Testkonzept V2

Version: 2.0

Teamname: Projektteam Abilium

Mitglieder*innen: Carolina Lucca, Yannis Racine, Dominic Kronig, Myroslav Pavlov, Linus Marti, Livia Brunner

Datum: April 2025

1. Vorwort: Ziel und Testfokus

Dieses Testkonzept dient der systematischen Überprüfung des Projekts *Raumstatusanzeige mit Odoo und Raspberry Pi*. Der Fokus liegt in dieser Phase auf Robustheit und Funktionssicherheit – insbesondere in der Kommunikation zwischen Odoo und dem Raspberry Pi.

Ein vollständiges automatisiertes Test-Coverage (100 %) ist aktuell nicht Ziel, sondern eine verlässliche, manuell verifizierbare Funktionalität bei typischen sowie fehleranfälligen Anwendungsfällen.

Contents

---

2. Unit Tests

2.1 Odoo-Modul

2.1.1 Models & Views

- Neue Models müssen folgende Tests bestehen:
  - Odoo-Server muss nach dem Zufügen von Model in __init__.py starten können
  - Wenn es eine View-Datei gibt, die auf das Model zugreifen kann, d.h. wenn die Interaktion mit dem Model über eine GUI möglich ist:
    - Neu erstellte GUI muss visuell den Erwartungen entsprechen (Mock-ups o.ä.)
    - Die Funktionalität des Models ist ggf. über die GUI zu testen:
      - Neue Records (z.B. ein neues Zimmer) sollen gespeichert werden können
      - nur sinnvolle Angaben sollen akzeptiert werden (z.B. Kapazität des Zimmers ist immer eine positive ganze Zahl)
      - automatisierte Felder des Models sollen ihre Daten sinnvoll anzeigen
  - Wenn keine View-Datei vorhanden ist, das Model aber unbedingt auf Funktionalität getestet werden muss, kann eine einfache View-Datei erstellt werden, die alle Daten des neuen Models und anderer Daten, die für den Test relevant sein könnten, visualisiert. Dabei ist das Aussehen der GUI zu vernachlässigen. Durch Interaktion mit der neu erstellten GUI sollte das Model, wie oben beschrieben, auf Sinnhaftigkeit und Funktionalität geprüft werden.

2.1.2 Verbindung von Odoo mit MQTT-Broker

- Auf einem privaten PC wird ein MQTT-Client erstellt, der sich mit demselben Broker wie Odoo und Raspberry Pi verbindet.

- Wenn erwartet wird, dass Odoo eine Nachricht über MQTT sendet, sollte diese Nachricht entweder direkt über «subscribe» zum entsprechenden Topic oder über Wildcard-Subscribe sofort empfangen werden.
    - Es wird getestet, ob die Nachricht im richtigen Topic landet und ob sie den Erwartungen entsprechend strukturiert ist.

2.1.3 Sonstiges

- Alle sonstigen Teile des Odoo-Moduls sollen folgende Tests bestehen:
    - Der Odoo-Server muss nach dem Einfügen neuer Komponenten ohne Fehler starten können.
    - Neue Funktionalitäten (z. B. Automatisierungen, Serveraktionen, neue Menüpunkte oder Einstellungen), die nicht unter Models oder MQTT fallen, werden manuell über die Benutzeroberfläche oder die Odoo-Einstellungen geprüft:
        - Es wird überprüft, ob die Funktionalität verfügbar und auffindbar ist (z. B. neuer Menüpunkt sichtbar).
        - Die erwartete Funktion wird durch Benutzerinteraktion (z. B. Klicks, Eingaben) nachvollzogen.
        - Dabei wird geprüft, ob die Funktion sich wie vorgesehen verhält (z. B. ob eine Serveraktion tatsächlich die Daten verändert oder eine gewünschte Meldung erscheint).
        - Falls keine GUI vorhanden ist, erfolgt der Test durch das gezielte Aufrufen der Funktion über die Entwicklertools (z. B. über technische Einstellungen im Backend oder durch Anlegen passender Datensätze).

2.2 Raspberry Pi

2.2.1 Datenbankzugriff

- Automatischer Empfang der Daten von MQTT-Broker.
- Ausgabe der Daten in der Konsole.

2.2.2 Visualisierung auf E-Ink-Display

- Visuelle Überprüfung der Display-Anzeige
    - Nach Mock-ups oder anderen Erwartungen
- Nutzung eines Raspberry Pi Zero WH mit 2.13 Zoll e-Paper HAT E-Ink Display.

---

3. Datenbanktests

- Automatisierter Abruf aus der Odoo-Datenbank:
    - **Abrufbare Daten:**
        - Zimmernummer
        - Anzahl Plätze
        - Liste aller Meetings in diesem Zimmer:
            - Zeitpunkt

        - Ersteller
        - Teilnehmerzahl
    - MQTT-Broker
        - Status: Verbunden/Nicht verbunden

---

4. Integrationstests

4.1 Odoo-Modul

4.1.1 Durchlauf mit GUI-Unterstützung im Admin-Modus

1. Erstellen von drei Sitzungszimmern:
    - «Testroom 1» Mit 2 Plätzen
    - «Testroom 2» Mit 4 Plätzen
    - «Testroom 3» Mit 1000 Plätzen
2. Zimmer bearbeiten:
    - «Testroom 3» auf «Testroooooooooooooooooooooom 3.2» mit 1000000 Plätzen überschreiben
    - «Testroooooooooooooooooooooom 3.2» löschen
3. Erstellen eines Meetings:
    - 1-5 Mitarbeiter hinzufügen und richtige Zimmerempfehlung überprüfen.
    - Keine Überschneidung von Meetings im gleichen Zimmer.
    - Zimmerkalender aufrufen und Angaben überprüfen.
4. Erstellen einer Verbindung mit MQTT-Broker:
    - Daten der Broker angeben und auf Verbindung warten.

4.1.2 Grenzfälle

- Sitzungszimmer mit einem leeren Namen erstellen versuchen
- Zwei Sitzungszimmern mit gleichen Namen erstellen versuchen
- Kapazität der Zimmer auf 0 und auf -1 setzen versuchen

4.1.3 Durchlauf mit GUI-Unterstützung im Nutzer-Modus (nicht Admin)

1. Es soll nicht möglich sein, ein Zimmer zu erstellen, bearbeiten oder löschen
2. Ein Meeting mit 2 anderen Mitarbeiter im Zimmer «Testroom 1» erstellen

4.2 Raspberry Pi mit E-Ink-Display

4.2.1 Durchlauf "von Hand"

1. Sitzungszimmer zuweisen und wechseln.

    - Ändern des Zimmernamens im Admin-Modus, wenn es zu Raspberry zugewiesen ist
2. Visuelle Überprüfung der Display-Anzeige.

4.2.2 Grenzfälle

- Stromausfall
- Fehlende Netzwerkverbindung und MQTT-Verbindung:
    - Sie muss im Feld «Letzte Aktualisierung» oder «Last Update» ablesbar sein.
- Überlange Anzeigedaten dürfen sich nicht mit anderen Daten überschneiden. Der Anfang muss sichtbar sein.

---

5. Installationstests

5.1 Odoo-Server-Schnittstelle

- Tests auf Linux Debian

5.2 Raspberry Pi

- Tests auf Raspberry Pi OS (32-bit).
- Hardware: Raspberry Pi Zero WH mit 2.13 Zoll e-Paper HAT E-Ink Display.

5.3 MQTT-Verbindung

- Test mit HiveMQ-Broker
- Test mit Mosquitto-Broker, falls erforderlich

---

6. GUI-Tests

- Durchführung "von Hand":
    - Die Benutzeroberfläche (z. B. Odoo-Webinterface oder Raspberry Pi-Display) wurde direkt von Testpersonen bedient.
    - Funktionen wurden über das Interface wie im echten Betrieb ausgeführt (z. B. Erstellen eines Meetings, Aufrufen des Kalenders, Anzeigen von Raumstatus).
- Es wurde geprüft:
    - Ob alle Elemente sichtbar und korrekt beschriftet sind.
    - Ob Eingaben möglich sind und erwartungsgemäss verarbeitet werden.
    - Ob bei typischen Abläufen keine Fehler auftreten (z. B. Meetings lassen sich korrekt speichern und anzeigen).
    - Ob die Benutzerführung logisch und verständlich ist

---

7. Stresstests

7.1 Odoo-Modul

- Für das Odoo-Modul ist ein Stresstest aus unseren Sicht wenig sinnvoll, da letztlich nur die Datenbank von Odoo selbst davon betroffen sein könnte.

7.2 MQTT-Broker und Raspberry Pi

- Eine Verbindung zwischen Odoo und Raspberry Pi wird hergestellt und ein Zimmer wird zugewiesen.
- Auf einem privaten PC wird ein MQTT-Client erstellt, der sich mit demselben Broker wie Odoo und Raspberry Pi verbindet.
- In einer Schleife werden 1000 nummerierte Zimmerumbenennungen im entsprechenden Topic durch MQTT veröffentlicht.
- In den Logs des Raspberry Pi sollen alle 1000 Zimmerumbenennungen angezeigt werden, wobei die 1000ste auch auf dem E-Ink-Display angezeigt werden soll.

---

8. Usability-Tests

8.1 Testpersonen

- Grundkenntnisse in Odoo-Modulen erforderlich.
- Keine Programmier- oder Datenbankkenntnisse nötig.
- Die künftigen Anwenderinnen der Software sind hauptsächlich Klienten von Abilium, die Sitzungszimmer über Odoo verwalten und buchen möchten. Sie können sich als Benutzer*in oder Admin einloggen.
- **Benutzer*innen**:
  Diese Gruppe nutzt das System, um Besprechungen zu planen und Räume zu buchen. Sie verfügen über grundlegende Computerkenntnisse und kennen sich mit Kalenderfunktionen oder webbasierten Tools aus. Kenntnisse in Odoo sind wünschenswert.
- **Admins**:
  Administrator*innen sind zusätzlich verantwortlich für das Erstellen, Bearbeiten und Löschen von Räumen. Sie haben vertieftere Kenntnisse in Odoo und sind oft in der IT oder Raumplanung tätig. Programmierkenntnisse sind nicht erforderlich, jedoch sollten sie sicher im Umgang mit webbasierten Verwaltungsoberflächen sein.
- Für die Usability-Tests werden repräsentative Personen beider Gruppen ausgewählt, die typische Anwendungsfälle durchspielen. Die Testpersonen werden gezielt unter Studierenden der Universität Bern gesucht, insbesondere aus dem Bereich Informatik, da diese mit ähnlichen Systemen vertraut sind und fundiertes Feedback geben können. Die Auswahl erfolgt nach Verfügbarkeit, Interesse und Grundkenntnissen in der Nutzung von webbasierten Anwendungen.

8.2 Ablauf

- Der Usability-Test findet in einem kontrollierten Rahmen statt und wird von einem Mitglied des Projektteams begleitet. Er umfasst folgende Schritte:

Einführung (ca. 10 Minuten):

- Kurze Einführung in das System.
- Erklärung der Rollen (Admin vs. Benutzer*in) und deren Rechte.
- Überblick über die wichtigsten Funktionen.

Aufgabenbasierter Test (ca. 30–45 Minuten):

Benutzer*innen-Aufgaben:

- Anmeldung im System.
- Erstellen einer Besprechung in einem bestehenden Raum.
- Überprüfung des Kalenders auf Terminüberschneidungen.

Admin-Aufgaben:

- Anlegen eines neuen Sitzungszimmers.
- Ändern und Löschen eines bestehenden Raumes.
- Erstellen eines Meetings als Admin zur Verifizierung der Admin-Funktionalität.

Feedback und Beobachtung (ca. 10–15 Minuten):

- Die Testperson äussert spontanes Feedback zur Bedienung.
- Beobachtungen zur Verständlichkeit, Navigation und Fehlerfreundlichkeit werden dokumentiert.

Mögliche Verbesserungsbereiche

- Zeitbedarf pro Aufgabe
- Anzahl auftretender Fehler
- Anzahl benötigter Hilfestellungen
- Ziel ist es, Schwachstellen im Interface oder in der Nutzerführung aufzudecken und daraus Verbesserungen abzuleiten – sowohl für Admins als auch für normale Benutzer*innen.
