

## BOT-43: Erstellung eines Common-Frameworks für (Content-)Services

Services im BHT-Bot kommunizieren alle über REST-Schnittstellen. Diese sind alle als Express-Anwendung mit JSON-Kommunikation implementiert, was für viel Code-Redundanz führt. Gleichzeitig benutzt kein Service strukturierte Darstellungen der Schnittstellen, wie Anfragen und Antworten zum Gateway, User Daten oder Rasa-Intents.

Ein spezieller, repetitiver, Unterfall der Microservices sind solche, die Content bereitstellen. Diese erhalten alle Nachrichten vom Gateway und senden ihre Antworten auch dort wieder zurück.

Request- und Response sind durch das Gateway definiert, die resultierende Struktur ist entsprechend für alle Content-Services prinzipiell identisch.

Zur Vermeidung von Code-Redundanzen und Erleichterung des „Kick-Off“ eines neuen Content-Services sollen die Common Funktionen und Entitäten in ein Framework gegossen werden

Initiale Schätzung	1
Technologien	<ul style="list-style-type: none"> <li>* Javascript</li> <li>* Typescript</li> <li>* Dockerfile</li> </ul>
Abhängigkeiten	* BOT-37
Anforderungen	<ul style="list-style-type: none"> <li>* Das Framework implementiert eine NodeJS Express REST-API, äquivalent zu den existierenden Content-Services</li> <li>* Das Framework lässt sich in NodeJS Anwendungen via Dependency-Management einbinden (npm/yarn)</li> <li>* Das Framework abstrahiert den (Express) Server und dessen Routing, so dass ein Content-Services nur noch die Response implementieren muss</li> <li>* Das Framework implementiert die Schnittstelle zum Datenbankservice um a) Userdaten zu speichern/abzurufen und b) eigene Daten zu speichern und abzurufen</li> <li>* Das Framework füllt die „debug-history“ der Requests so, dass ein Service dieses Feature zwangsweise implementiert / nutzt</li> <li>* Requests, Responses, User, Rasa-Intents und ggf. weitere Entitäten werden durch das Framework als typisierte (typescript) Objekte definiert</li> <li>* Das Framework wird in allen bestehenden und geplanten Content-Services implementiert: Wetter, Mensa, Reminder</li> <li>* Die Nutzung des Frameworks ist verständlich dokumentiert</li> <li>* Die Contribution wird mittels Linting, Dokumentation und Build-Skripts erleichtert</li> <li>* Das Framework nutzt Types aus der (noch zu entwickelnde) BHT-Bot Library um Redundanzen zwischen Client-Bibliothek und Service-Framework zu vermeiden</li> </ul>

## Tasks

- \* BOT-44 Common Code, Features, Entitäten identifizieren → Use Case ableiten
- \* BOT-45 Typisiertes Javascript Framework erstellen
- \* BOT-46 Framework einbinden in Weather Service
- \* BOT-47 Framework einbinden in Mensa Service
- \* BOT-48 Framework einbinden in Reminder Service
- \* BOT-108 Framework dokumentieren
- \* BOT-117 Request History implementieren - Nutzung enforce

Nutzungshinweis: Auf dieses vorliegende Schulungs- oder Beratungsdokument (ggf.) erlangt der Mandant vertragsgemäß ein nicht ausschließliches, dauerhaftes, unbeschränktes, unwiderrufliches und nicht übertragbares Nutzungsrecht. Eine hierüber hinausgehende, nicht zuvor durch *datenschutz-maximum* bewilligte Nutzung ist verboten und wird urheberrechtlich verfolgt.