Wie funktioniert WarnMe eigentlich im Hintergrund?

Dieses Dokument erklärt die technischen Vorgänge, die sicherstellen, dass Warnmeldungen zuverlässig und schnell die richtigen Personen erreichen. Es beschreibt, wie das System Daten empfängt, verarbeitet und bewertet. Am Ende dieses Prozesses stehen konkrete Alarme. Ziel ist, die technischen Abläufe von WarnMe verständlicher und zugänglicher zu machen.

Datenverarbeitung und Alarmierung

Datenspeicherung / Application-Server

Die Daten des TTN-Netzwerks werden nicht dauerhaft gespeichert, weshalb sie für die Analyse an einen eigenen Server weitergeleitet und dort gespeichert werden müssen. TTN bietet hierfür verschiedene Integrationen wie die REST-API, WebHooks oder das MQTT-Protokoll an. Nach mehrfachem Ausprobieren verschiedener Möglichkeiten haben wir uns dazu entschieden einen Server bei Hetzner anzumieten, auf dem die Anwendung in einem Docker-Container betrieben wird. Zu Beginn verwendeten wir ein Bash-Skript, um Daten über die REST-API regelmäßig herunterzuladen. Mit zunehmender Anzahl an Sensoren und erschwerten Übertragungsbedingungen stieß dieses Verfahren jedoch an seine Grenzen.

Die REST-API ist nicht auf kontinuierliche Datenströme ausgelegt. Eine skalierbare Lösung wurde durch das MQTT-Protokoll implementiert, das eine sofortige Übertragung neuer Sensordaten ermöglicht. Ein maßgeschneidertes Python-Programm mit der Paho-Bibliothek übernimmt die Datenabfrage und Weiterleitung.

Da das Speichern der Daten in einer großen JSON-Datei mit wachsendem Datenvolumen ineffizient wurde, haben wir eine SQLite-Datenbank implementiert.

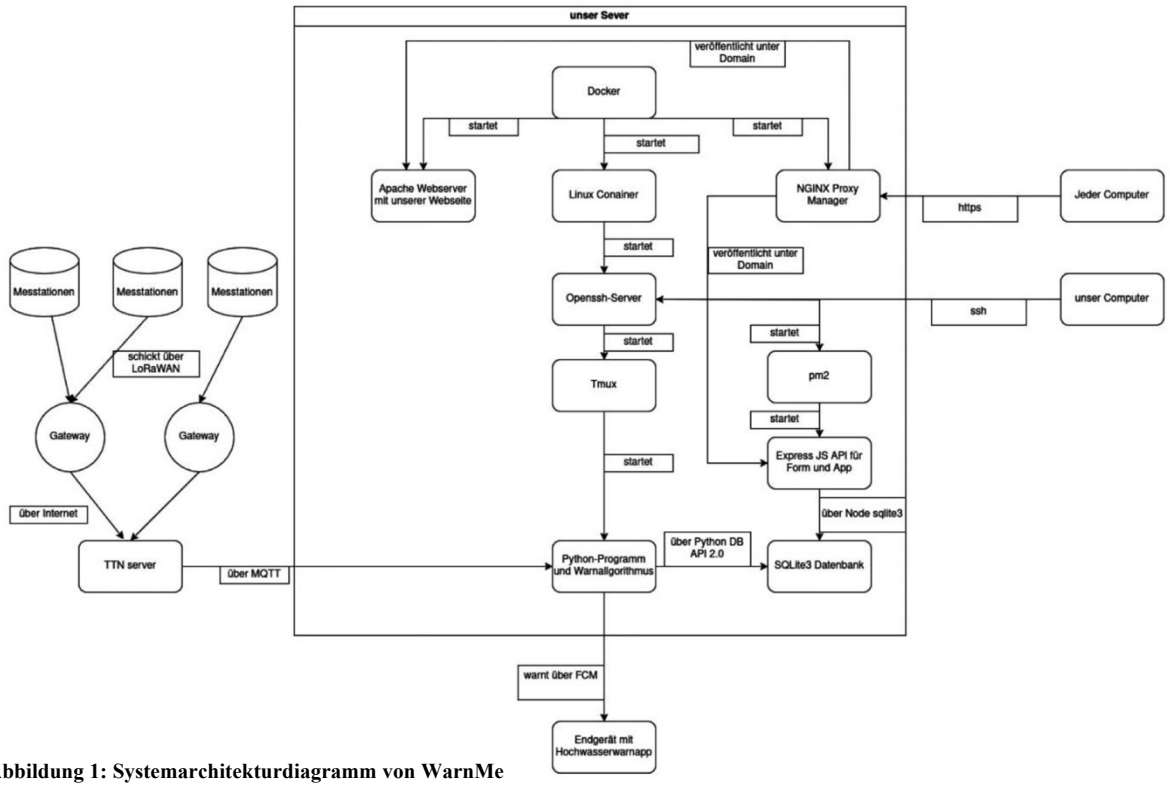

Abbildung 1: Systemarchitekturdiagramm von WarnMe

Um den Dauerbetrieb des Python-Programms sicherzustellen, wird Tmux verwendet. Dieses Tool ermöglicht das Fortführen von Prozessen im Hintergrund und erleichtert die Verwaltung. Das aktuelle Setup umfasst einen Docker-Container auf dem Hetzner-Server, in dem das Python-Programm die Daten über MQTT abruft und in die SQLite-Datenbank speichert.

Darüber hinaus haben wir eine API mithilfe von Express JS programmiert, die die letzten Pegelstände und die durschnitte an die App schickt. Diese API nimmt auch zusätzlich neue Sensorregistrierungen von warnme.info/form.html an und speichert sie in der Datenbank. Der ganze Code hierfür ist Open-Source und hier zu finden: [1].

Alarmierung

Wenn ein Hochwasser erkannt wird, müssen die Endnutzer natürlich gewarnt werden. Ursprünglich wollten wir dafür eine E-Mail-Liste verwenden, stellten jedoch schnell fest, dass dies für zeitkritische Warnungen heutzutage ungeeignet ist. Daraufhin versuchten wir, uns in die Warnsysteme von NINA und Katwarn einzuklinken, wurden jedoch ignoriert. Daher haben wir schließlich unsere eigene App entwickelt.

Die App ist mit Flutter und Dart programmiert, wodurch sie plattformübergreifend funktioniert. Es gibt also nur eine Codebasis für Android, iOS, Linux, Windows und macOS. Sie kann Warnmeldungen auch dann verschicken, wenn sie geschlossen ist und nicht im Hintergrund läuft. Dies ist möglich, weil wir Firebase Cloud Messaging nutzen, um direkt mit dem Push-Notification-Dienst des jeweiligen Geräts zu kommunizieren.

Neu ist, dass man in der App zwischen zwei Algorithmusstärken wählen kann. So können die Warnungen an den Wohnort angepasst werden, je nachdem, ob man näher am Bach oder weiter davon entfernt wohnt. Darüber hinaus zeigt die App nun die Pegelstände, den Zeitpunkt der Messung sowie die zugehörigen Durchschnittswerte an.

Eine weitere neu integrierte Funktion ist die Expertenrunde. Dies bedeutet, dass bei einer Hochwassergefahr zunächst ein Kreis von Experten informiert wird. Dieser kann beispielsweise die örtliche Feuerwehr und Gemeindemitarbeiter umfassen, welche über das jeweilige Hochwasserschutzkonzept unterrichtet sind. Diese können dann Fehlwarnungen ausschließen und sofort entscheiden welche Maßnahmen ergriffen werden müssen. Im Anschluss daran werden dann die Nutzer der App gewarnt. Durch diese hierarchische Warnung kann potenzielles Chaos vermieden und die notwendigen Maßnahmen sofort umgesetzt werden.

Abbildung 2: Screenshot der geöffneten Warnapp

Darüber hinaus ist die App nun im Google Play Store verfügbar. Der Play Store hat für Accounts, die nach dem 13. November 2023 erstellt wurden, ein neues Veröffentlichungssystem eingeführt. Zuerst muss ein interner Test durchgeführt werden, anschließend ein geschlossener Test, bevor die App schließlich veröffentlicht werden kann [2]. Wir haben den internen Test erfolgreich bestanden und befinden uns nun im geschlossenen Test. Das bedeutet, dass jeder, der möchte, die App über einen Link offiziell aus dem Play Store herunterladen kann [3]. Über die Google-Play-Suche ist die App jedoch noch nicht auffindbar. Wir gehen davon aus, dass wir bald Produktionszugriff erhalten und die App dann auch über die Suche verfügbar sein wird. Auch der Code für die App ist FOSS, sie finden ihn hier [4].

Literaturquellenverzeichnis

[1] https://github.com/NiklasHubGit/WarnMe-API, 01.01.2025

[2] https://www.knowband.com/blog/de/nachrichten/google-macht-geschlossene-tests-für-die-neue-app-bereitstellung-zur-pflicht-richtlinienaktualisierung-für-neue-individuelle-entwicklerkonten/, 01.01.2025

[3] https://play.google.com/store/apps/details?id=com.warnme.warnapp, 01.01.2025

[4] https://github.com/NiklasHubGit/WarnMe-App, 01.01.2025