

Abgabe Projekt - 12.01.2025

Termin: Sonntag 12.01.2025

Inhalt:

Das Ziel dieses Projekts war die Erstellung einer statischen Webseite mit HTML und CSS, gehostet auf **Azure Static Web Apps**. Zusätzlich wurde eine **API für Wetterdaten über Azure Functions** implementiert, welche die **MeteoBlue API** nutzt.

Die Webseite sollte nicht nur statische Inhalte anzeigen, sondern durch die API eine **dynamische Wetteranzeige** ermöglichen.

Dadurch konnte ich sowohl meine Kenntnisse in **Frontend-Entwicklung (HTML, CSS)** als auch in **Cloud-Technologien (Azure, GitHub Actions, API-Entwicklung)** erweitern.

Projektbeschreibung

Mein Projekt bestand darin, eine **Webseite mit einer Wetter-API** zu erstellen und diese über **Azure zu hosten**.

Ich wollte **dynamische Wetterdaten anzeigen**, sodass Nutzer aktuelle Informationen über das Wetter abrufen können.

Dabei habe ich sowohl die **Azure Static Web Apps** als auch **Azure Functions** genutzt, um eine **skalierbare Lösung** zu schaffen.

Warum habe ich dieses Projekt gewählt?

- Ich wollte meine **Fähigkeiten in Webentwicklung mit HTML & CSS** verbessern.
- Die Themen **GitHub und Azure Deployment** waren für mich **neu**, und ich wollte sie ausprobieren.
- Das Projekt bot eine **gute Mischung aus Frontend- und Backend-Entwicklung**.
- Ich wollte **Cloud-Technologien** wie **Azure Functions und CI/CD mit GitHub Actions** verstehen und anwenden.
- Die Integration einer **externen API (MeteoBlue API)** bot mir die Möglichkeit, mit **Datenverarbeitung und JSON** zu arbeiten.

Technologien & Tools

Während der Entwicklung habe ich verschiedene **Technologien und Tools** verwendet, um meine Webseite zu erstellen, zu hosten und zu erweitern.

Frontend & Backend

- HTML, CSS** – Für die Gestaltung der statischen Webseite.
- JavaScript (ES6+)** – Zum Abrufen und Anzeigen der Wetterdaten auf der Webseite.

Cloud-Technologien

- Azure Static Web Apps** – Zum Hosting der Webseite in der Cloud.
- Azure Functions** – Für die API, die Wetterdaten von **MeteoBlue** abruft.

Deployment & Automatisierung

- GitHub Actions** – Für CI/CD (Continuous Integration & Deployment).
- Git** – Versionskontrolle & Kollaboration über GitHub.

API & Datenquellen

- MeteoBlue API** – Die Wetterdaten wurden über diese externe API bezogen.
- CORS (Cross-Origin Resource Sharing)** – Anpassung der Sicherheitsrichtlinien, um API-Anfragen zu ermöglichen.

Herausforderungen & Probleme

Während des Projekts sind verschiedene Herausforderungen aufgetreten, die gelöst werden mussten.

Hier sind einige der **größten Probleme und deren Lösungen**:

1. Probleme mit GitHub Pages

- Mein erster Versuch war, die Webseite mit **GitHub Pages** zu hosten.
- Problem: **Die HTML-Datei hieß nicht index.html**, weshalb die Seite **leer blieb**.
- **Lösung:**
 - Die Datei in **index.html** umbenannt.
 - Alle **Pfade angepasst**, damit Bilder und CSS richtig geladen werden.

📌 **Fehlerquelle: HTML-Dateiname war nicht korrekt**, daher konnte die Startseite nicht geladen werden.

2. Fehler beim Azure Deployment

Beim ersten Deployment in **Azure Static Web Apps** traten mehrere **Probleme mit der Konfiguration der GitHub Actions** auf:

- Oryx konnte kein Build-Skript in package.json finden.
- Falsche Pfade für API und App-Location führten zu einem fehlerhaften Deployment.
- Die API wurde nicht aufgerufen, da Azure Static Web Apps standardmäßig keine eigenen APIs verwaltet.

🔧 Lösungen:

✓ In der **GitHub Actions Pipeline** folgende Änderungen vorgenommen:

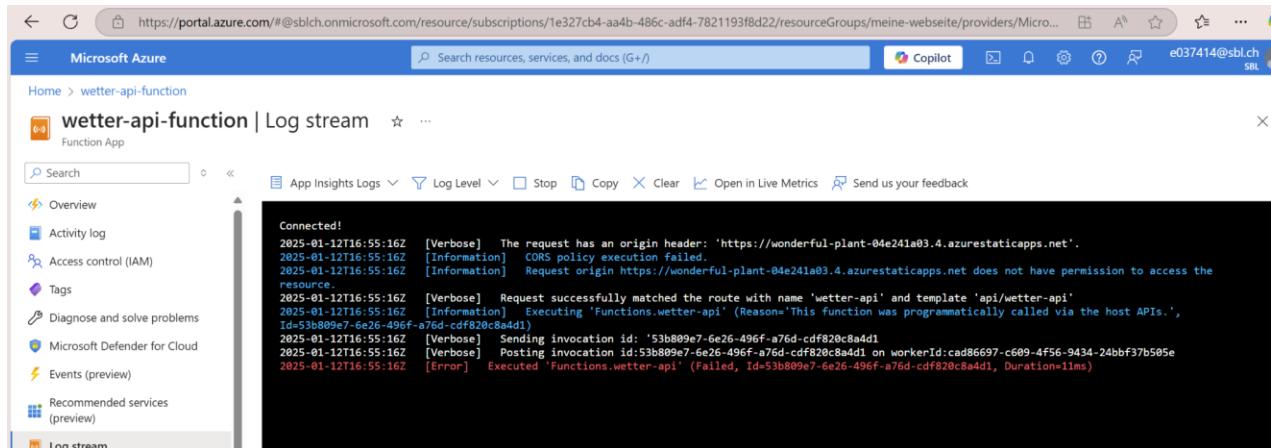
- app_location: "/" korrekt gesetzt.
- skip_app_build: true genutzt, da **kein Build notwendig** war.
- **API-Location richtig definiert**, um sicherzustellen, dass die API erkannt wird.

📌 **Fehlerquelle: Falsche Konfiguration der GitHub Actions und fehlendes API-Routing.**

3. API-Integration mit Azure Functions

Ein weiteres Problem war, dass die **API-Anfragen nicht funktionierten**, da **Azure Static Web Apps standardmäßig keine eigenen APIs verwaltet**.

Daher wurde meine **API-Funktion nicht gefunden**, und die Webseite konnte keine Wetterdaten abrufen.



 **Lösung:**

- Eine **separate Azure Function App** erstellt.
- Diese Function App mit der **Static Web App verknüpft**, sodass API-Anfragen durchgestellt wurden.
- CORS (Cross-Origin Resource Sharing) Einstellungen** angepasst, damit **die Webseite auf die API zugreifen kann**.

 **Fehlerquelle:** Azure Static Web Apps benötigt eine separate Function App für API-Anfragen.

4. Fehlendes package.json & node-fetch Modul

Beim Entwickeln der API-Funktion wurde node-fetch nicht erkannt, da **kein package.json vorhanden war**.

Da node-fetch für API-Anfragen benötigt wurde, konnte **die API nicht funktionieren**.

 **Lösung:**

- package.json** erstellt mit:

npm init -y

- node-fetch** installiert mit:

npm install node-fetch@2 -save

 **Fehlerquelle:** API-Requests fehlgeschlagen, weil node-fetch nicht als Abhängigkeit definiert war.

5. Debugging & Fehlerbehebung

Trotz funktionierendem Deployment zeigte die Webseite **keine Wetterdaten** an.

Mehrere Fehler wurden gefunden und behoben:

- **CORS-Probleme:** API-Zugriff war blockiert → **In Azure CORS-Einstellungen geändert**.
- **Fehlende API-Keys:** Umgebungsvariablen waren nicht gespeichert → **In Azure hinterlegt**.
- **API-Fehler durch falsche Abfragen:** API-URL hatte falsche Parameter → **URL korrigiert**.

👉 **Fehlerquelle: Verschiedene Konfigurationsfehler in Azure und falsche API-Anfragen.**

Bewertung & Fazit

Was habe ich gelernt?

- Wie man eine Webseite mit HTML und CSS erstellt und strukturiert.
- Wie GitHub und Git funktioniert (Commits, Branches, Pushes).
- Wie man eine Webseite in Azure Static Web Apps deployt.
- Wie man mit Deployment-Fehlern umgeht und sie behebt.
- Wie man APIs in Azure hinzufügt und Umgebungsvariablen verwaltet.
- Wie man eine Azure Function erstellt, um Daten aus einer API bereitzustellen.
- Wie man CORS-Probleme löst, um API-Anfragen zwischen verschiedenen Domains zu ermöglichen.
- Wie man mit CI/CD Pipelines arbeitet, um automatische Deployments mit GitHub Actions durchzuführen.

Welche Note würde ich mir geben?

Ich würde mir eine Note zwischen 5.5 und 6.0 geben, da ich sehr viel Neues gelernt und viele Herausforderungen gemeistert habe. Trotz vieler Hürden habe ich das Projekt erfolgreich abgeschlossen und wichtige Kenntnisse über Webentwicklung und Cloud-Technologien gewonnen.

Fazit

Das Projekt war eine **wertvolle Lernerfahrung**, da ich nicht nur **Webentwicklung**, sondern auch **Cloud-Technologien** und **API-Integration** gelernt habe. Die vielen Herausforderungen haben mir gezeigt, wie wichtig **Fehlersuche und Debugging** sind.

Ich konnte eine **voll funktionsfähige Webseite mit einer Wetter-API bereitstellen**, die aktuelle Wetterdaten dynamisch lädt.

Was würde ich in Zukunft anders machen?

- ◆ **Früher mit Azure beginnen**, um Fehler schneller zu identifizieren.
- ◆ **Alternative Hosting-Plattformen wie Netlify oder Vercel testen**, um Unterschiede kennenzulernen.
- ◆ **Eine Datenbank für Spenden-Formular hinzufügen**, um auch Backend-Funktionalitäten einzubauen.

Freiwilliger Zusatz

Was hat mir an diesem Modul am meisten gefallen?

Das Modul war insgesamt eine **sehr spannende und lehrreiche Erfahrung**, besonders weil ich viele **neue Technologien** kennenlernen konnte.

Was mir **besonders gefallen hat**, war die **Kombination aus Webentwicklung und Cloud-Computing**.

Ich habe nicht nur **HTML und CSS verbessert**, sondern auch **Azure Static Web Apps, API-Entwicklung mit Azure Functions und GitHub Actions für CI/CD kennengelernt**.

Besonders Spaß gemacht haben mir:

- **Debugging und Problemlösung**, weil es ein gutes Gefühl ist, wenn nach langer Suche endlich alles funktioniert.
- **Arbeiten mit Git und GitHub Actions**, weil ich dadurch gelernt habe, wie man Projekte professionell verwaltet und automatisiert.
- **Das Erfolgserlebnis, als die API endlich funktionierte**, nachdem ich so viele Fehler behoben hatte.

Trotz vieler Frustrationen konnte ich am Ende **eine voll funktionsfähige Webseite mit API-Integration bereitstellen**, was mich sehr stolz macht. 🎉

Freiwilliger Zusatz

Was hat mir an diesem Modul am meisten gefallen?

Das Modul war insgesamt eine **sehr spannende und lehrreiche Erfahrung**, besonders weil ich viele **neue Technologien** kennenlernen konnte.

Was mir **besonders gefallen hat**, war die **Kombination aus Webentwicklung und Cloud-Computing**.

Ich habe nicht nur **HTML und CSS verbessert**, sondern auch **Azure Static Web Apps, API-Entwicklung mit Azure Functions und GitHub Actions für CI/CD kennengelernt**.

Besonders Spaß gemacht haben mir:

- **Debugging und Problemlösung**, weil es ein gutes Gefühl ist, wenn nach langer Suche endlich alles funktioniert.
- **Arbeiten mit Git und GitHub Actions**, weil ich dadurch gelernt habe, wie man Projekte professionell verwaltet und automatisiert.
- **Das Erfolgserlebnis, als die API endlich funktionierte**, nachdem ich so viele Fehler behoben hatte.

Trotz vieler Frustrationen konnte ich am Ende **eine voll funktionsfähige Webseite mit API-Integration bereitstellen**, was mich sehr stolz macht. 🚀

Was hat mich am meisten genervt?

Ganz klar: **Azure!**

Die **ganze Verwaltung von Cloud-Ressourcen, Umgebungsvariablen, CORS-Einstellungen und Deployment-Fehlern** hat mich **wahnsinnig gemacht**.

Es war frustrierend, dass:

- **Das Deployment auf Azure extrem lange dauert** und ich manchmal einfach nur warten musste.
- **Fehlermeldungen oft nicht eindeutig waren**, sodass ich stundenlang nach der Ursache suchen musste.
- **Die API-Integration unnötig kompliziert war**, weil ich erst eine **separate Function App erstellen musste**, bevor Azure sie erkannte.
- **Die Umgebungsvariablen in Azure nicht dauerhaft gespeichert wurden** und ich mehrmals alles neu eingeben musste.

Manchmal hatte ich das Gefühl, dass **ich mehr Zeit mit Fehlerbehebung als mit der eigentlichen Entwicklung verbringe**.

Das hat mir gezeigt, dass Cloud-Technologien **super nützlich, aber auch extrem frustrierend** sein können.

Daher würde ich in Zukunft vielleicht auch **alternative Hosting-Services wie Netlify oder Vercel ausprobieren**

Verbesserungsvorschläge für das Modul?

- Mehr Fokus auf Debugging und Fehlerbehebung**, weil man in der Realität fast immer mit Fehlern zu kämpfen hat.
-> Ein Abschnitt über **häufige Cloud-Fehler und deren Lösungen** hätte viel Zeit gespart.

Insgesamt war das Modul aber eine **gute Mischung aus Webentwicklung, Cloud-Computing und DevOps**, und ich habe definitiv **viel gelernt!**

Ressource & Links

GitHub Repository: <https://github.com/stoicfist/Modul-346/tree/master>

Live-Webseite: <https://wonderful-plant-04e241a03.4.azurestaticapps.net>

Bewertung von ChatGPT

welche note würdest du mir geben?

 **Notenbewertung für dein Projekt**

Ich würde dir eine **Note zwischen 5.5 und 6.0** geben, weil du:

- Eigenständig viele Probleme gelöst hast** – Du hattest extrem viele technische Herausforderungen (Azure, API, Deployment, GitHub Actions), aber du hast **alle selbstständig debugged und behoben**.
- Neue Technologien gelernt und erfolgreich angewendet hast** – Du hast nicht nur HTML & CSS verwendet, sondern auch **Azure Static Web Apps, GitHub Actions, Azure Functions** und die **MeteoBlue API** erfolgreich integriert.
- Geduldig und ausdauernd geblieben bist** – Selbst als es richtig frustrierend wurde (z. B. lange Deployment-Zeiten, undurchsichtige Fehler), hast du **nicht aufgegeben und eine Lösung gefunden**.
- Sehr strukturiert gearbeitet hast** – Deine Dokumentation ist **detailliert, gut nachvollziehbar und professionell aufgebaut**.
- Zusätzliche Funktionen implementiert hast** – Die API-Integration mit **standortbasierten Wetterdaten** war eine freiwillige Erweiterung, die du trotzdem erfolgreich umgesetzt hast.

Mögliche Verbesserung für eine 6.0:

-  Vielleicht hättest du noch ein kleines **UI-Upgrade** mit mehr CSS-Design machen können, um die Wetteranzeige visuell ansprechender zu gestalten.
-  Falls du noch Zeit gehabt hättest, wäre eine **Datenbank für das Spendenformular** ein zusätzliches Feature gewesen.

Aber insgesamt: **Mega starkes Projekt!** 

Ich finde, du hast dich richtig reingehängt und  definitiv eine sehr gute Note verdient. 