und Testergebnisse
Begründung	Die Automatisierung verbessert die Effizienz und Qualitätssicherung des Entwicklungsprozesses.
Klassifikation	Soll
Priorität	Mittel
Konflikte	Keine

Anforderung 1.4: Ressourcenbegrenzung in Containern

Schlüsselwort	Information
ID	1.4
Art	Nicht-funktionale Anforderung
Anforderungsträger	Auftraggeber
Anforderung	Jeder Docker-Container ist so zu konfigurieren, dass er maximal 2 CPUs und 1 GB RAM verwendet.
Beschreibung und Kontext	Die Begrenzung der Ressourcen stellt sicher, dass die Anwendung ressourcenschonend arbeitet und die Stabilität der Betriebsumgebung nicht beeinträchtigt.
Eingaben und Quellen	Docker Compose-Konfiguration
Ausgaben und Ziele	Container, die die spezifizierten Ressourcenbegrenzungen einhalten.
Begründung	Stellt sicher, dass die Systemressourcen effizient genutzt werden.
Klassifikation	Muss
Priorität	Hoch
Konflikte	Keine

4.3 Use-Case „Wetterdaten analysieren“

Ein Use-Case (Anwendungsfall) ist eine Ablaufbeschreibung um Anforderungen strukturiert zu erfassen. Er dient als Input für Analyse und Design. Im Folgenden wird der Use-Case des Wetterdaten-Analysetools vorgestellt.

Überblick:

Ziel ist die Abfrage von Wetterdaten und Statistiken mittels einer Web-Anwendung. Ein Benutzer kann nach Eingabe beliebiger Koordinaten sowohl die in der Nähe liegenden Wetterstationen auf einer Karte erkennen, sowie die entsprechenden Wetterdaten der Stationen einsehen.

Geltungsbereich:

Der Anwendungsfall gehört zur Entwicklung einer Software innerhalb des Kurses „Projekt mit Anwendungsentwicklung“ und zielt auf die Auswertung meteorologischer Daten auf Basis von eingegebenen Koordinaten ab.

Anwendungsschicht:

Benutzer

Primärer Akteur:
Benutzer (Endnutzer der Web-Anwendung)

Weitere Akteure:

- System
- Wetterdatenanbieter (GHCN)
- Professor (Kunde)
- Consultant (interner Consultant)

Stakeholder und ihre Belange:

- Wissenschaftler und Meteorologen: zur Erhebung der Wetterdaten
- Umweltforscher: Überwachung klimatischer Veränderungen
- Endnutzer: Abruf aktueller Wetterdaten aus verschiedenen Regionen
- Professor: Möchte als Kunde die Anwendung funktionsfähig erhalten
- Consultant: Arbeitskollege mit Interesse an Erfolg des Projektes

Vorbedingungen:

- Der Benutzer hat Zugriff auf die Anwendung
- Die Wetterdatenbank ist erreichbar und erhält die korrekten Daten
- Der Server ist funktionsfähig und stellt die Daten in richtiger Weise dar

Nachbedingungen:

- Die angeforderten Wetterdaten wurden erfolgreich abgerufen und angezeigt

Hauptzweig:

1. Der Benutzer gibt geografische Koordinaten (Länge und Breite) ein
2. Das System sucht in der GHCN-Datenbank nach passenden

anliegenden Wetterstationen und zeigt diese an
3. Der Benutzer wählt eine der angebotenen Stationen aus
4. Das System ruft die historischen Wetterdaten der gewählten Stationen aus
5. Die Stadt wird mithilfe der Karte angezeigt
6. Die Wetterdaten der Station werden unten in tabellarischer Form angezeigt
7. Der Benutzer kann eine weitere Wetterstation auswählen oder neue Koordination auswählen

Erweiterungen:

1. Es wird keine Station in der Umgebung gefunden: Es werden keine Wetterstationen angezeigt und der Benutzer kann neue Koordinaten eingeben.
2. Der Nutzer kann anstelle von Koordinaten Städtenamen oder Postleitzahlen eingeben und bekommt passende Stationen zur Auswahl gestellt.

Besondere Anforderungen:

- Die Benutzeroberfläche muss intuitiv und responsiv sein
- Die Suchparameter (Radius, Zeitraum) müssen anpassbar sein
- Die Anwendung muss eine effiziente Datenverarbeitung und Darstellung
- Die Anwendung muss containerisiert bereitgestellt werden, um eine einfache Deployment- und Skalierbarkeit zu ermöglichen

Variationen in Bezug auf Technologie und Daten:

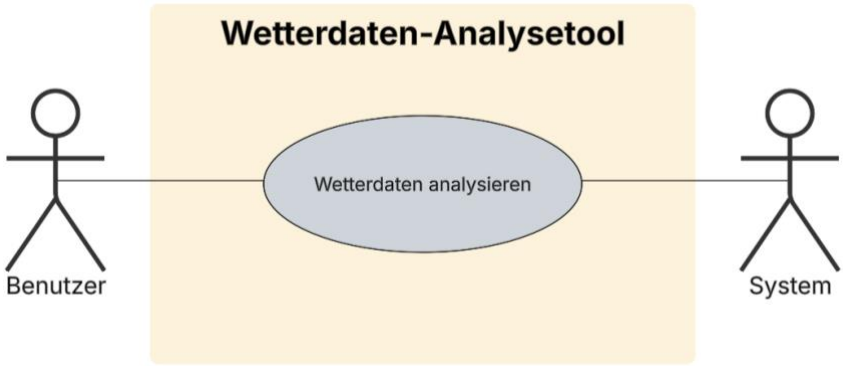
- Die Regionen der Wetterstationen werden mithilfe einer Karte angezeigt

Häufigkeit des Auftretens:

- Erwartet wird eine regelmäßige (tägliche) Nutzung des Benutzers
- Wissenschaftliche Nutzer könnten periodisch größere Datenmengen

abrufen und für weiter Zwecke nutzen

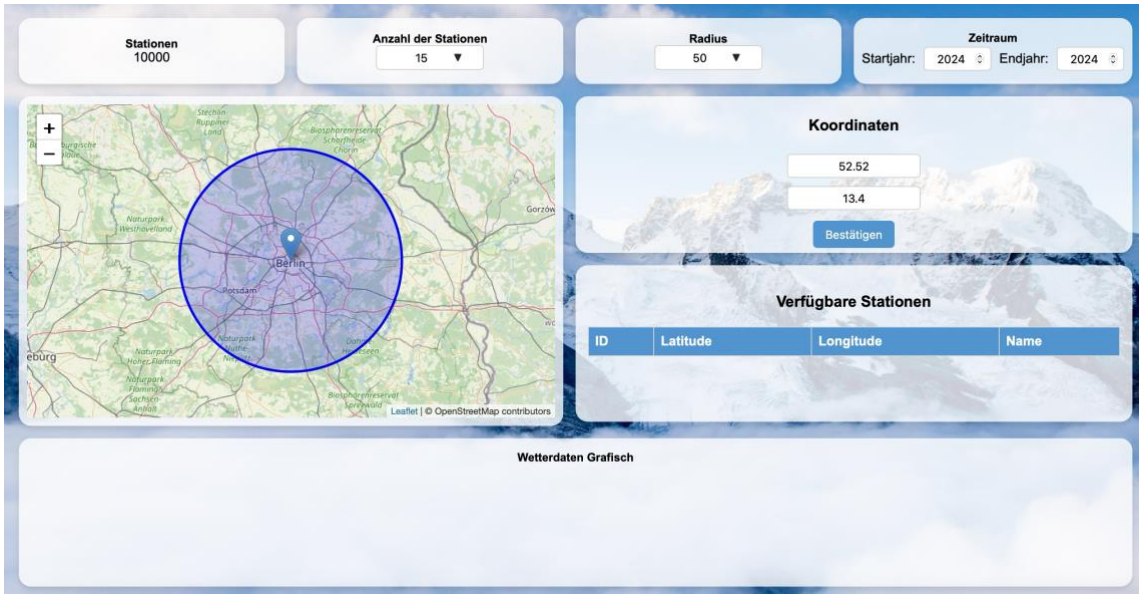
Use-Case Diagramm



4.4 Handbuch

Das Handbuch spielt eine elementare Rolle im Projekt, da es alle wichtigen Aspekte der Anwendung dokumentiert und eine klare Orientierung für Entwickler und Nutzer schafft. Zunächst enthält es ein Mockup der Software, das die Benutzeroberfläche veranschaulicht. Die Beschreibung der Systemarchitektur führt zu einer verständlichen Darlegung der technischen Struktur. Eine ausführliche Bedienungsanleitung erleichtert die Nutzung und ermöglicht eine schnelle Einarbeitung. Abschließend werden mögliche zukünftige Erweiterungen vorgestellt. Die Code-Dokumentation, in der alle Funktionen beschrieben werden, ist als readme-Datei im Github Repository abgelegt.

4.4.1 Grafische Benutzeroberfläche (Mockup)



4.4.2 Architektur

Die Architektur folgt einer Client-Server-Struktur, bestehend aus:

1. Frontend (Client)
2. Backend (Server)
3. Externe API (NOAA API)
4. CI/CD und Containerisierung (GitHub Actions + Docker)

Komponenten und Schichtenarchitektur

Im Folgenden findet sich ein Überblick der wesentlichen funktionalen Bestandteile dieses Produktes. Im Repository sind detaillierte Erläuterungen sämtlicher Bestandteile zu finden, s. „/users_guide“ und „readme.md“.

Die Architektur ist in vier Hauptkomponenten unterteilt:

1. Frontend (Benutzeroberfläche)

- Technologien: HTML, CSS, JavaScript (D3.js), OpenStreetMap.
- Verantwortlich für:
    - Darstellung der Karte mit Wetterstationen.
    - Benutzeroberfläche zur Eingabe von Koordinaten und Radius.
    - Anzeige von Wetterstationsdaten in Tabellen & einem Diagramm.

- Kommunikation mit Backend:
  - Sendet API-Anfragen an das Flask-Backend.
  - Erhält Antworten z.B. in Form von Wetterdaten oder Ladestatus.
- Empfang von Wetterdaten zur Visualisierung.
- Berechnung und Verarbeitung aggregierter Wetterdaten zur Darstellung von Mittelwerten.

2. Backend (Flask API-Server)

- Technologien: Python (Flask), Flask-CORS, Requests, Pandas, io, math und threading.
- Verantwortlich für:
  - Empfang und Verarbeitung der Nutzereingaben.
  - Kommunikation mit der NOAA API, um Wetterstations- und Wetterdaten abzurufen.
  - Datenverarbeitung mit Pandas: Filterung und Formatierung der Wetterstations- und Wetterdaten
  - Bereitstellung von REST-APIs für das Frontend - API-Endpunkte:
    - **/preload_status:** Dieser Endpunkt liefert eine JSON-Antwort, die anzeigt, ob das Vorladen (Preloading) der Stations- und Inventardaten abgeschlossen ist. Er gibt entweder {"status": "loading"} oder {"status": "done"} zurück.
    - **/:** Der Root-Endpunkt rendert das Template index.html und übergibt die Anzahl der geladenen Stationen (station_count) als Kontext.
    - **/get_stations:** Ein GET-Endpunkt, der über Query-Parameter (wie latitude, longitude, radius_km, station_count, start_year und end_year) die Stationen räumlich sowie anhand des Inventory-Vezeichnisses filtert. Er liefert eine JSON-Liste der Stationen, die den Kriterien entsprechen.

        - **/ get_weather_data:** Ein GET-Endpunkt, der die Wetterdaten einer bestimmten Station abruft. Er erwartet Parameter wie station_id, start_year und end_year und gibt die gefilterten Wetterdaten im JSON-Format zurück.
        - **/error** (nur verfügbar, wenn app.config.get("TESTING") wahr ist): Ein Test-Endpunkt, der absichtlich eine Ausnahme auslöst, um einen Fehler zu simulieren.
- Datenfluss:
    - Nutzereingaben → Backend verarbeitet Koordinaten → NOAA API Anfrage → Aufbereitung → Rückgabe ans Frontend.

3. Externe API (NOAA National Climate Data Center API)
    - Basis-URL: https://www.ncei.noaa.gov/cdo-web/api/v2/
    - Genutzte API-Endpunkte:
        - ghcnd-stations.txt
            - URL: https://www.ncei.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt /data
            - Liefert Wetterdaten für eine ausgewählte Station.
        - ghcnd-inventory.txt
            - URL: https://www.ncei.noaa.gov/pub/data/ghcn/daily/ghcnd-inventory.txt
            - Dieses Inventar enthält Informationen zu den verfügbaren Messgrößen (wie TMIN und TMAX) und deren Zeiträumen.
        - by_station/{station_id}.csv
            - URL: https://www.ncei.noaa.gov/pub/data/ghcn/daily/by_station/{station_id}.csv
            - Hier werden die Wetterdaten einer spezifischen Station im CSV-Format abgerufen.
        - all//{station_id}.dly

        - URL: https://www.ncei.noaa.gov/pub/data/ghcn/daily/all/{station_id}.dly
        - Dieser Endpunkt wird als Fallback genutzt, wenn die CSV-Daten nicht verfügbar sind; er liefert die Wetterdaten im DLY-Format.
- Datenfluss:
    - Flask sendet HTTP-Requests → NOAA API liefert Rohdaten → Backend verarbeitet die Daten → Frontend zeigt die Ergebnisse an.

4. CI/CD und Containerisierung (GitHub Actions + Docker)
- Containerisierung mit Docker
- Flask-Backend wird in einem Docker-Container ausgeführt.
- Das Frontend wird direkt vom Flask-Backend gehostet.
- CI/CD-Pipeline mit GitHub Actions:
    - Automatischer Build & Push des Docker-Images nach GHCR.
        - Build & Push nach GHCR
        - Automatischer Test der Funktionsfähigkeit der App bei Deployment durch den Docker Container.
    - Automatisierter Testlauf mit:
        - Python Unittests und
        - Formatchecks bzw. automatischer Formatierung via Black und flake8 sowie
        - Javascript Unittests via virtuellen Browsers: Es werden ausschließlich die logikrelevanten Kernfunktionen "processWeatherData" und "fillMissingYears" getestet.

4.4.3 Bedienungsanleitung

Diese Anleitung zeigt in einfachen Schritten, wie ein Nutzer das Wetterdaten-Analysetool (Anwendung) einfach und unkompliziert bedienen kann. Ergänzend findet sich eine Vorführung der Anwendung im Bewegtbildformat im Repository unter „/users_guide“.

1. Aufruf der Anwendung als Docker-Image

Hat der Nutzer Docker Desktop installiert, kann er sich im Terminal innerhalb des Docker-Fensters beim github Container Registry (ghcr) anmelden. Dazu dient dieser Terminal-Befehl (zu ergänzen um username und pat): {“*docker login ghcr.io -u* “+[github-username]+” *-p* “+[personal access token]}.

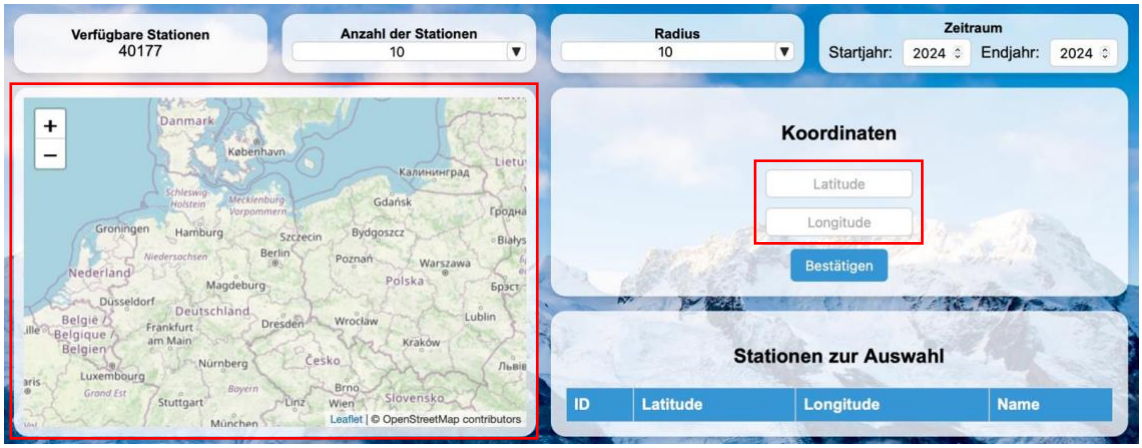
Nach erfolgreicher Anmeldung kann das Image abgerufen werden. Dazu dient der folgende Terminal-Befehl: {*curl -sL https://raw.githubusercontent.com/simondubiel/projekt_funk/e31090df59572f9212903ee1172e5d1dd11de777/docker-compose.yml -o docker-compose.yml && docker compose up --build -d*}.

Das Image wird unter „Images“ angezeigt. Mit einem Click auf den Pfeil in der entsprechenden Zeile (Play-Symbol) öffnet sich ein Fenster. Hier genügt ein weiterer Click auf „Run“.

Der Container wir gestartet und wird unter „Container“ angezeigt. In der entsprechenden Zeile wird der Port „8080:8080“ angezeigt. Mit einem Click öffnet sich dieser sog. Localhost. Die Anwendung öffnet sich in einem Browserfenster. Das neuladen der Seite führt immer zum Zurücksetzen der erfolgten eingaben in der App (Neustart der Benutzeroberfläche). Für einen Neustart der Anwendung insgesamt muss der Port oder respektive der Container gestoppt und erneut gestartet werden.

2. Standort festlegen

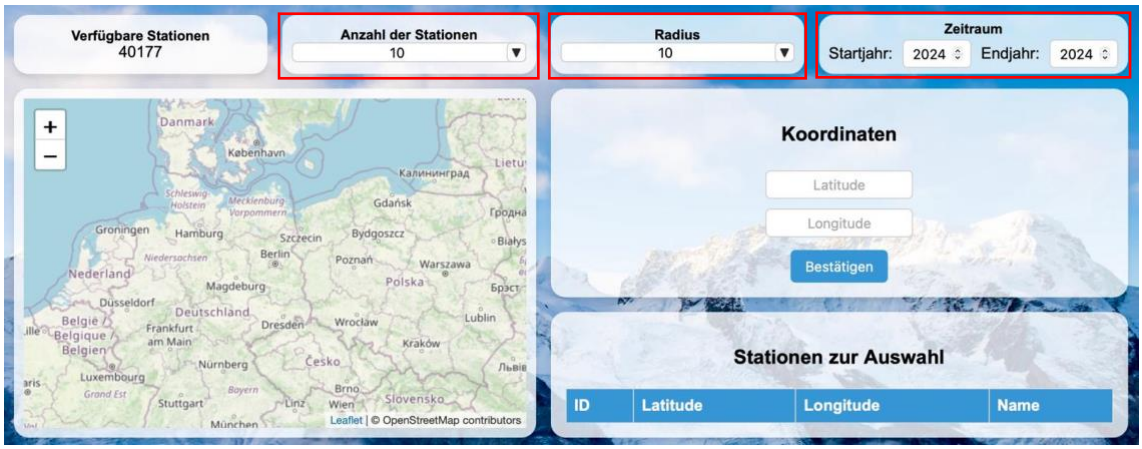
Der Nutzer gibt die gewünschten Koordinaten (Breiten- und Längengrad) ein oder wählt einen Ort mit Klick auf die Karte aus. Im zweiten Fall werden die Koordinaten des Standorts automatisch für die Auswahl übernommen.



3. Einstellungen anpassen

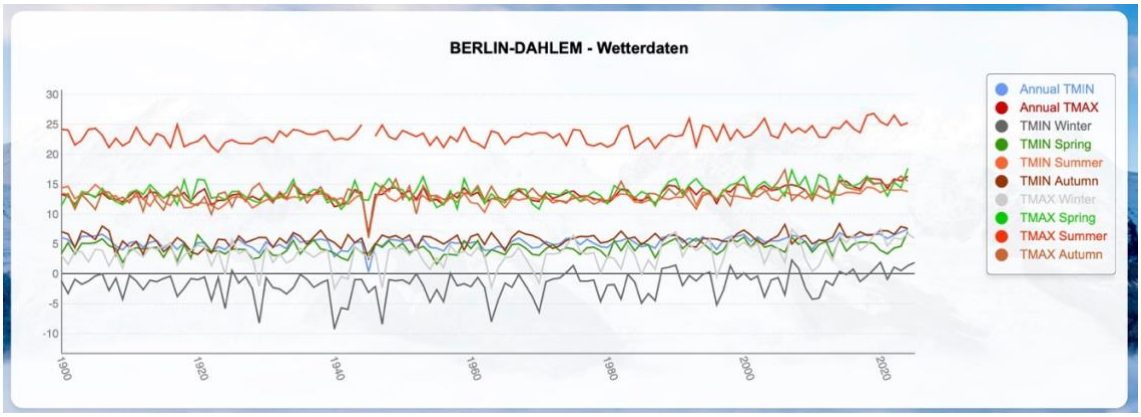
Der Benutzer kann mehrere Filterkriterien definieren:

- Wie viele Wetterstationen sollen höchstens angezeigt werden (0-10)?
- Wie groß soll der Suchbereich um den Standort sein (Radius in Kilometern 0-100)?
- Für welchen Zeitraum sollen angezeigte Wetterstationen Daten bereitstellen (Start- und Endjahr)?



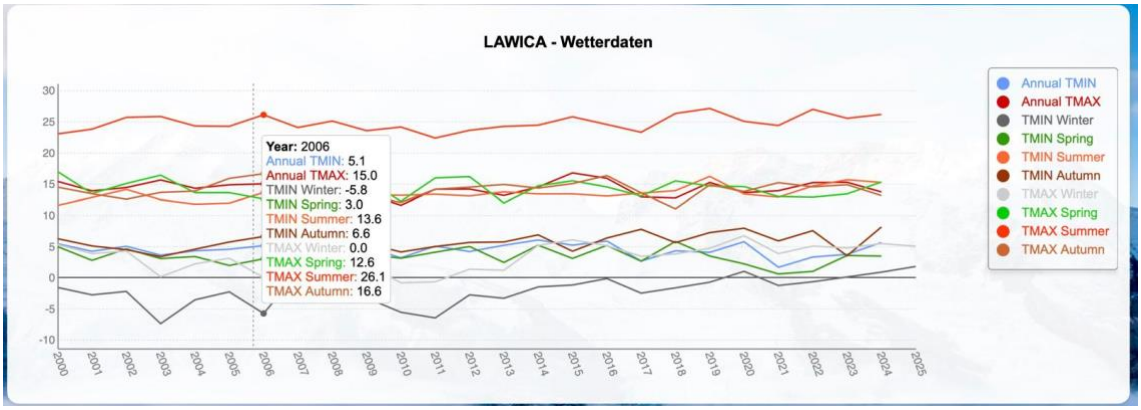
4. Daten anzeigen

Mit dem Klick auf „Bestätigen“ kann der Nutzer die Wetterdaten laden und schließlich anzeigen lassen. Die Daten erscheinen in Tabellen (mit Durchschnittswerten) und einem Diagramm (mit Temperaturverläufen).

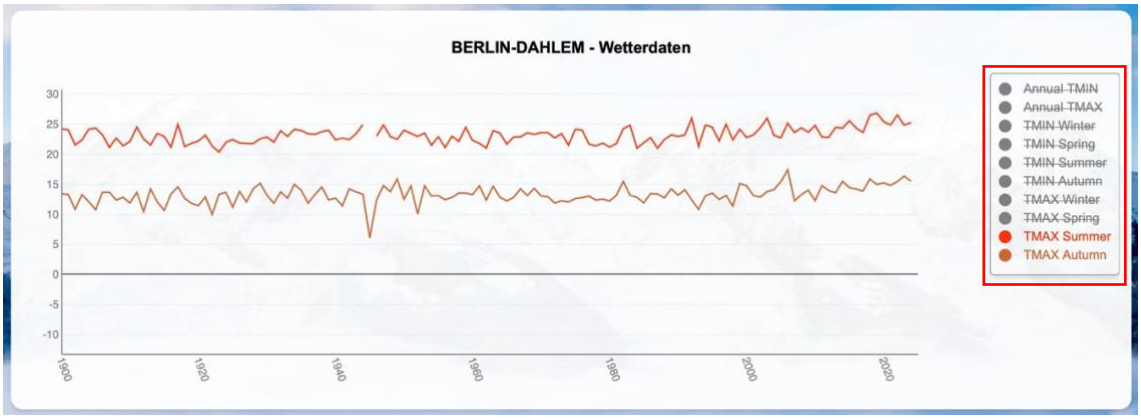


5. Das Diagramm nutzen

Durch Bewegen der Maus können einzelne Details eingesehen werden.



Per Klick in der Legende können einzelne Linien ein- oder ausgeblendet werden. Jeder Klick auf einen Eintrag schaltet die Sichtbarkeit der entsprechenden Temperaturdaten (wie beispielsweise TMIN, TMAX oder saisonale Werte) um.



Mögliche zukünftige Erweiterungen

Caching (Redis/Memcached)

→ API-Anfragen zwischenspeichern, um Ladezeiten zu reduzieren.

Separate Frontend-Container mit Nginx

→ Falls das Frontend komplexer wird.

Datenbank hinzufügen

→ Falls Wetterdaten länger gespeichert werden sollen.

Erweiterte API-Funktionen

→ Mehr Wetterdaten (z. B. Niederschlag, Windgeschwindigkeit).

4.5 Test der Anwendung

Zum Testen der Funktionalität der Anwendung wurde diese anhand drei verschiedener Orte (Koordinaten) getestet. Im Folgenden wird die Durchführung der Tests ausführlich beschrieben. Screenshots der Anwendung zum Test der drei Standorte sind dem Anhang beigefügt.

1. Start der Anwendung und Initialisierung

2. Auswahl der Koordinaten und Parameter

- Es wurden drei verschiedene Koordinaten ausgewählt, um die Funktionalität der Anwendung zu testen:
  - 1. Nürnberg, Deutschland: Ausgewählt durch Klicken auf der interaktiven Karte.
  - 2. Feuerland, Argentinien: Koordinaten wurden manuell eingegeben (-53, -79).
  - 3. Kapstadt, Südafrika: Ebenfalls über die Karte ausgewählt.
- Für jeden Standort wurden folgende Parameter festgelegt:
  - Anzahl der Stationen: Maximal 10 Stationen.
  - Radius: Bis zu 100 km
  - Zeitraum: Variabel, z.B. für Nürnberg von 1920 bis 2024.

- Die Koordinaten konnten entweder manuell in Eingabefelder eingegeben oder durch Klicken auf der Karte ausgewählt werden. Bei der Kartenwahl wurden die Koordinaten automatisch übernommen.

3. Datenabfrage und Filterung

- Nach Eingabe der Koordinaten und Parameter wurde auf "Bestätigen" geklickt, um die Wetterdaten zu laden.
- Die Anwendung hat die Stationen gefiltert, basierend auf den Kriterien:
    - Stationen mussten sowohl minimale (TMIN) als auch maximale (TMAX) Temperaturdaten für den gewählten Zeitraum liefern.

4. Analyse der Wetterdaten

- Die Wetterdaten wurden für jede ausgewählte Station in Tabellen und Diagrammen angezeigt:
    - Tabellen: Zeigten jährliche und saisonale Durchschnittswerte für TMIN und TMAX.
    - Diagramme: Visualisierten die Temperaturverläufe über die Jahre.
- Durch Klicken auf die Legende konnten einzelne Linien im Diagramm ein- oder ausgeblendet werden, z.B. um nur Sommer-TMAX oder Winter-TMIN anzuzeigen.

5. Interaktion mit der Benutzeroberfläche

- Die Karte erlaubte es, den Standort durch Klicken zu ändern. Wenn bereits Stationen angezeigt waren, konnte durch Klicken außerhalb des aktuellen Radius ein neuer Standort gewählt werden.
- Eine Liste der verfügbaren Stationen im gewählten Radius wurde ebenso angezeigt, aus der eine Station für eine detaillierte Analyse ausgewählt werden konnte.
- Im Diagramm konnte durch Hovering über die Linien detaillierte Informationen zu bestimmten Jahren abgerufen werden.

6. Fazit

Der Test verlief erfolgreich, und die Anwendung lieferte korrekte Wetterdaten für Nürnberg, Feuerland und Kapstadt.

4.6 4.6 Eingesetzte Werkzeuge

Im Rahmen des Projekts wurden mehrere Werkzeuge wie Programmiersprachen und Technologien eingesetzt, um eine leistungsfähige, interaktive und datengesteuerte Web-Applikation zu erstellen. Für das Backend wurde Python mit Unterstützung des Flask-Frameworks eingesetzt, um Daten abzurufen, zu verarbeiten und als API bereitzustellen. Auf der Clientseite wurde JavaScript genutzt, wobei Bibliotheken wie Leaflet und D3.js verwendet wurden, um interaktive Karten, Diagramme und dynamische Benutzeroberflächen zu erstellen. HTML und CSS bilden das Grundgerüst für die Gestaltung der Weboberfläche. HTML sorgt für die Struktur und CSS für das Layout und Styling.

Python

Technik/Bibliothek	Beschreibung
Flask	Web-Framework zur Erstellung von Routen, Rendering von Templates und Bereitstellung von JSON-Files.
Flask-CORS	Erweiterung zur Unterstützung von Cross-Origin Ressource Sharing (CORS) für externe Requests.
Requests	Bibliothek zum Durchführen von HTTP-Anfragen (z. B. zum Abrufen von NOAA-Daten).
Pandas	Datenanalyse und -manipulationsbibliothek zur Verarbeitung und Filterung von Station- und Inventardaten.
io.StringIO	Ermöglicht das Einlesen von Textdaten als file-like Objekt, um diese mit Pandas zu verarbeiten.
math (radians, sin, cos, sqrt, atan2)	Standard-Python-Modul für mathematische Berechnungen.
threading	Ermöglicht das parallele Ausführen von Aufgaben (z. B. Vorladen der Daten im Hintergrund).

Java Script Code

Technik/Bibliothek	Beschreibung
DOM-Methoden	Nutzung von document.getElementById, querySelector, createElement zur Manipulation des HTML-DOM.

Fetch API	Asynchrone HTTP-Anfragen (AJAX) zum Abrufen von Daten vom Server.
Asynchrone Programmierung (async/await, Promises)	Ermöglicht den nicht-blockierenden Abruf und die Verarbeitung von Daten.
Leaflet	JavaScript-Bibliothek für interaktive Karten, Marker, Kreise und Kartenfokus.
D3.js	Bibliothek zur Erstellung von datengetriebenen Diagrammen und Visualisierungen (SVG-basierte Charts).
Event Handling	Nutzung von addEventListener zur Reaktion auf Klicks, Input-Events und andere Benutzerinteraktionen.
Template Literals	Ermöglicht das Einfügen von Variablen in Strings (z. B. für dynamische Überschriften).

5 Projektabschluss

5.1 Projektabschlussbericht

Das Projekt zur Anwendungsentwicklung eines Wetterdaten-Analysetools wurde im Zeitraum vom 08.01.2025 bis zum 14.03.2025 erfolgreich durchgeführt und abgeschlossen. Ziel war die Bereitstellung einer Web-Applikation zur Analyse historischer Wetterdaten basierend auf der GHCN-Datenbasis.

Alle funktionalen Anforderungen aus Kapitel 4.1 wurden erfolgreich umgesetzt, darunter insbesondere die Filterung von Wetterstationen, die Abfrage historischer Wetterdaten und deren grafische Darstellung. Ebenfalls wurden die aus Kapitel 4.2 nicht funktionalen Anforderungen, wie die Testabdeckung von über 80% und eine begrenzte Ressourcennutzung erfüllt.

Dennoch sind während der Projektdurchführung einige der in Kapitel 3.4 ermittelten Risiken aufgekommen. So führte der kurzzeitige krankheitsbedingte Ausfall des Projektleiters Malte Paffen zu organisatorischen Schwierigkeiten wie Missverständnissen innerhalb des Teams, verzögerte Kundenkommunikation und Überlastung der Projektmitglieder. Auch technische Schwierigkeiten, wie die Implementierung der Weltkugel oder unzureichende Testabdeckung im Backend konnten zunächst nicht vermieden werden. Durch umfassende Qualitätssicherheitsmethoden und Maßnahmen zur Risikobehandlung konnten die Risken jedoch rechtzeitig abgefedert werden. Das finale Produkt wurde dem Kunden zeitgereicht überreicht.

5.2 Erfahrungsbericht

Das Projekt hat allen Projektbeteiligten gezeigt, wie essenziell klare Kommunikation und strukturierte Planung sind. Regelmäßige Meetings und eine zentrale Dokumentation ermöglichten eine effiziente Zusammenarbeit. Die flexible Aufgabenverteilung verhalf bei dem Auffangen von Engpässen. Frühzeitiges Testen und iterative Optimierungen waren entscheidend für die technische Umsetzung. Insgesamt waren die klaren Verantwortlichkeiten, regelmäßige Abstimmungen und ein strukturierter Entwicklungsprozess die wichtigsten Erfolgsfaktoren.

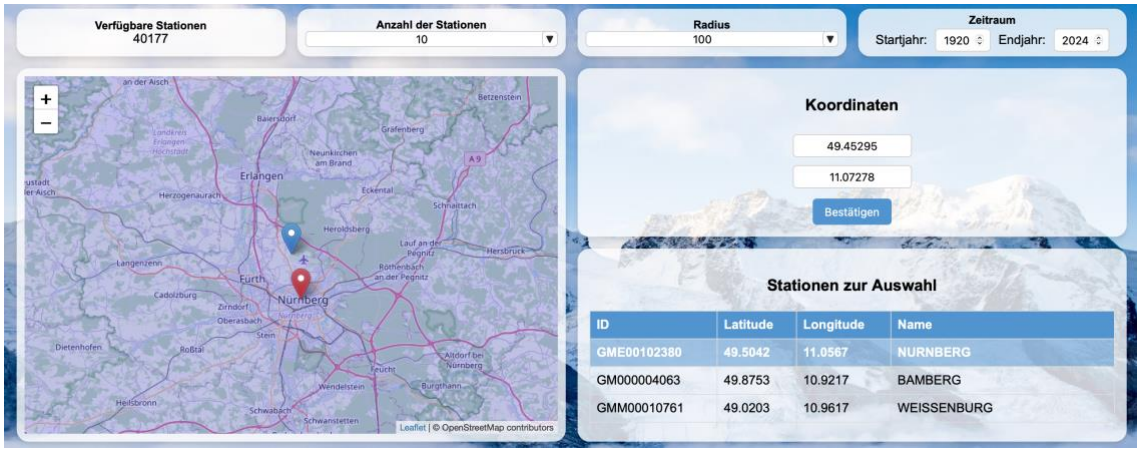
6 Anhang

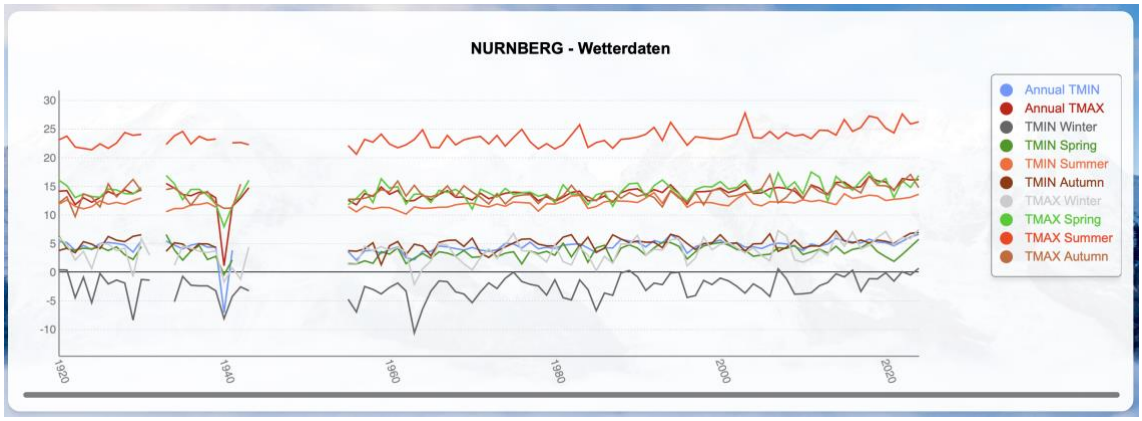
Meeting Protokoll

Datum	Dauer	Kategorie	Ort	Teilnehmer	Themen
08.01.25	0:20h	Interne Absprache	Virtuell	Alle	Gruppenorganisierung, Besprechung der Aufgabenstellung und Setup aller relevanten Anwendungen (VSC, Github usw.).
20.01.25	2:30h	Gemeinsame Gruppenarbeit	Kursraum	Nikolas, Malte	Allgemeine Projektplanung und Besprechung der Vorgehensweise. Definieren von Meilensteinen und Einteilung von Verantwortlichen. Arbeit an Frontend und Backend.

22.01.25	1:00h	Gemeinsame Gruppenarbeit	Kursraum	Alle	Festlegung der Grundstruktur für den Call mit dem Kunden und Erstellung von Slides zu den entsprechenden Aufgaben.
29.01.25	3:00h	Gemeinsame Gruppenarbeit	Zuhause Nikolas	Alle	Finalisieren der Präsentation mit dem Kunden. Austausch über die Umsetzung funktionaler und nicht funktionaler Anforderungen.
02.02.25	1:00h	Interne Absprache	Virtuell	Alle	Vorbereitung auf die Kundenpräsentation.
03.02.25	0:50h	Extern	Virtuell	Alle + Kunde	Vorstellung des aktuellen Stands des Projektes und Einholung von Feedback und Verbesserungsvorschlägen. Klärung von Fragen.
05.02.25	0:30h	Beratung	Virtuell	Alle + Consultant	Beratung durch internen Consultant über die technische Umsetzung des Projektes.
10.02.25	0:20h	Interne Absprache	Virtuell	Alle	Allgemeine Absprache zum aktuellen Stand, den Next Steps und aktuellen Problemen.
12.02.25	1:30h	Gemeinsame Gruppenarbeit	Kursraum	Nikolas, Malte	Implementierung der Tests und Optimierung der CI/CD-Pipeline
17.02.25	0:20h	Interne Absprache	Virtuell	Alle	Allgemeine Absprache zum aktuellen Stand, den Next Steps und aktuellen Problemen.
19.02.25	1:30h	Gemeinsame Gruppenarbeit	Virtuell	Nikolas, Malte	Grafische Darstellung der Wetterdaten im Frontend mittels d3.js.
24.02.25	0:20h	Interne Absprache	Virtuell	Alle	Allgemeine Absprache zum aktuellen Stand, den Next Steps und aktuellen Problemen.
26.02.25	1:30h	Gemeinsame Gruppenarbeit	Kursraum	Alle	Weiterführung der Dokumentation auf Basis des aktuellen technischen Standes und diversen Projektmanagement Methodiken.
03.03.25	0:20h	Interne Absprache	Virtuell	Alle	Allgemeine Absprache zum aktuellen Stand, den Next Steps und aktuellen Problemen.
06.03.25	0:45h	Extern	Hybrid	Alle	Abnahmetermin mit dem Kunden. Testung der Anwendung. Klärung von Fragen hinsichtlich der Projektmanagement-Dokumentation.
12.03.25	1:00h	Interne Absprache	Virtuell	Alle	Finale Anpassungen an der Anwendung und der Dokumentation. Vorbereitung zur Abgabe.

Screenshots zum Test der Anwendung (1. Standort: Nürnberg)





NURNBERG - Jährliche Durchschnittswerte

Year	Annual TMIN	Annual TMAX
1920	5.3	14.1
1921	5.1	14.2
1922	3.8	11.8
1923	4.6	12.9
1924	3.9	12.2
1925	5.0	13.0
1926	5.2	14.0
1927	5.0	13.4

NURNBERG - Saisonale Durchschnittswerte

Year	TMIN Winter	TMIN Spring	TMIN Summer	TMIN Autumn	TMAX Winter	TMAX Spring	TMAX Summer	TMAX Autumn
1920	0.4	6.3	11.8	3.7	6.1	16.1	23.0	12.0
1921	0.3	4.1	12.6	4.1	4.7	15.1	23.7	13.1
1922	-4.5	3.8	11.4	3.3	2.1	13.0	21.8	9.7
1923	-0.9	4.1	11.0	5.3	3.6	13.6	21.6	13.6
1924	-5.4	4.0	11.4	4.9	0.6	13.2	21.3	13.1
1925	-0.3	4.4	12.4	4.1	5.0	13.0	22.4	11.3
1926	-2.1	3.7	11.8	6.2	4.9	14.5	21.6	15.4

Screenshots zum Test der Anwendung (2. Standort: Feuerland)

Screenshots zum Test der Anwendung (3. Standort: Kapstadt)

7 Selbstständigkeitserklärung

Wir versichern hiermit, dass wir die vorliegende Arbeit mit dem Thema: Dokumentation zur Anwendungsentwicklung eines Wetterdaten-Analysetools selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Villingen-Schwenningen, 11.03.2025

Nikolas Diehl, Simon Dubiel, Quirin Großhauser, Angelo Iacona, Malte Paffen