Testresultate PSE

Übersicht

Dieses Dokument fasst die Ergebnisse der Tests zusammen, die zur Qualitätssicherung unserer Übersetzungsbibliothek für Ruby on Rails-Anwendungen durchgeführt wurden. Ziel war es, die Funktionalität, Robustheit und Benutzerfreundlichkeit der Bibliothek zu gewährleisten. Das Testkonzept bestand aus Unit-Tests, Integrationstests und einem Usability-Test.

---

1. Unit-Tests

Die Unit-Tests wurden mit dem Framework RSpec durchgeführt. Ziel dieser Tests war es, einzelne Methoden und Module isoliert zu testen, um sicherzustellen, dass sie wie erwartet funktionieren.

Getestete Komponenten:

- FixmeFinder
- LocaleTranslationService
- LocalWriter
- ArAttributesTranslationService
- ArAttributesWriter
- OpenaiModel
- MissingArAttributesFinder
- InteractionHelper

Beispielhafte Resultate:

- FixmeFinder: Korrektes Auslesen der #FIXME-Kommentare in YAML-Dateien wurde verifiziert.
- ArAttributesWriter: Sicherstellung, dass die Übersetzungen korrekt in die Datenbank geschrieben werden.
- OpenaiModel: Methoden wie translate und language_from_filename wurden auf korrekte Funktionsweise getestet.

Alle Unit-Tests (mit Mocks, wo notwendig) wurden erfolgreich bestanden.

---

2. Integrationstests

Da nur eine begrenzte Anzahl von API-Aufrufen mit dem Sprachmodell zur Verfügung stand, konnten keine End-to-End-Tests mit echten Antworten durchgeführt werden. Stattdessen wurden Mocks verwendet, um die Antworten des Modells zu simulieren.

Vorgehen:

- Verwendung von allow(...).to receive(...).and_return(...), um API-Antworten zu simulieren
- Tests der Interaktion zwischen Modulen: Prompt-Generierung, Antwortverarbeitung, Datenbankaktualisierung

Resultate:

- Integrationstests wie ArTranslationExporter.run_interactive haben gezeigt, dass der gesamte Workflow korrekt abläuft
- Ein weiterer Integrationstest auf TranslationGemE.translate_fields hat gezeigt, dass das logische Verhalten der Methode korrekt ist – d. h. die Felder werden richtig gefiltert, in einen Prompt umgewandelt, an das Modell übergeben und die Antworten korrekt zurückübersetzt.
- Auch Benutzerinteraktionen über die Konsole (z. B. stdin) wurden simuliert

Auch diese Tests verliefen erfolgreich und zeigten keine strukturellen Fehler im Zusammenspiel der Komponenten.

---

3. Usability-Tests

Ziel war es, die Benutzerfreundlichkeit für Entwickler sicherzustellen, die das Gem in ihre Projekte integrieren.

Durchgeführte Tests:

- Installation und Ausführung des Gems in der von Eonum bereitgestellten Beispiel-Applikation
- Konfiguration und Test der Übersetzungsfunktionen über Rake-Tasks

Erkenntnisse:

- Die Installation auf Windows war nur eingeschränkt möglich, da Rails eine POSIX-kompatible Umgebung benötigt. Die Lösung war die Verwendung von WSL (Windows Subsystem for Linux).
- Das Setup unter WSL funktionierte reibungslos.

Zusätzlich zu unseren internen Usability-Tests wurde unsere Übersetzungs-Bibliothek auch von Mitarbeitenden der Firma Eonum getestet. Dabei stand nicht der Quellcode, sondern die Nutzbarkeit und Verständlichkeit der Dokumentation im Vordergrund – insbesondere des Benutzerhandbuchs (README). Eonum hat uns wertvolles Feedback gegeben, wie wir das README klarer und strukturierter gestalten können, z. B. durch die bessere Hervorhebung von Installationsschritten oder die ausführlichere Beschreibung der Konfigurationsoptionen. Diese Hinweise wurden umgesetzt, um die Einstiegshürde für externe Entwickler weiter zu senken.

Bemerkung

Im ursprünglichen Testkonzept wurden auch Stress-Tests und Datenbank-Tests in Erwägung gezogen. Letztlich haben wir uns jedoch entschieden, auf deren Durchführung zu verzichten. Stress-Tests wären wenig aussagekräftig gewesen, da wir aufgrund der begrenzten Anzahl an API-Calls ohnehin eingeschränkt waren. Zudem handelte es sich bei der zur Verfügung gestellten Datenbank um eine von Eonum bereitgestellte Beispieldatenbank. Daher kamen wir zum Schluss, dass es keinen sinnvollen Mehrwert bringen würde, diese zu testen.

Fazit

Alle automatisierten Tests wurden erfolgreich durchgeführt, wobei Mocks für das Sprachmodell verwendet wurden, um Ressourcen zu sparen. Die Usability wurde innerhalb der Gruppe, und dann noch von Entwickler:innen der Firma Eonum, getestet. Die Tests haben gezeigt, dass die Bibliothek funktional stabil ist und korrekt mit den verschiedenen Komponenten zusammenarbeitet. Dank dieser Tests konnten wir die Lesbarkeit des Readme verbessern und so die Installation für Entwickler, die unser Gem verwenden möchten, vereinfachen. Die Strukturierung in Unit- und Integrations-Tests hat sich als effektiv erwiesen, um die Qualität der Software umfassend abzusichern.