Testkonzept V2

Testkonzept und Testresultate

1. Unit Tests

Ja, wir setzen Unit Tests ein, da sie essenziell sind, um einzelne Komponenten isoliert zu testen.

Umfang:

Wir schreiben Unit Tests für die Kernfunktionen des Ruby-Gems, insbesondere für die Übersetzungslogik und den KI-Wrapper.

Idee: RSpec, ein beliebtes Test-Framework für Ruby.

2. Datenbank Tests

Wir benutzen die Datenbank, die uns von Eonum bereitgestellt wird (Active Records) und der Inhalt der Datenbank wird also übersetzt. Das heisst, alle Tests betreffen die Datenbank.

3. Integrationstest

Test von User Stories:

Automatisierte Übersetzung eines fehlenden Interface-Textes

→ Funktioniert die Übersetzung korrekt?

Automatisierte Übersetzung von Active-Record-Modelltexte:

→ Wird der Text korrekt gespeichert?

Spezial- & Grenzfälle:

Sonderzeichen, z. B. é, ö, ç, ß

Sehr lange Texte oder extrem kurze Texte

4. Installationstest

Testsysteme:

Lokale Installation mit Ruby 3.1 auf macOS und Linux

Installation innerhalb einer Ruby-on-Rails-Anwendung

Ziel: Sicherstellen, dass die Installation mit den Anweisungen ohne Probleme funktioniert.

5. GUI-Test

Unser Gem hat keine eigene GUI, daher keine automatisierten GUI-Tests nötig.

6. Stress-Test

Muss die Software hohen Belastungen standhalten?

Ja, wenn viele parallele Übersetzungsanfragen kommen.

Teststrategie:

Simulation von 2-3 gleichzeitigen Übersetzungen von YML Dateien

Performance-Messung der API-Antwortzeit

Mögliches Tool dafür: JMeter (von Chat GPT empfohlen)

7. Usability-Test

Wer sind die Nutzer?

Entwickler, die Ruby-on-Rails-Projekte mit automatisierter Übersetzung nutzen wollen.

Testpersonen:

Entwickler aus unserem Team

Testprozess:

Geben wir den Testpersonen eine Aufgabe („Installiere das Gem und übersetze einen Text“)

Beobachten wir, wo es Schwierigkeiten gibt (z. B. Installationsprobleme, unklare Fehlermeldungen, usw).