3-Schichten-Modell Sprachausgabe (DRAFT)

Einleitung

Persönlicher Prolog Christian Fenzl

Ich erlaube mir, meine Einleitung in persönlicher Form, also nicht in informativem Charakter, zu verfassen. (cf)

Ich habe in den letzten Jahren zwei verschiedene Sprachausgabe-Engines für mehrere Haushalte implementiert. Dabei offenbarte sich mit meinen Implementierungen folgendes Problem, dass bei einem Projekt wie LoxBerry noch viel schwerer wiegt: Unterschiedliche Leute möchten unterschiedliche Sprachausgaben auf unterschiedlichen Ausgabesystemen. Das hatte zur Folge, dass für jede Installation erneut Anpassungen notwendig waren.

Für die Neuentwicklung einer Sprachausgabe - die es für LoxBerry früher oder später geben wird, soll das gleich richtig angegangen werden - mit einem Drei-Schichten-Modell für die Sprachausgabe. Hier mein DRAFT zur Diskussion, Erweiterung und Spezifikation.

Die drei Schichten

Layer 1 - Die TTS-Engine

Die TTS-Engine ist für die Erzeugung der Sprache aus Text verantwortlich. Die Engine liefert als Ergebnis fertige Audio-Dateien (z.B. MP3, WAV, etc.). Je nach Bedarf können unterschiedliche TTS-Engines entwickelt werden, beispielsweise eine Cloud-basierte Lösung (TTS wird von einem Anbieter online erzeugt), oder eine lokal installierte TTS-Engine. Ergibt es sich, dass eine TTS-Engine nicht mehr verfügbar ist (z.B. Änderung des Lizenzmodells eines Cloud-Anbieters), wird auf eine andere TTS-Engine umgestellt.

Input: Text

Output: Audio-Datei

Konfiguration: Spezifisch für die TTS-Engine (z.B. Tokens, Sprache. Stimme, usw.)

Layer 2 - Die REST API

Das Herz der Sprachausgabe ist die REST-API. Sie führt die TTS-Engine und die Ausgabe-Engine zusammen, und stellt zudem ein HTTP REST-Interface z.B. für Loxone zur Verfügung.

Zudem ist die REST API für die Optimierung der Sprachausgabe zuständig, z.B. für das Caching und

Queuing von Ansagen.

Die REST-API soll mehr als nur einen Text als Eingabe-Parameter ermöglichen:

Pflicht: Text

Optional: Ziel (Liste; Namen der Zonen oder ALLE); Unterbrechend Ja/Nein, Ausgabeengine (wenn nicht Standard-Engine); Ausgabe-Engine-spezifische Parameter (wenn nicht Standard)

Konfiguration: Liste mit Zone ↔ Ausgabe-Engine + Parameter

Das Caching soll bereits berechnete Texte ohne Aufruf der TTS-Engine vorhalten.

Das Queuing ist dafür verantwortlich, dass Texte an die Ausgabe-Engines verteilt werden, und innerhalb einer Ausgabe-Engine nicht gleichzeitig abgespielt werden. Mit dem Parameter Unterbrechend Ja/Nein soll gesteuert werden, ob eine neue Ansage vorhergehende Ansagen dieser Zone verwirft und laufende Ansagen abbricht. Ggf. muss die Ausgabe-Engine definieren, ob sie überhaupt abbrechen kann.

Eine Ausgabe-Engine kann auch auf einem entfernten System (anderer LoxBerry) laufen. Die Art der Kommunikation ist in der Konfiguration zu definieren.

Layer 3 - Die Ausgabe-Engine

Die Ausgabe-Engine ist verantwortlich für die Wiedergabe der synthetisierten Stimme auf dem jeweiligen Audio-System. Dies könnte ein Logitech Media Server sein, ein Denon Heos oder ein Sonos-System, aber auch die lokale Ausgabe wäre als Ausgabe-Engine möglich. Die gewünschte Ausgabe-Engine (oder Engines) wird installiert und konfiguriert.

Input: Audio-Datei; Ziel, Unterbrechend Ja/Nein, Engine-spezifische Parameter

Output: Fertig oder Fehler

Konfiguration: Spezifisch für die Ausgabe-Engine (z.B. IP-Adressen, Zonen usw.)

Möglicher Aufbau der API

Alle drei Layers sollten der Einfachheit halber zumindest mittels HTTP oder einem TCP-Socket (CLI) aufgerufen und der Request mit dem jeweiligen Output beantwortet werden. Zu überlegen ist auch, ob ein lokaler Datentransfer per RAMDISK erfolgen soll (wobei dies die Funktionalität auf ein lokales System einschränkt).

Zu definieren ist:

- Die REST-Kommandos für den Aufruf der Sprachausgabe, z.B. /tts.cgi?say=Hallo wie gehts&zone=Wohnzimmer
- Die Schnittstelle zwischen REST API und der TTS-Engine, inkl. Format der Rückgabe

- Die Schnittstelle zwischen REST API und Ausgabe-Engine, inkl. Format der Rückgabe.

Vor- und Nachteile

Nachteile

- Bestehende ad-hoc Implementierungen funktionieren nicht und müssen geändert und aufgeteilt werden.
- Schnittstellen zwischen TTS und Ausgabe erschweren die Entwicklung
- Performance-Reduktion
- Mehr Fehlerquellen

Vorteile

- Unterschiedliche TTS-Engines können mit unterschiedlichen Ausgabegeräten gekoppelt werden.
- Sprachausgabe wird damit unabhängiger
- Die Entwicklung einer TTS-Engine und einer Ausgabe-Engine als jeweils ein eigenständiges Projekt ist überschaubarer
- EINE zentrale Schnittstelle für Sprachausgabe, nicht viele kleine Insellösungen die nur in bestimmten Konstellation funktionieren
- Mehrere unterschiedliche Ausgabe-Engines in einer Infrastruktur möglich

Für Entwickler

Wenn du gerade eine Sprachausgabe machst, kann man mit diesem Modell natürlich noch nichts konkret umsetzen.

Achte beim Entwickeln auf einen modularen Aufbau auf Basis der drei Teile. Dann ist ein späterer Umbau auf dieses Konzept relativ einfach.