ChatAlly
Werkzeugkasten für selbst-gehostete Chat-Services

Schlussbericht

Zuwendungsempfänger:

Christian Fuß

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS24S03 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Chatbots sind ein niedrigschwelliges Informationsangebot, das für viele Zwecke eingesetzt werden kann. Neue Nutzer:innen oder jene, die das Angebot nicht häufig benutzen, können von einer klaren Struktur profitieren, gerade wenn das Angebot über gängige Messenger-Clients, wie WhatsApp oder Signal angeboten wird.

Obwohl es auf den ersten Blick Open-Source oder freie Plattformen zu geben scheint, mit denen man einfach Chatbots erstellen kann, sind die freien Kontingente stark begrenzt, und Open-Source bezieht sich meist nur auf das Software Development Kit (SDK). Die Datenkontrolle liegt in der Regel bei den Plattformanbietern. Es gibt zwar einige FOSS-Frameworks und -Bibliotheken für Chatbots, jedoch sind sie komplex in der Einrichtung und unterstützen nicht die Integration mit gängigen Messenger-Clients. ChatAlly bietet ein JavaScript-Framework das Web-Application-Frameworks wie Express.js, Fastify oder Koa ähnelt und das einfache Self-Hosting von Chat-Anwendungen ermöglicht. Die Erstellung der Software sollte in den folgenden Meilensteinen erfolgen:

- Modulare Architektur mitsamt Schnittstellen definieren: Die Schichten der Architektur sind klar beschrieben, die Schnittstellen so definiert, dass sie eine einfache Integration mit bestehenden Frameworks möglich machen, alles ist auf der Website veröffentlicht
- Vervollständigung und Dokumentation des WhatsApp-Cloud-Connectors
- Integration der ersten Module (Konnektor, NLU, Logik) mit der orchestrierenden Middleware als erstes Beispielszenario
- Implementierung und Dokumentation des Signal-Connectors
- Integration und Dokumentation von weiteren Beispielszenarien/Templates
- Implementierung und Dokumentation des WhatsApp-Web-Connectors: Für bestimmte Anwendungsarten ist die Anbindung an WhatsApp über den WhatsApp-Web-Client günstiger als die Cloud API

Beitrag des Projektes zu den Zielen der Förderinitiative „Software-Sprint“

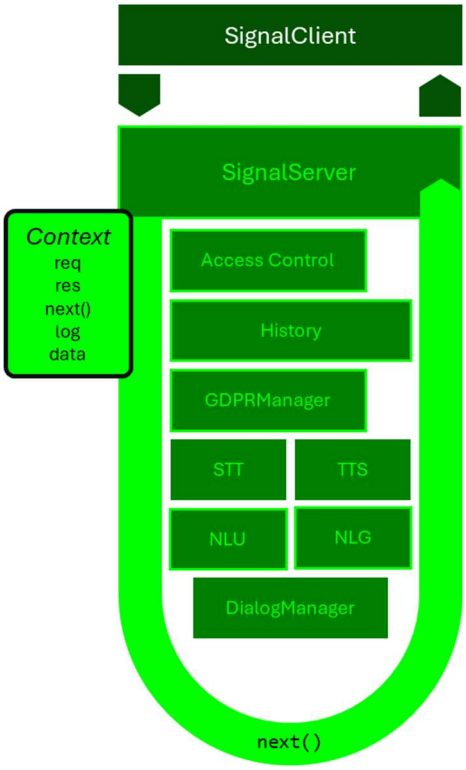
ChatAlly ermöglicht kleinen Organisationen und Communities, kostengünstig und DSGVO-konform, ihre eigenen Chatbots bereitzustellen. Gerade die Implementierung in JavaScript öffnet das Feld der Chatbots für die große Community von Web-Entwickler:innen, von denen einige gemeinnützige Organisationen oder Vereine unterstützen. Auch gibt es im Rahmen des Software-Sprint immer viele Projekte, die einen Chatbot für ihr Anliegen erstellen wollen, deren Umsetzung sich mit ChatAlly deutlich vereinfachen lässt.

Ausführliche Darstellung der Ergebnisse

Im Rahmen der Förderung wurde ein minimales und flexibles node.js-Framework für Chat-Applikationen entwickelt. Das Framework besteht aus verschiedenen Modulen, die bei npmjs.com unter dem Prefix `@chatally` veröffentlicht wurden. Der Kern ist aufgeteilt auf die Pakete `core`, `logger` und `utils` und kann durch sogenannte *Server* und *Middleware* ausgebaut werden.

Zentral im Paket `core` ist die Klasse `Application`, die die synchrone oder asynchrone Middleware koordiniert, Fehlerbehandlung und Logging bietet. Das Paket `logger` definiert eine Fassade zum Logging, die einfach durch gängige Frameworks implementiert werden kann. Das Paket bietet aber auch eine einfache Basisimplementierung für die Konsole. Das Paket `utils` veröffentlicht einige Funktionen, die auch bei der Implementierung von Middleware durch andere Entwickler:innen nützlich sein können.

Server stellen die Verbindung zu Chat-Clients, z.B. WhatsApp, Signal oder Websockets her, während Middleware den eingehenden `ChatRequest` interpretiert und eine `ChatResponse` erzeugt, die dann wieder vom Server an den Messenger-Client übermittelt wird. Im Kern wird ein einheitliches Datenmodell für Chat-Messages verwendet, so dass Chatbots einfach und gleichzeitig mit verschiedenen Clients kommunizieren können. Bisher veröffentlicht wurde ein Server für die Konsole, der eine einfache Möglichkeit zum Testen während der Entwicklung darstellt (NPM-Modul `console`) und ein Server für die WhatsApp-Cloud-API, der als Proof-of-Concept fast vollumfänglich das WhatsApp Feature-Set implementiert (NPM-Modul `whatsapp-cloud`).



Die nebenstehende Abbildung veranschaulicht an einem Beispiel, wie die Applikation das Request/Response-Paar eines Servers entgegennimmt, in einen `Context` integriert und das Handling durch die Middleware koordiniert. Middleware-Module werden dabei in der Reihenfolge ihrer Registrierung aufgerufen, können den Kontext anreichern, Hilfsmethoden daraus benutzen und die Response oder einen Teil davon schreiben. Mit einem asynchronen Aufruf der Methode `await next()` können Middleware-Module ihre Verarbeitung oder einen Teil davon auch auf das Downstream-Leg verschieben. Beispielhaft wurde das Paket `consent-manager` als Middleware veröffentlicht, das es den Entwickler:innen vereinfacht, eine

Einwilligung in die Datenverarbeitung einzuholen. Darüber hinaus wurde das Paket `nlpjs` veröffentlicht, das eine einfach zu nutzende Bibliothek für das *Natural Language Processing* in ein Middleware-Modul verpackt. Dies erlaubt einen schnellen Einstieg in die Entwicklung natürlich wirkender Chatbot-Dialoge.

Die Implementierung wurde weitgehend entsprechend der definierten Meilensteine durchgeführt. Aufgrund von Verzögerungen beim geplanten Signal-Connector wurden die Beispielszenarien vorgezogen. Sowohl die Schnittstellen als auch die Szenarien wurden immer zeitgleich mit der Umsetzung dokumentiert und die vereinheitlichte Gesamtdokumentation unter https://chatally.org veröffentlicht.

Während der Projektarbeit war das Coaching durch die Open Knowledge Foundation und beauftragte Partner insbesondere in Hinblick auf Community Building und Networking sehr hilfreich. Jedoch konnten leider aufgrund der hohen Arbeitslast durch die Implementierung in einem Solo-Projekt nicht alle Hinweise und Schritte umgesetzt werden. Hier soll die Zielgruppe der Web-Entwickler:innen, die Projekte mit Gemeinwohlorientierung unterstützen im Nachgang noch mehr über die Möglichkeiten mit ChatAlly informiert werden.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Es haben sich schon einige interessierte Organisationen gemeldet, die ChatAlly für unterschiedlichste Anwendungen einsetzen wollen. Bei Sportvereinen hat die Moderation von Gruppenchats eine hohe Priorität, andere Vereine mit öffentlichen Veranstaltungen wollen diese gerne abfragbar machen oder sogar Tickets per Chat verkaufen. Ein Makerspace möchte den Chatbot für Hausautomatisierung einsetzen, z.B. als fernsteuerbaren Türöffner. Alle diese Anwendungen lassen sich mit entsprechend angepasster Middleware mit ChatAlly umsetzen.

Die App und Website https://dooiy.org der gemeinnützigen Organisation Hack Your Shack gGmbH, aus deren Anforderungen die Idee für das Framework ChatAlly entstand, wird die Ergebnisse nutzen, um aus dem bisherigen Prototyp ein umfassendes Chatbot-Angebot, insbesondere für Erstnutzer:innen zu entwickeln.

Neben der Unterstützung solcher Early Adopters ist die Weiterentwicklung der begonnenen Anbindung an Signal geplant. Außerdem soll die alternative Anbindung per `whatsapp-web.js` an WhatsApp sowie Anbindungen an Matrix und Telegram ergänzt werden. Auch eine bislang nur intern genutzte Anbindung von generischen Clients, z.B. Web oder native Mobile-Apps per Websockets soll noch veröffentlicht werden.

Die Arbeit in dem Projekt hat es mir erlaubt, mich vertieft mit Web-Application-Frameworks und anderen asynchronen Middleware-Frameworks auseinanderzusetzen und die dahinterliegenden Architekturmuster zu analysieren und auch eigene beizutragen.

Neben diesen technischen Erkenntnissen hat insbesondere die Vollzeitarbeit an einem Open-Source-Projekt zu vielen neuen Erkenntnissen geführt, z.B. durch das Coaching der Open Knowledge Foundation zu Business Modellen oder zum Requirements Engineering mit unbekannten oder noch nicht existierenden Communities.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Bei der Projektplanung war ich noch davon ausgegangen, dass ein bestehendes Middleware-Framework wie z.B. Jovo zum Einsatz kommen sollte. Bei genauerer Analyse vor der Umsetzung zeigte sich aber, dass die strategische Ausrichtung dieses Frameworks und auch anderer auf Home Assistant Systeme eine für unsere Zwecke unnötige Komplexität bedingen. So wurden zwar einige Architekturmuster und geschickte Lösungsansätze, z.B. für asynchrone Middleware-Module übernommen aber am Ende selbst implementiert.

Auch konnte die Implementierung des Servers zur Anbindung von Signal-Clients im Zeitrahmen des Projekts nicht abgeschlossen werden. Die Implementierung unterstützt zum Förderende lediglich direkte Textnachrichten und noch keine Medien oder Gruppenmoderation. Die Quellen des anzuschließenden Signal-Messengers (Clients und Server) sind zwar Open-Source, allerdings undokumentiert und nicht semantisch versioniert, da sie von Signal lediglich zu Security-Verifikationszwecken veröffentlicht werden. Die mangelnde Dokumentation hat die Anbindung erheblich erschwert, so dass die Umsetzung deutlich länger gedauert hat. Die Implementierung wird aber auch nach Förderende fortgesetzt. Der Server zur Anbindung mittels WhatsApp-Web ist entsprechend auch bis nach der Förderlaufzeit verzögert. Die Implementierung wird aber auch nach Ende des Förderzeitraums fortgesetzt.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Das Projekt ist ausführlich auf der Website https://chatally.org dokumentiert, dort finden sich auch alle aktuellen Kontaktmöglichkeiten (Email, Chat, Discord). Die Quellen des Frameworks und der Beispielszenarien wurden auf Github veröffentlicht: https://github.com/chatally. Die installierbaren NPM-Module findet man unter https://www.npmjs.com/search?q=chatally.

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Die Arbeits- und Kostenplanung wurde im Groben eingehalten. In der Planung wurde bereits berücksichtigt, dass es unter Umständen bei einigen Paketen zu Verzögerungen kommen könnte, entsprechend wurden die Arbeitspakete mit niedrigerer Priorität weiter nach hinten gestellt.

Aufgrund des Mehraufwands bei der Implementierung der Signalanbindung konnte das Paket nicht zu der geplanten Reife umgesetzt werden und die zweite alternative Anbindung von WhatsApp, per inoffizieller whatsapp-web.js Bibliothek wurde nicht im Rahmen des Förderzeitraums begonnen.

Diese Pakete hatten jedoch eine niedrigere Priorität, so dass die Nichteinhaltung der Arbeitsplanung durch die Verzögerungen nicht die allgemeine Nutzbarkeit der Ergebnisse gefährdet haben.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Es gab in der Zwischenzeit keine unerwarteten Entwicklungen an anderen Stellen, die maßgeblichen Einfluss auf meine Arbeiten oder die Zielsetzung hatten. In den früh durchgeführten Nutzungstests mit Nutzer:innen der Chatbots zeigte sich, dass bei der Einbindung von komplexeren Inhalten oder Abläufen eine nahtlose Verknüpfung von Messenger-Clients und Web hilfreich ist, dies wird von der Implementierung auf einfache Art und Weise durch speziell formatierte Hyperlinks unterstützt.