BOT-30: Chatbot Library: Vereinheitlichung der Kommunikation von Javascript Chatbots mit dem Gateway

Problem: Derzeit muss jede Anwendung, die den BHT-Bot als Chatbot implementieren möchte selbst implementieren, wie die Kommunikation zwischen Anwendung und Gateway aussieht, als auch die Schnittstellen Parameter in Anfrage und Antwort.
Um diese Implementationsredundanz zu verhindern, wird die Kommunikation und Typedefinitionen in einer zentralen Javascript Bibliothek zusammengefasst.
Dies ermöglicht auch weitere geplante Funktionalitäten (wie der asymmetrische Kommnuikationskanal zur Requestunabhängigen Server → Client Kommunikation) zentral entwickelt und mittels Dependency Management schnell in die ChatClients überführt werden.

Initiale Schätzung	1
Technologien	* Javascript * Typescript
Abhängigkeiten	keine
Anforderungen	* Die Library lässt sich in Node und Browser Javascript einbinden * Die Library nutzt semantische Versionierung zur Ermöglichung von Non-Breaking-Updates * Die fertige Library lässt sich via Dependency-Management (npm/yarn/webpack) userseitig einbinden und updaten * Die Library enthält typisierte (typescript) Entitäten für Common Request und Response Format(e) * Die Library enthält Unit-Tests für essentielle Funktionen und Typen * Die Library ist dokumentiert, sowohl was Nutzung, als auch Contribution angeht * Die Library verbessert die Collaboration mittels Linting-Regeln und Workflow-Scripten
Tasks	* BOT-33 Library Usage Dokumentieren * BOT-34 Library in Discord Bot integrieren * BOT-35 Library in Telegram Bot integrieren * BOT-36 Library in Website integrieren * BOT-31 Common Funktionalität / Use Cases identifizieren * BOT-32 Typescript Library für Bot erstellen

BOT-37: Discord Integration des BHT-Bot

Discord ist eine weit verbreitete Kommunikatinsplattform, auf der Nutzer sich in „Servern“ vernetzen und dort meist thematisch organisiert kommunizieren können. Die Projektgruppe des WS2020 ist selbst Teil der Zielgruppe des Discord-Messengers, wodurch sich diesese Plattform besonders eignet um einen weiteren Chatservice (neben Telegram) an den BHT-Bot anzubinden.
Eine Implementation des Chatbots innerhalb der Discord-Struktur steigert somit zum Einen die Verbreitung(smöglichkeit) des BHT-Bot und bietet gleichzeitig eine gute Möglichkeit Debug-Bot-Instanzen im präferierten Messenger zu betreiben.

Initiale Schätzung	2
Technologien	* Javascript * Docker

Abhängigkeiten	* BOT-30
Anforderungen	* Der Discourse Bot benutzt die zu entwickelnde zentralisierte Library zur Gateway Kommunikation um Coderedundanz mit dem Telegram Bot zu verhindern * Der Discourse Bot kann (direkte) Nutzer-Nachrichten mittels Gateway verarbeiten und antwortet dem User entsprechend * Wenn keine Verbindung mit dem Gateway besteht oder Fehler bei Anfragen auftreten reagiert der Chatbot durch Präsentation einer hilfreichen Fehlermeldung * Der Discourse Bot wird äquivalent zum Telgram Bot in das BHT-Bot Universum mittels Docker + compose integriert * Der Discourse Bot ist nicht Teil des BHT Bot, er wird als paralleler, unabhängiger Service betrieben * Alle Credentials und Urls/Ports werden aus dem Environment bezogen, es gibt keine hard-coded Referenzen zu Strukturen des BHT-Bot Gateways
Tasks	* BOT-38 NodeJS Chatbot erstellen * BOT-39 Docker Container + Compose für Container erstellen * BOT-40 Bot Usage dokumentieren * BOT-41 Bot Account anlegen für release (https://discord.com/developers/applications) * BOT-42 Bot Container in Beuth-Docker-Netzwerk einbinden (release)

BOT-XXX: EPIC_TITLE

EPIC_DESCRIPTION

Initiale Schätzung	TIME
Technologien	* <Programmiersprache 1> * <Containerisierung 1> * <Bot Servie 1>
Abhängigkeiten	* <Ticket ID1> * <Ticket ID2>
Anforderungen	* <Anforderung 1> * <Anforderung 2>
Tasks	* <Task1> * <Task2>

BOT-XXX: EPIC_TITLE

EPIC_DESCRIPTION

Initiale Schätzung	TIME
Technologien	* <Programmiersprache 1> * <Containerisierung 1> * <Bot Servie 1>

Abhängigkeiten
* <Ticket ID1>
* <Ticket ID2>

Anforderungen
* <Anforderung 1>
* <Anforderung 2>

Tasks
* <Task1>
* <Task2>

BOT-XXX: EPIC_TITLE

EPIC_DESCRIPTION

Initiale Schätzung TIME

Technologien
* <Programmiersprache 1>
* <Containerisierung 1>
* <Bot Servie 1>

Abhängigkeiten
* <Ticket ID1>
* <Ticket ID2>

Anforderungen
* <Anforderung 1>
* <Anforderung 2>

Tasks
* <Task1>
* <Task2>

BOT-XXX: EPIC_TITLE

EPIC_DESCRIPTION

Initiale Schätzung TIME

Technologien
* <Programmiersprache 1>
* <Containerisierung 1>
* <Bot Servie 1>

Abhängigkeiten
* <Ticket ID1>
* <Ticket ID2>

Anforderungen
* <Anforderung 1>
* <Anforderung 2>

Tasks
* <Task1>
* <Task2>

BOT-XXX: EPIC_TITLE

EPIC_DESCRIPTION

Initiale Schätzung TIME

Technologien
* <Programmiersprache 1>
* <Containerisierung 1>
* <Bot Servie 1>

Abhängigkeiten	* <Ticket ID1> * <Ticket ID2>
Anforderungen	* <Anforderung 1> * <Anforderung 2>
Tasks	* <Task1> * <Task2>