**Cloud Computing Webseite mit Azure**

**Projektbeschreibung**

Mein Projekt bestand darin, eine **statische Webseite mit HTML und CSS** zu erstellen und diese über **Azure Static Web Apps** zu hosten. Die Webseite stellt meine Interessen, Projekte und eine Spendenfunktion dar. Ursprünglich lief die Webseite nur lokal mit Live Server in Visual Studio Code, aber ich wollte sie dauerhaft online verfügbar machen.

**Projektprozess & Entscheidungen**

**Warum habe ich dieses Projekt gewählt?**

* Ich wollte **Webentwicklung mit HTML & CSS** lernen und verbessern.
* GitHub und Azure Deployment waren neu für mich und ich wollte es ausprobieren.
* Es war eine **gute Mischung aus Frontend-Entwicklung und Cloud-Hosting**.

**Was hat gut geklappt?**

* **Webseite mit HTML und CSS war schnell umgesetzt**.
* **Debugging und Git**: Ich konnte Fehler systematisch finden und lösen.
* **Azure Static Web Apps konnte ich am Ende erfolgreich zum Laufen bringen**.

**Herausforderungen & Probleme**

**1. Probleme mit GitHub Pages**

* **Mein erster Versuch war, die Webseite mit GitHub Pages zu hosten.**
* **Problem:** Die HTML-Datei hieß nicht index.html, sodass die Seite leer blieb.
* **Lösung:** Umbenennen in index.html und Pfade anpassen.

**2. Fehler bei Azure Deployment**

* **Probleme beim Hochladen auf Azure Static Web Apps**
  + Erst falsches Repository oder Pfade angegeben
  + Build-Prozess ist fehlgeschlagen
  + Oryx konnte kein build-Script in package.json finden (obwohl ich keins brauchte)
* **Lösung:**
  + Ich habe skip\_app\_build: true im GitHub Workflow hinzugefügt
  + Habe app\_location: "/" korrekt gesetzt
  + Nach erneutem git push hat Azure die Webseite erfolgreich deployed

**3. Administrative Hürden mit Azure**

* Registrierung und Authentifizierung bei Azure war kompliziert.
* Ich musste **warten, bis Microsoft meinen Account freigeschaltet hat**.
* Einige Begriffe in Azure waren verwirrend (Resource Group, API Token, OIDC-Client).

**Gelerntes & Selbstbewertung**

* **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.
* **Welche Note würde ich mir geben?**
  + Ich würde mir eine **5.0 oder 5.5** geben.
  + Ich habe viele neue Dinge gelernt und bin drangeblieben, auch wenn es nervig war.
* **Wie könnte ich mein Lernen verbessern?**
  + Ich würde vielleicht früher mit Azure starten, um Probleme früher zu erkennen.
  + Vielleicht mehr mit Tutorials oder Dokumentationen arbeiten, statt alles selbst herauszufinden.

**Ressourcen & Screenshots**

* **GitHub Repository:** [Link zu deinem Repo]
* **Azure Static Web App URL:** [Link zu deiner Webseite]
* **Screenshots vom Debugging & Fehlerbehebung:** (Bereits gesendet)

**Freiwilliger Zusatz**

* **Was hat mir am meisten gefallen?**
  + Debugging und das Lösen von Fehlern
  + Git und Versionskontrolle
  + Am Ende eine funktionierende Webseite zu haben
* **Was hat mich am meisten genervt?**
  + Die ganzen administrativen Dinge mit Azure
  + Registrierung, Authentifizierung, Freischaltung
  + Warten auf Microsoft und Azure-Freigaben
* **Verbesserungsvorschläge für den Kurs**
  + Vielleicht eine kurze Azure-Einführung als Teil des Kurses
  + Alternative Cloud-Hosting-Möglichkeiten wie Netlify oder Vercel zeigen

API Key fü meteoblue separat auf azure mit enviroments variables, damit die keys nicht auf github falls ich es bei wetter.js speichere zu sehen und dann missbraucht wird.

**Projektziel**

Das Ziel dieses Projekts ist die Implementierung einer Wetter-API mittels **Azure Functions** und die Bereitstellung der Wetterdaten auf einer **Azure Static Web App**. Die Wetterdaten werden von der **MeteoBlue API** bezogen.

**1. Ausgangssituation & Probleme**

**1.1 Fehlende API-Anbindung in Azure Static Web Apps**

* Ursprünglich war die Wetter-API als **Azure Function** im Ordner /api geplant.
* Die API wurde jedoch nicht aufgerufen, da Azure Static Web Apps standardmäßig **keine eigenen APIs verwaltet**, es sei denn, eine Function App ist explizit verlinkt.
* Fehlermeldung: **HTTP ERROR 404** beim Aufruf von https://wonderful-plant-04e241a03.4.azurestaticapps.net/api/wetter-api.

**1.2 Environment Variables nicht verfügbar**

* Die Variable METEOBLUE\_API\_KEY wurde über **Azure Static Web Apps -> Environment Variables** gesetzt.
* Beim **Neuladen** der Seite verschwand die Variable automatisch.
* Die API konnte daher den API-Key nicht abrufen und lieferte keine Daten.

**1.3 Deployment-Probleme mit GitHub Actions**

* Die GitHub Action für den **Build & Deploy Prozess** schlug fehl.
* Fehler: **Oryx konnte keinen Build-Prozess starten**, da kein build oder build:azure Skript in package.json gefunden wurde.
* Fehlerhafte Konfiguration der api\_location, da /api nicht richtig erkannt wurde.

**1.4 Node.js nicht verfügbar in Lokaler Entwicklungsumgebung**

* Beim Testen von node wetter-api.js im lokalen Terminal gab Bash die Meldung command not found aus.
* Vermutung: Node.js ist nicht installiert oder nicht richtig in den PATH-Variablen.

**2. Lösungsversuche & Korrekturen**

**2.1 Upgrade auf eine bezahlte Azure Hosting-Plan für API-Support**

* **Lösung:** Das **Azure Static Web App Hosting** wurde auf den **Standard-Plan** aktualisiert.
* **Erwartung:** Unterstützung für **eigene APIs** und dauerhafte Speicherung der Environment Variables.

**2.2 Erstellen einer separaten Function App**

* **Schritt:** Eine **Azure Function App** wurde erstellt und mit **Flex Consumption Plan** gehostet.
* **Erwartung:** Die Function App kann als Backend für die API-Anfragen genutzt werden.

**2.3 Anpassen der wetter-api.js Datei in Azure Function**

* process.env["METEOBLUE\_API\_KEY"] wurde debugged und geprüft.
* Eine Logging-Anweisung console.log("API Key:", apiKey) wurde eingefügt.

**2.4 Anpassen der GitHub Actions Pipeline**

* **api\_location wurde auf "api" gesetzt**, damit die Functions korrekt deployed werden.
* **skip\_app\_build wurde aktiviert**, da keine spezielle Build-Phase benötigt wird.

**2.5 Installation von Node.js für lokale Tests**

* Node.js wurde lokal installiert und der PATH-Variable hinzugefügt.
* node -v überprüft, ob die Installation erfolgreich war.

**3. Nächste Schritte**

✅ **Warten auf die erfolgreiche Bereitstellung der Azure Function App** ✅ **Verlinkung der Function App mit der Static Web App** ✅ **Testen der API unter /api/wetter-api** ✅ **Debugging, falls Fehler auftreten**