PyQt GUI Dashboard der automatisierten Laborstraße - Technische Dokumentation

Ujwal Subedi & Wissam Alamareen

7. Februar 2024

Inhaltsverzeichnis

Kapitel 1

Einführung

1.0.1 Überblick über die Anwendung

Die Labortasse ist eine fortschrittliche, automatisierte Lösung, die darauf abzielt, Laborprozesse zu optimieren und zu vereinfachen. Diese Anwendung ist speziell für Labore entwickelt, die mit einer Vielzahl von Proben und Experimenten arbeiten. Sie bietet eine Benutzeroberfläche und eine Reihe von Funktionen, die den Workflow im Labor effizienter gestalten.

1.0.2 Zweck und Umfang

Die Labortasse wurde entwickelt, um den Anforderungen moderner Labore gerecht zu werden, die eine effiziente und genaue Verarbeitung einer großen Anzahl von Proben und Experimenten erfordern. Ihr Hauptzweck ist es, die Komplexität der Laborarbeit zu reduzieren, indem sie automatisierte Prozesse und eine Datenverwaltung bietet.

1.1 Einleitender Abschnitt: Welche Rolle liest was?

1.1.1 Willkommen, Entwickler!

Als Entwickler sind Sie für die Implementierung, Wartung und Fehlerbehebung der Software verantwortlich. Diese technische Dokumentation richtet sich an Entwickler, die mit der Software vertraut werden möchten und detaillierte Informationen zu deren Funktionsweise und Entwicklung benötigen. Lesen Sie die Anweisungen im Abschnitt 2.3.

Hier finden Sie die wichtigsten Abschnitte für Entwickler:

- **Installation:** Erfahren Sie, wie Sie die Software herunterladen, installieren und konfigurieren.
- **Programmierschnittstelle:** Entdecken Sie die Funktionen der Software, um eigene Anwendungen zu entwickeln.
- **Entwicklungstools:** Lernen Sie die verfügbaren Entwicklungstools kennen, um die Software zu testen und zu debuggen.

1.1.2 Willkommen, Systemadministrator!

Als Systemadministrator sind Sie für die Verwaltung und Bereitstellung der Software verantwortlich. Diese technische Dokumentation richtet sich an Systemadministratoren, die die Installation, Konfiguration, Wartung und Sicherung der Software durchführen müssen. Lesen Sie die Anweisungen im Abschnitt 2.1.2.

Hier finden Sie die wichtigsten Abschnitte für Systemadministratoren:

- **Systemanforderungen:** Informieren Sie sich über die Hardware- und Softwareanforderungen für den Betrieb der Software.
- **Installation und Konfiguration:** Erfahren Sie, wie Sie die Software auf Ihrem Rechner installieren und konfigurieren.
- **Wartung und Sicherung:** Lernen Sie die Verfahren für die Wartung, Behebung von Fehlern und Sicherung der Software kennen.

1.1.3 Willkommen, Laborassistent!

Als Laborassistent sind Sie für den Betrieb und die Nutzung der Software in einem Laborumgebung verantwortlich. Diese technische Dokumentation richtet sich an Laborassistenten, die die Software verstehen und bedienen müssen, um Experimente durchzuführen oder Datenerfassungsaufgaben zu erledigen. Lesen Sie die Anweisungen im Abschnitt 2.1.

Hier finden Sie die wichtigsten Abschnitte für Laborassistenten:

- **Benutzeroberfläche:** Lernen Sie die Benutzeroberfläche der Software kennen und wie Sie sie für Ihre Aufgaben verwenden können.
- **Datenerfassung und -analyse:** Erfahren Sie, wie Sie mit der Software Daten erfassen, analysieren und interpretieren können.
- **Fehlerbehebung:** Lernen Sie die Verfahren zur Fehlerbehebung und bei Problemen mit der Software.

1.2 Verwendete Software und Werkzeuge

- **Software Development and Design Tools**
  - **Programming Languages and Frameworks:** Python, PyQt6
  - **Integrated Development Environments (IDEs):** Visual Studio Code (VS Code), PyCharm
- **Design Tools**
  - **UI Design and Prototyping:** PyQt Designer, Figma, Miro, Mermaid, Whimsical
  - **Iconography and Graphic Elements:** Flaticons, Iconduck.com

- **Database Management and Interaction Tools**
  - **Database Engine:** SQLite - Eine leichte, dateibasierte Datenbank, die für die Speicherung, Abfrage und Verwaltung der Daten innerhalb der automatisierten Laborstraße verwendet wird.
  - **Database Interface:** DB Browser für SQLite - Ein visuelles Tool zur Interaktion mit SQLite-Datenbanken, das das Erstellen, Designen und Bearbeiten von Datenbankdateien erleichtert.
- **Collaboration and Version Control:** GitHub
- **Testing and Debugging Tools:** PyCharm's debugging capabilities
- **Documentation Tools:** Overleaf LaTeX, Obsidian

1.3 Optimierungsmöglichkeiten

1.3.1 Übersicht über abgeschlossene Aufgaben

1.3.2 Zukünftige Verbesserungen

1.4 Installation und Einrichtung

1.4.1 Einführung

Diese Anleitung beschreibt die Installation und Einrichtung des Agile Roboticsystems Laborstrasse GUI Dashboards, welches mit PyQt6 erstellt wurde.

1.4.2 Voraussetzungen

Stellen Sie sicher, dass die folgenden Komponenten installiert sind:

- Python 3.9 oder neuer
- pip (Python Paketmanager)

1.4.3 Einrichtung

Repository klonen

Klonen Sie das Repository auf Ihren lokalen Rechner mit Git.

```
git clone https://github.com/Prof-Adrian-Mueller/
    Agile_Roboticsystems_Laborstrasse.git
```

Navigieren

Wechseln Sie in das Hauptverzeichnis `Agile_Roboticsystems_Laborstrasse`.

```
cd Agile_Roboticsystems_Laborstrasse
```

Virtuelle Umgebung erstellen

Erstellen Sie eine virtuelle Umgebung im Hauptverzeichnis. Wir nennen sie ‘venv‘.

```
python3 -m venv venv
```

Virtuelle Umgebung aktivieren

Aktivieren Sie die virtuelle Umgebung. Unter Windows verwenden Sie:

```
venv\Scripts\activate
```

Unter Unix oder MacOS:

```
source venv/bin/activate
```

Erforderliche Pakete installieren

Installieren Sie alle erforderlichen Pakete mit:

```
pip install -r requirements.txt
```

1.4.4 Anwendung starten

Nach der Einrichtung können Sie die Anwendung mit dem folgenden Befehl starten:

```
python -m GUI.MainWindow
```

Dies startet die GUI-Anwendung des Dashboards.

ODER

PyCharm Run Configuration einrichten

Run Configuration für `GUI.MainWindow`

Um eine Run Configuration in PyCharm einzurichten, gehen Sie wie folgt vor:

1. Öffnen Sie PyCharm und wählen Sie das Projekt aus.
2. Gehen Sie zum Menü „Run“ und wählen Sie „Edit Configurations“.
3. Klicken Sie auf das Plus-Symbol, um eine neue Konfiguration hinzuzufügen.
4. Wählen Sie „Python“ als Konfigurationstyp.
5. Geben Sie der Konfiguration einen Namen, z.B. „Dashboard GUI“.
6. Stellen Sie sicher, dass das richtige Python-Interpreter aus Ihrer virtuellen Umgebung ausgewählt ist. Wählen Sie unter „Python interpreter“ die Option „Existing interpreter“ und navigieren Sie zum ‘venv/scripts/python.exe‘ Ihrer Projektumgebung. S. Abb. 1.1 & 1.2

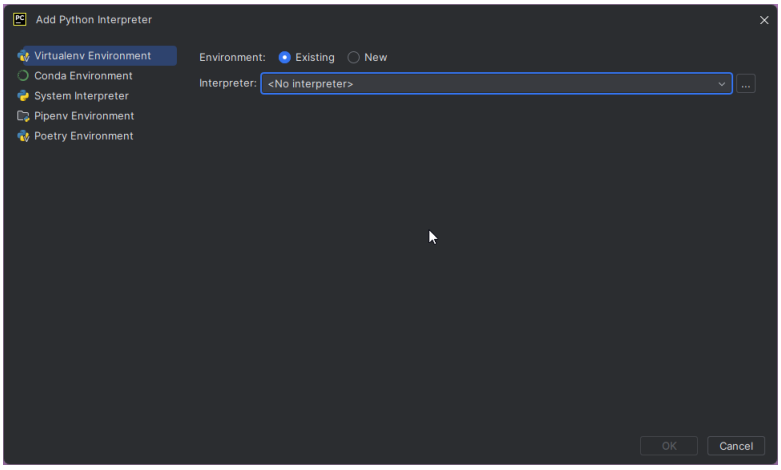


Abbildung 1.1: Python Interpreter hinzufügen

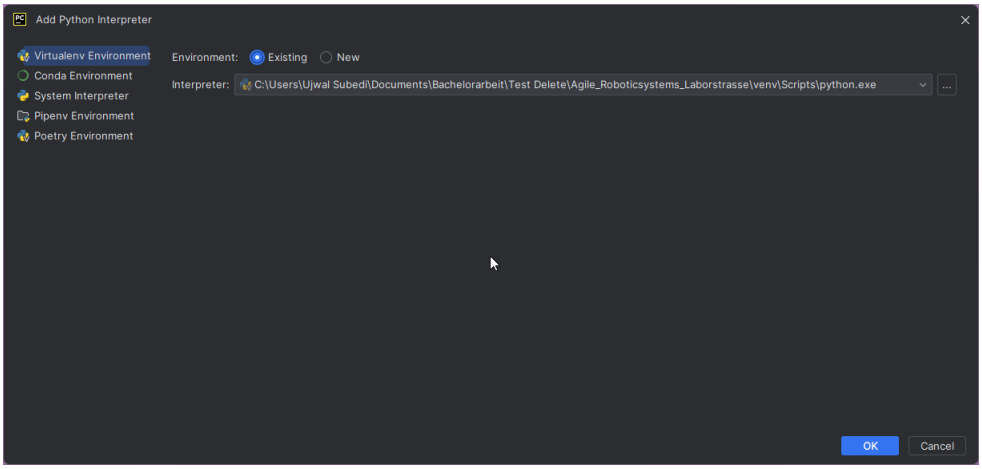


Abbildung 1.2: Python Interpreter ausgewählt

7. Im Feld „Script path“ geben Sie den Pfad zur `MainWindow.py` Datei an oder wählen Sie „Module name“ und geben Sie `GUI.MainWindow` ein. S. Abb. 1.3

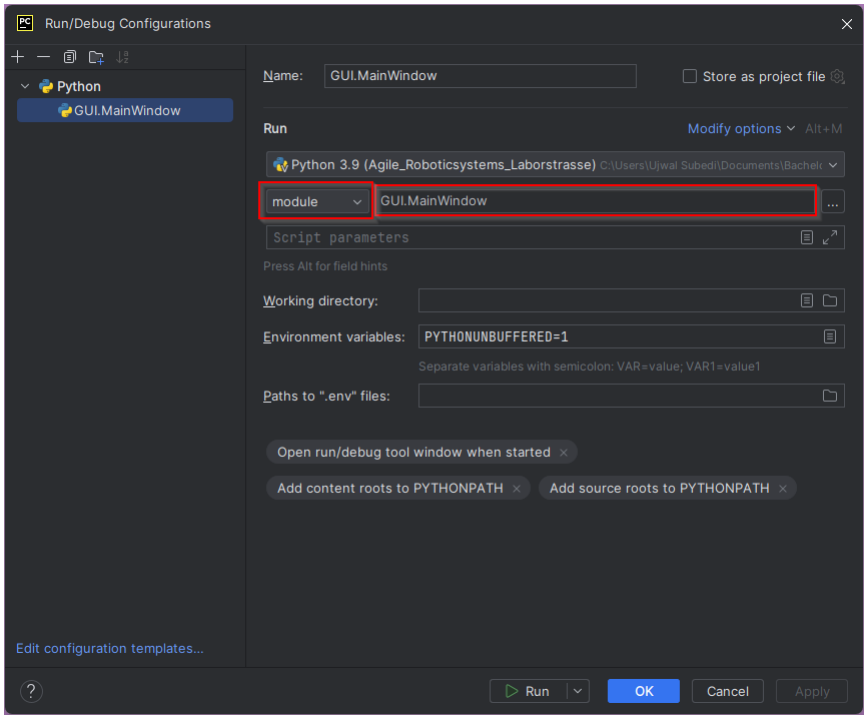


Abbildung 1.3: Module Configuration Settings for PyCharm

8. Bestätigen Sie mit „OK“.

Nun können Sie das Projekt direkt aus PyCharm heraus starten.

Kapitel 2

Benutzerhandbuch

2.1 Anleitung Labor-Assistent

Lieber Laborassistent, wir haben das Protokoll für die Einleitung von Experimenten neu entwickelt und weitere nützliche Funktionen für die Verwaltung von Experimenten im Dashboard Laborstraße dieser Anwendung hinzugefügt. Anbei finden Sie die ausführliche Anleitung zur Nutzung der Anwendung. Es ist unerlässlich, sich genau an diese Anweisungen zu halten, um Präzision und Effektivität in unseren experimentellen Arbeitsabläufen zu gewährleisten.

2.1.1 Prozess für den Laborassistenten

Folgen Sie diesen Schritten im Rahmen Ihrer Tätigkeit als Laborassistent:

1. Experimentvorbereitung
2. Eingabe von Experiment-ID, Name, Nachname, Anzahl der Plasmide, Anzahl der Tubes, Liste der Plasmide in durch Kommas getrenntem Text (z.B. PHB654,PHB655)
3. Eingabe der Liste der Tube Nr. in durch Kommas getrenntem Text (z.B. 1,2)
4. Drucken des QR-Code-Bildes
5. Starten der Erfassung und Tracking Applikation
6. Live Ansicht Überprüfen
7. Ergebnis der Experimente überprüfen

Hinweise zur Durchführung

- Achten Sie darauf, dass alle Eingaben korrekt und vollständig sind, um Fehler im Experiment zu vermeiden.
- Überprüfen Sie die gedruckten QR-Codes sorgfältig auf ihre Lesbarkeit und korrekte Zuordnung.
- Stellen Sie sicher, dass die Überwachungsanwendung korrekt konfiguriert ist, bevor Sie das Experiment starten.

2.1.1.1 Import Plasmid Metadaten

Das korrekte Importieren von Plasmid-Metadaten ist ein entscheidender Schritt für die Vorbereitung und Durchführung von Experimenten im Dashboard Laborstraße. Bitte folgen Sie diesen Schritten, um die Metadaten erfolgreich zu importieren:

1. **Navigieren Sie zum Import-Bereich:** Wählen Sie im linken Menü der Anwendung das Symbol mit dem nach unten zeigenden Pfeil aus. Dies ist der Bereich für den Import von Dateien. S. Abb. 2.1

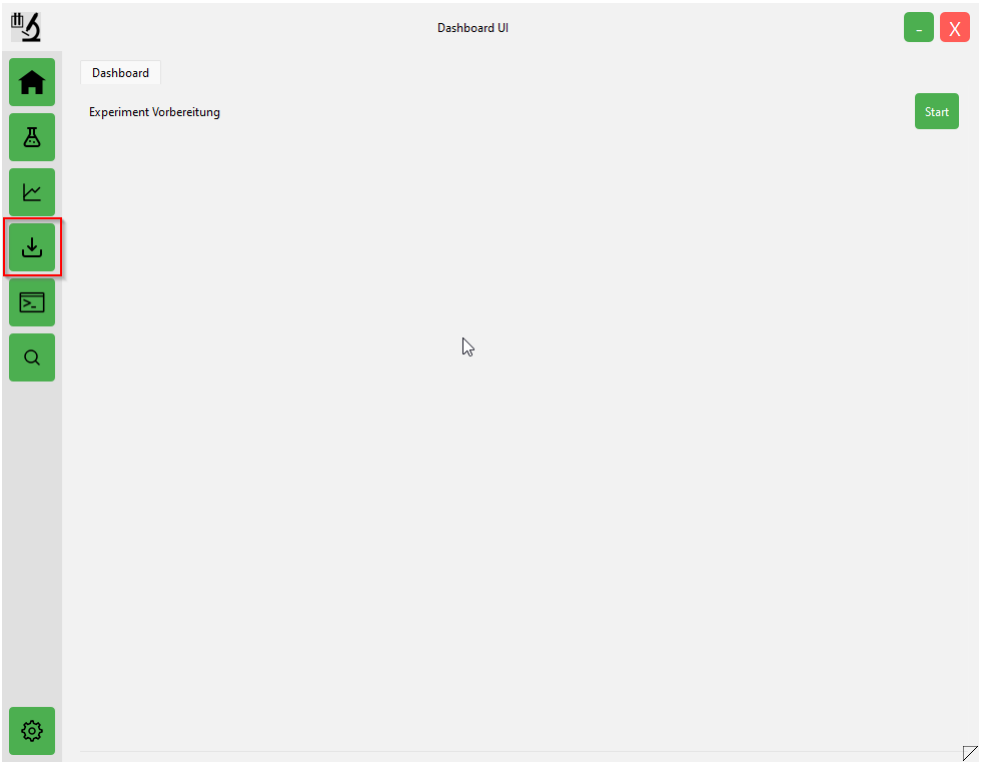


Abbildung 2.1: Import Plasmid Navigation

2. **Import-Bereich betreten:** Nach der Auswahl des Import-Symbols gelangen Sie in den Bereich, in dem Sie Plasmid-Metadaten importieren können.

3. **Import der Plasmid-Metadaten starten:** Klicken Sie auf die Schaltfläche „Plasmid Metadaten Importieren“, die sich im unteren Bereich des Dashboards befindet. S. Abb. 2.2

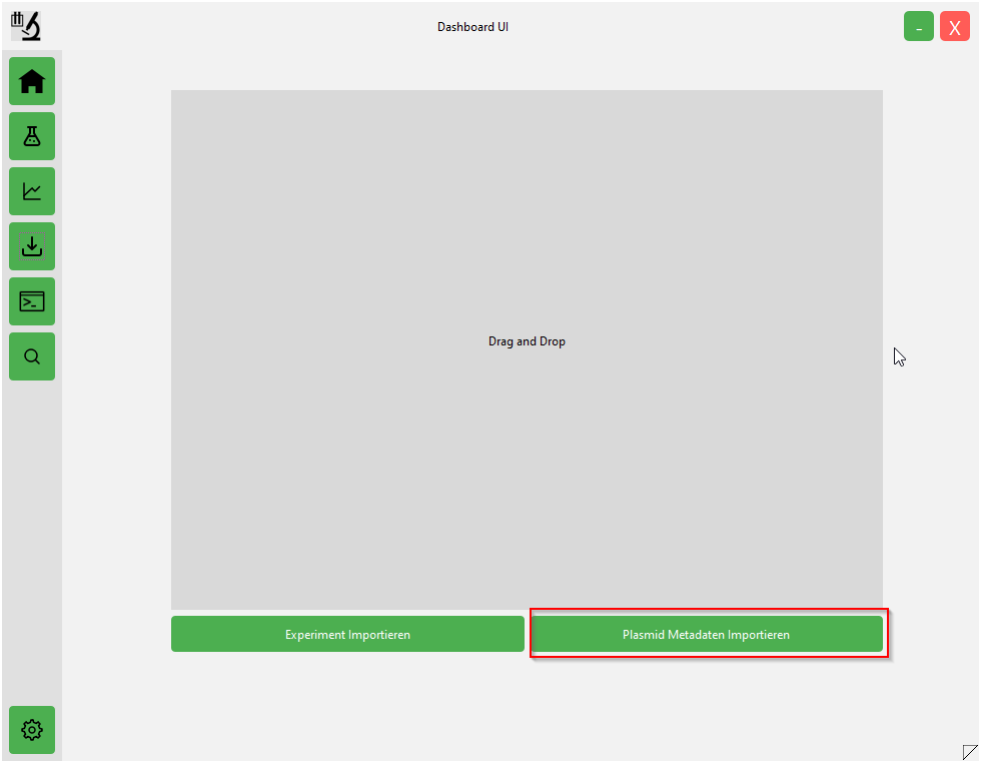


Abbildung 2.2: Import Plasmid Window

4. **Auswahl der Plasmid-Datei:**

   Haben Sie bereits eine Plasmid-Tabelle, navigieren Sie im Datei-Öffnen-Dialog zum Speicherort Ihrer Datei. Sollten Sie noch keine Tabelle besitzen, finden Sie eine Vorlagedatei im Ordner DBService/DefaultTemplate. Navigieren zum DefaultTemplate-Ordner: Im Dialogfenster „Datei öffnen“ nutzen Sie die Seitenleiste, um zum Ordner DBService/DefaultTemplate zu gelangen. S. Abb. 2.3

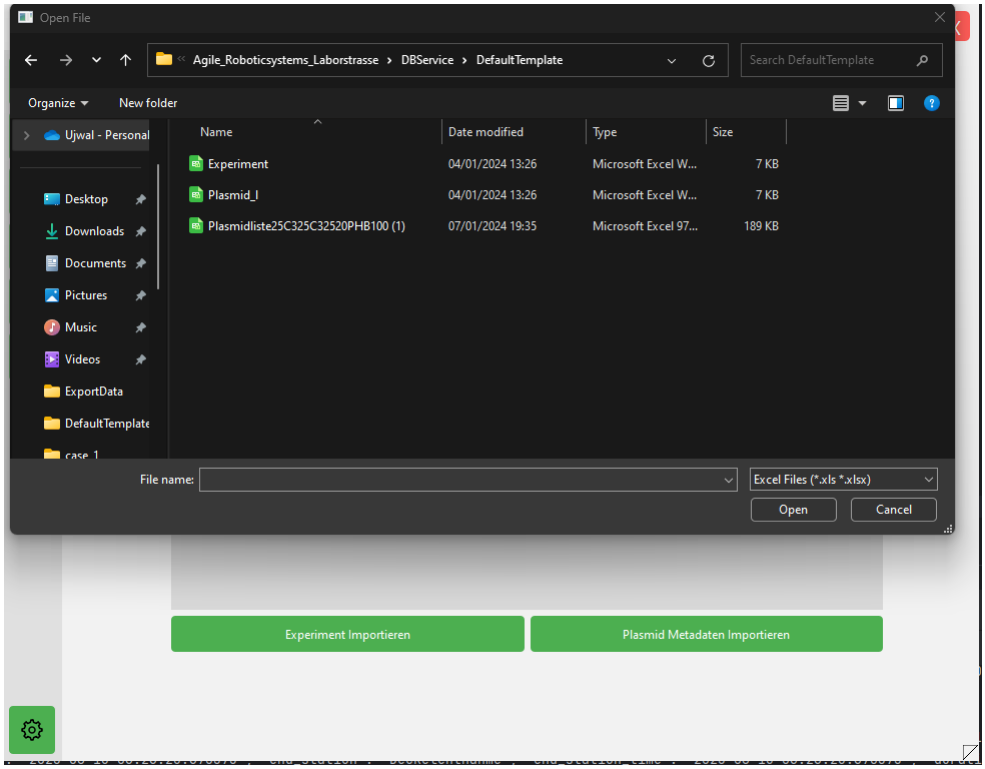


Abbildung 2.3: Plasmid-Metadaten Excel Datei Auswahl Dialog

5. **Auswählen der Plasmid-Tabelle:**

   Wählen Sie die benötigte Excel-Datei aus. Achten Sie darauf, dass die Datei im richtigen Format vorliegt (z.B. .xls oder .xlsx). Bestätigen Sie Ihre Auswahl mit dem „Öffnen“-Button. Durchführung des Imports: Die ausgewählte Datei wird nun importiert. Kontrollieren Sie anschließend, ob alle Daten korrekt in das Dashboard übernommen wurden. S. Abb. 2.4

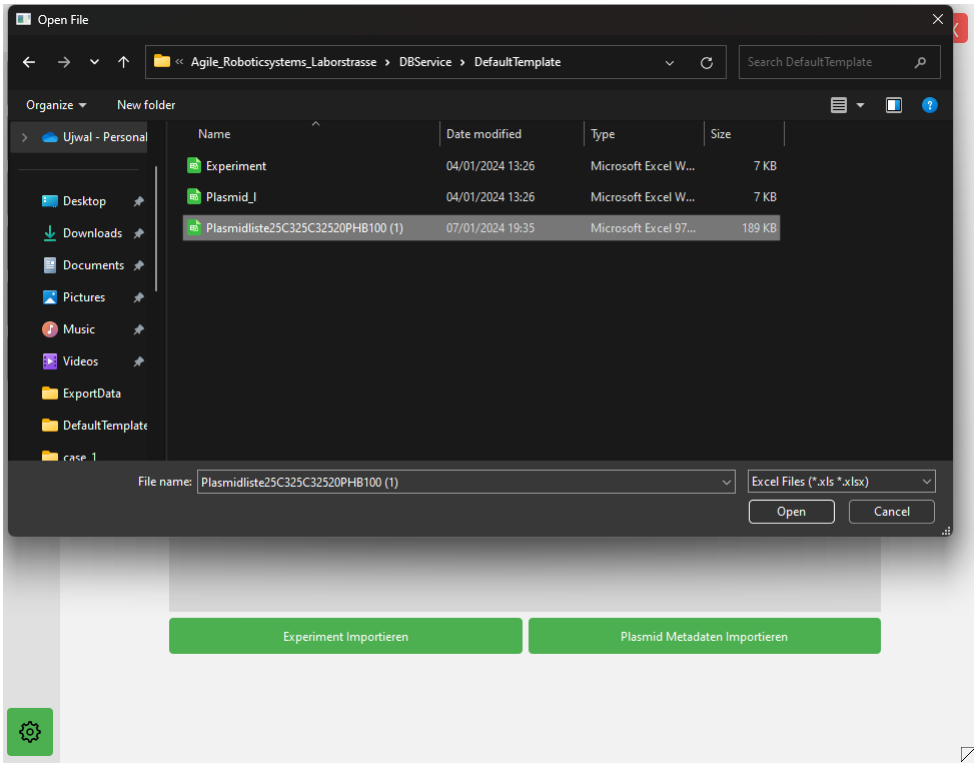


Abbildung 2.4: Plasmid-Metadaten Excel Datei ausgewählt

6. **Import abschließen:** Bei erfolgreichem Import erhalten Sie eine Bestätigungsnachricht. Überprüfen Sie, ob die importierten Daten vollständig und korrekt sind. S. Abb. 2.5

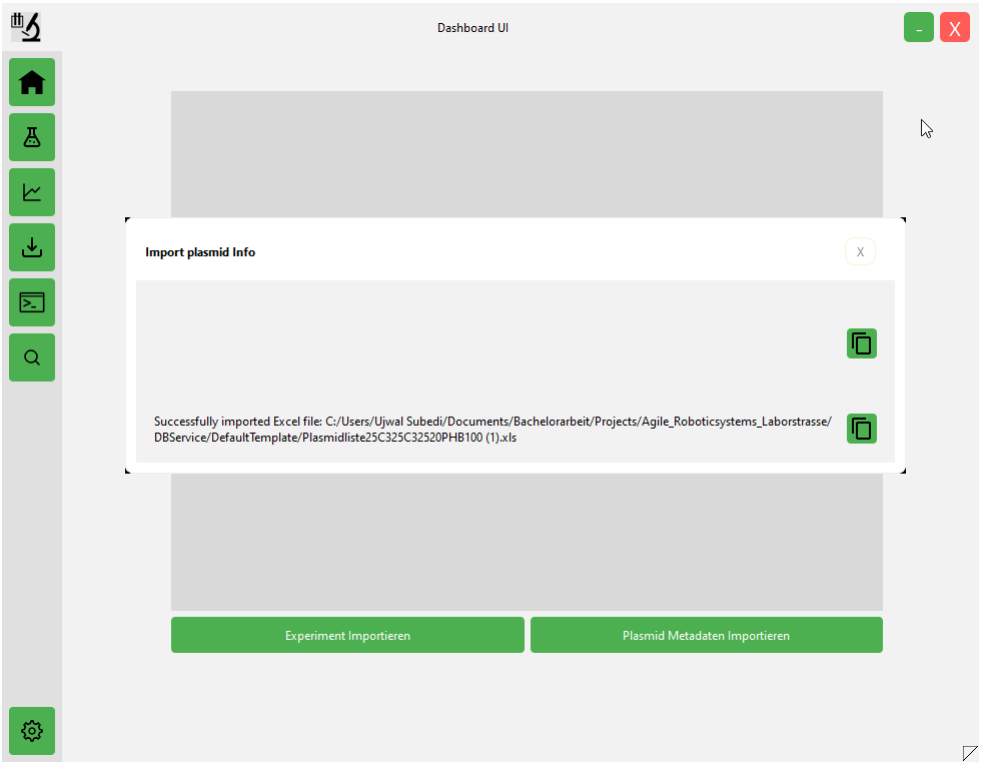


Abbildung 2.5: Import Erfolg Dialog

Bitte stellen Sie sicher, dass die zu importierende Excel-Datei kompatibel ist und die Plasmid-Informationen korrekt strukturiert sind, um Fehler beim Importprozess zu vermeiden.

Erforderliche Excel-Spalten für Plasmid Metadaten

1.	Plasmid Nr.
2.	Antibiotika
3.	Vektor
4.	Insert
5.	Spezies/Quelle
6.	Sequenz Nr. Name
7.	Datum Maxi
8.	Quelle + Datum der Konstruktion
9.	Verdau
10.	Klonierungsstrategie
11.	Bemerkung
12.	Farbcode der Plasmide

2.1.1.2 Die Erstellung eines neuen Experiments.

Der Prozess zur Einrichtung eines neuen Experiments beginnt mit Ihrer persönlichen Eingabe von Namen und Vornamen. Als nächsten Schritt geben Sie bitte die Anzahl der Plasmide und Tubes an, die für das Experiment benötigt werden. Beachten Sie, dass die Anzahl der Plasmide zwischen 1 und 32 liegen sollte und diese nicht die Anzahl der Tubes überschreiten darf. Die Tube-Anzahl muss eine gerade Zahl sein, die zwischen 2 und 32 liegt. Bitte tragen Sie die Plasmide, getrennt durch Kommas und ohne Leerzeichen nach den Kommas, ein. Ihre Sorgfalt und Genauigkeit bei diesen Angaben sind entscheidend für den erfolgreichen Ablauf des Experiments.

Schritt 1: Start

- Öffnen Sie die Anwendung und klicken Sie auf den grünen **Start**-Button in der oberen rechten Ecke der Dashboard-Oberfläche, um mit der Einrichtung des Experiments zu beginnen.

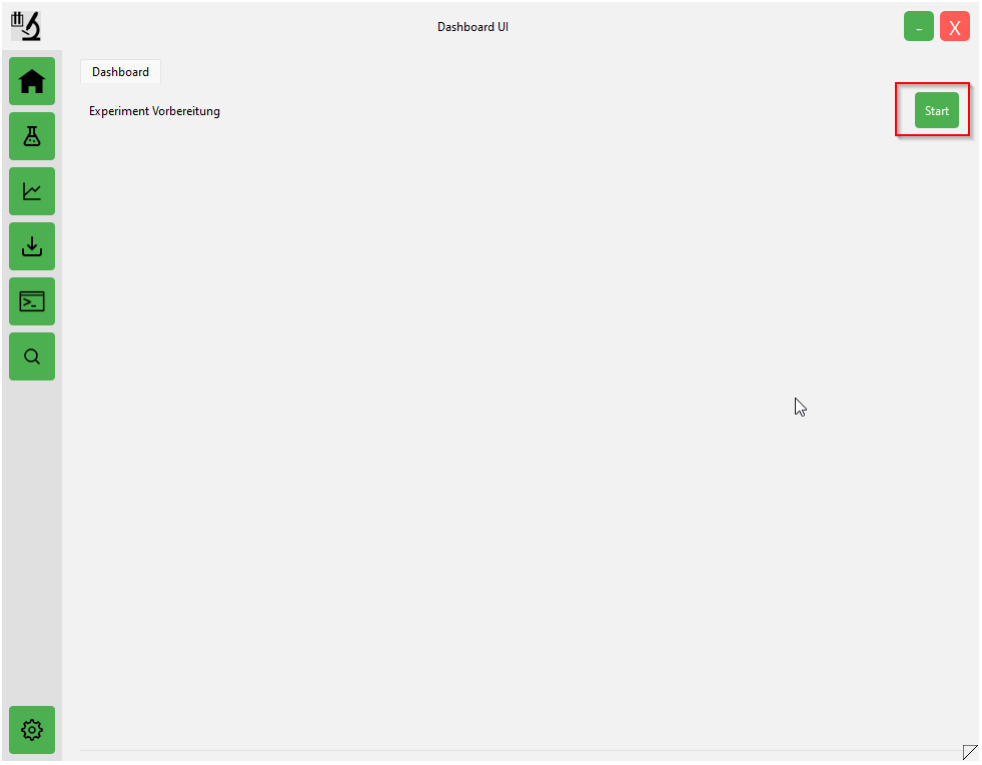


Abbildung 2.6: Start Experiment Vorbereitung

Schritt 2: Experimentdetails eingeben

- Auf der Seite *Experiment Vorbereitung* geben Sie folgende Details ein:
  - **Experiment ID**: Wenn es sich um ein neues Experiment handelt, lassen Sie dieses Feld leer; andernfalls geben Sie die ID eines bestehenden Experiments ein. z.B. ***Max1***
  - **Name**: Geben Sie den Nachnamen der Person ein, die das Experiment durchführt. z.B. ***Max***
  - **Vorname**: Geben Sie den Vornamen der Person ein, die das Experiment durchführt. z.B. ***Mustermann***
  - **Anzahl Plasmid**: Geben Sie die Gesamtzahl der im Experiment verwendeten Plasmide ein, die zwischen 1 und 32 liegt. z.B. ***2***
  - **Anzahl Tubes**: Geben Sie die Anzahl der verwendeten Tuben ein, die zwischen 2 und 32 liegt und gerade ist. z.B. ***Max1***
  - **Liste von Plasmid**: Geben Sie die IDs der Plasmide getrennt durch Kommas ein. Die bereitgestellten Plasmide sollten in unserer Datenbank vorhanden sein. Falls nicht, importieren Sie die Plasmide bitte mit dem Plasmid Metadatataten Excel-Importer 2.1.1.1. Beispiel Plasmid Liste ***PHB654,PHB655***
  - **Datum**: Das Datum wird automatisch mit dem aktuellen Datum ausgefüllt; falls nötig, anpassen.

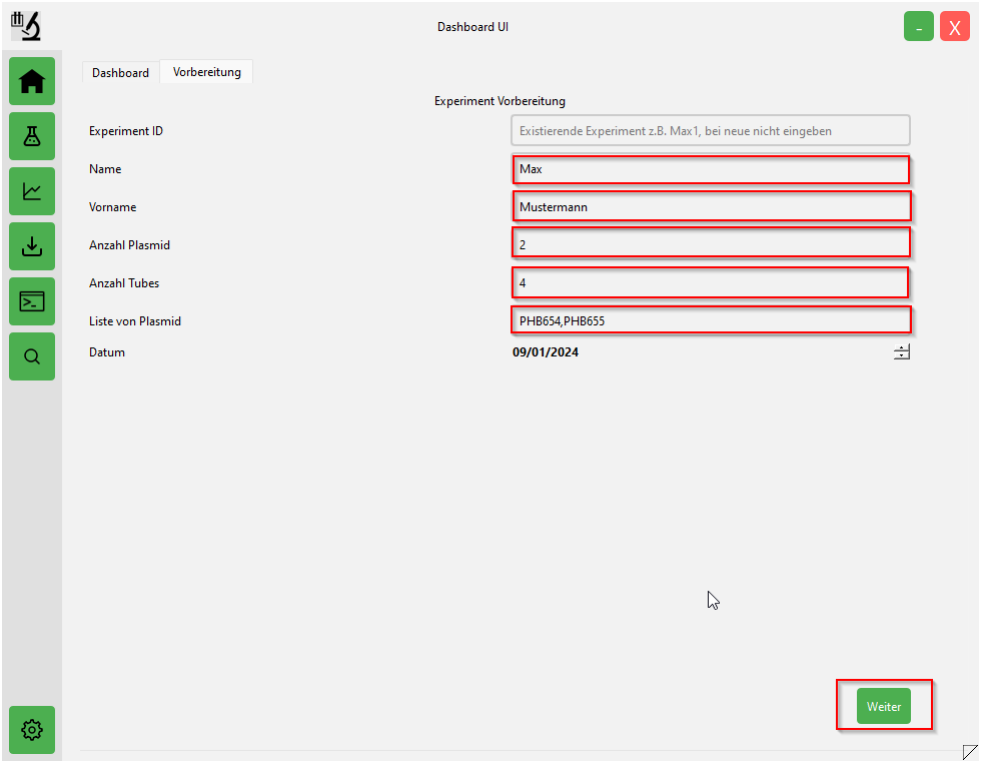


Abbildung 2.7: Experiment Daten Eingabe

- Nach der Eingabe der Details klicken Sie auf den grünen **Weiter**-Button.

2.1.1.3 Tubes zuweisen

Nach der Eingabe der grundlegenden Informationen weist der Laborant auf dem zweiten Bildschirm den Plasmiden bestimmte Tubes zu. Dies geschieht durch Eingabe der Tube-Nummern in die Felder neben den Plasmid-Nummern. Sobald der Laborant auf ***Weiter*** klickt, werden die Eingaben überprüft und in der Datenbank gespeichert.

Anschließend wird eine Bestätigungsnachricht gezeigt, die dem Benutzer signalisiert, dass die Eingabe korrekt war und die Tubes erfolgreich den Plasmiden zugewiesen wurden. Es werden spezifische Nachrichten für die erstellten Tubes angezeigt, jeweils für die Plasmide `PHB655` und `PHB654`, mit den zugeordneten Tube-Nummern. Am Ende gibt es eine abschließende Nachricht, die die erfolgreiche Operation bestätigt: „Prima! Alle Daten sehen Gut aus.“ Siehe Abbildung 2.9.

Schritt 3: Tubes den Plasmiden zuordnen

- Ordnen Sie auf dem nächsten Bildschirm jedem Plasmid die entsprechenden Tube-Nummern zu:
  - Für jedes aufgeführte Plasmid geben Sie die entsprechenden Tube-Nummern in die dafür vorgesehenen Felder ein.
- Klicken Sie auf den grünen **Weiter**-Button, um fortzufahren.

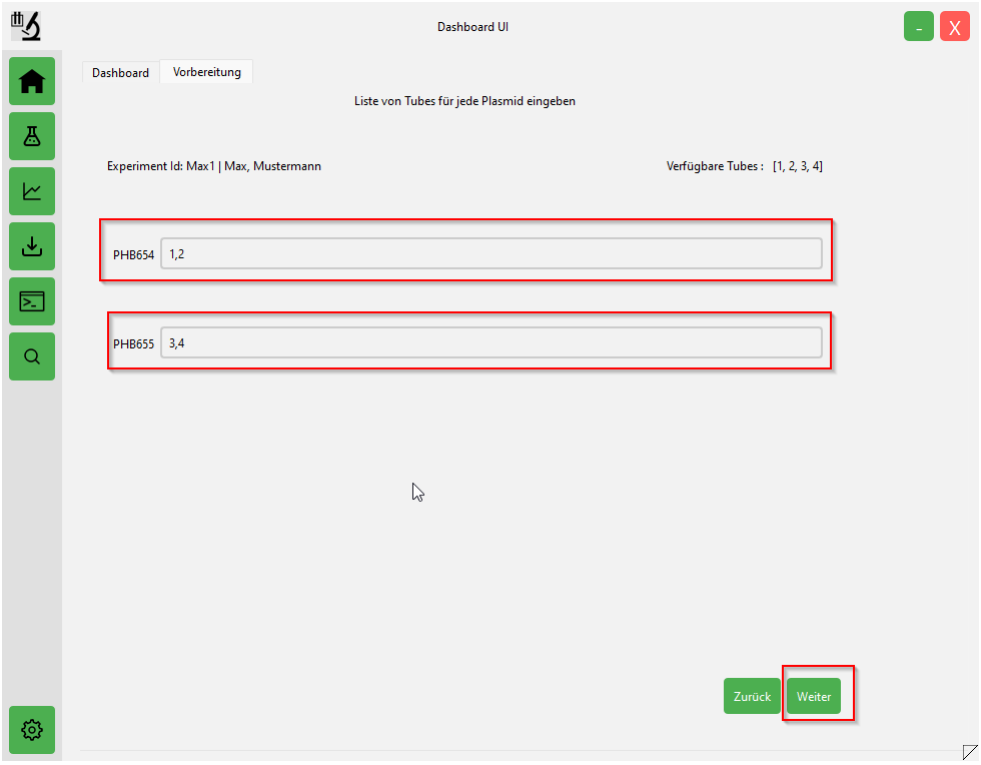


Abbildung 2.8: Zuordnung der Röhrchen zum Plasmid

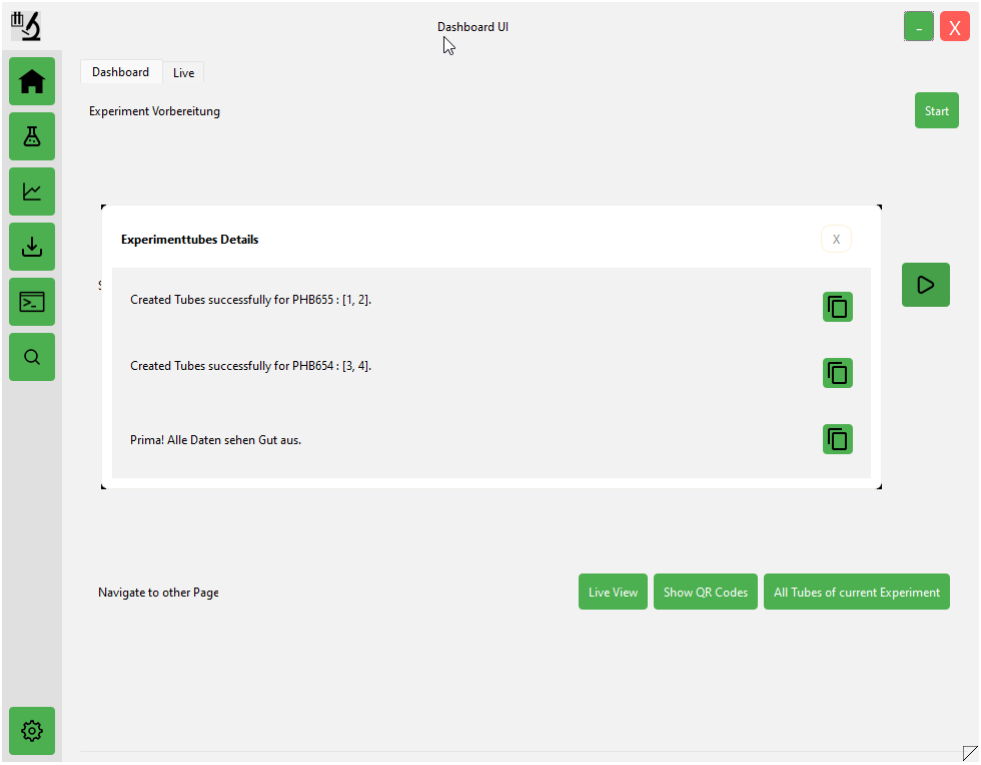


Abbildung 2.9: Bestätigung der erfolgreichen Zuweisung von Tubes zu Plasmiden

2.1.1.4 Prüfen auf vorhandenes Experiment:

Wenn der Laborant seinen Namen und Vornamen erneut eingibt, wird überprüft, ob bereits ein Experiment vorhanden ist. Falls ein Experiment existiert (z.B., ein Laborant mit dem Namen "Max"hat bereits ein Experiment), wird das neue Experiment als das nächste in der Reihenfolge erstellt, z.B., mit der Experiment-ID "Max2". Siehe Abbildung 2.7

2.1.1.5 Experiment Daten abrufen und aktualisieren:

Wenn der Laborant die Experiment-ID eingibt (z.B., "Max1"), werden die ID des vorhandenen Experiments heruntergeladen und in jedem Eingabefeld der GUI für das Experiment automatisch aktualisiert, nachdem die ID des vorhandenen Experiments eingegeben wurde. Der Laborant kann nun das Experiment aktualisieren, insbesondere bei der Anzahl der Tubes und Plasmide. Die Anzahl der Tubes und Plasmide darf nicht kleiner sein als die vorherige Anzahl.

Hinweis: Es ist zu beachten, dass eine Erhöhung der Anzahl an Tuben nur möglich ist, wenn Sie das bestehende Experiment modifizieren möchten; eine Reduzierung der Tubenanzahl in einem Experiment, dessen Daten bereits in verschiedenen Tabellen der Datenbank hinterlegt sind, ist nicht durchführbar. Daher empfiehlt es sich, zusätzliche Tuben ausschließlich bei zwingender Notwendigkeit hinzuzufügen. Sollte es Zweifel an der korrekten Anzahl der Tuben geben und diese geringer als bei der letzten Eingabe sein, ist es ratsam, das Experiment zu löschen und neu zu beginnen. Zum Beispiel: Bei einem Experiment namens *Max1* mit 4 Tuben kann die Anzahl der Tuben für dieses Experiment nicht weniger als 4 betragen.

2.1.1.6 Löschen eines bestehenden Experiments

Um ein Experiment zu löschen, klicken Sie auf das Suchsymbol in der linken Navigationsleiste. Geben Sie die Experiment-ID in das Eingabefeld ein und klicken Sie auf die Schaltfläche *Laden*. Auf der rechten Seite, unterhalb der *Laden*-Schaltfläche, sehen Sie ein Mülleimer-Symbol. Klicken Sie darauf, um das Experiment zu löschen.

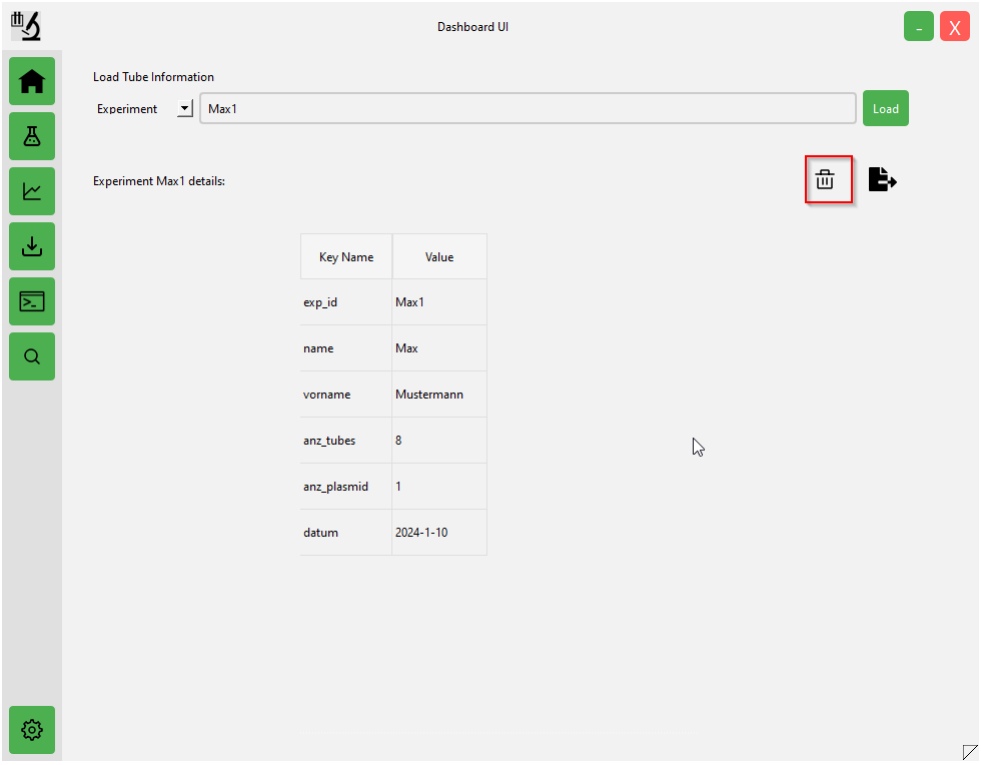


Abbildung 2.10: Experiment Löschen

Wenn ein Popup mit der Frage „**Möchten Sie das Experiment wirklich löschen?**„erscheint, bestätigen Sie mit **Ja**.

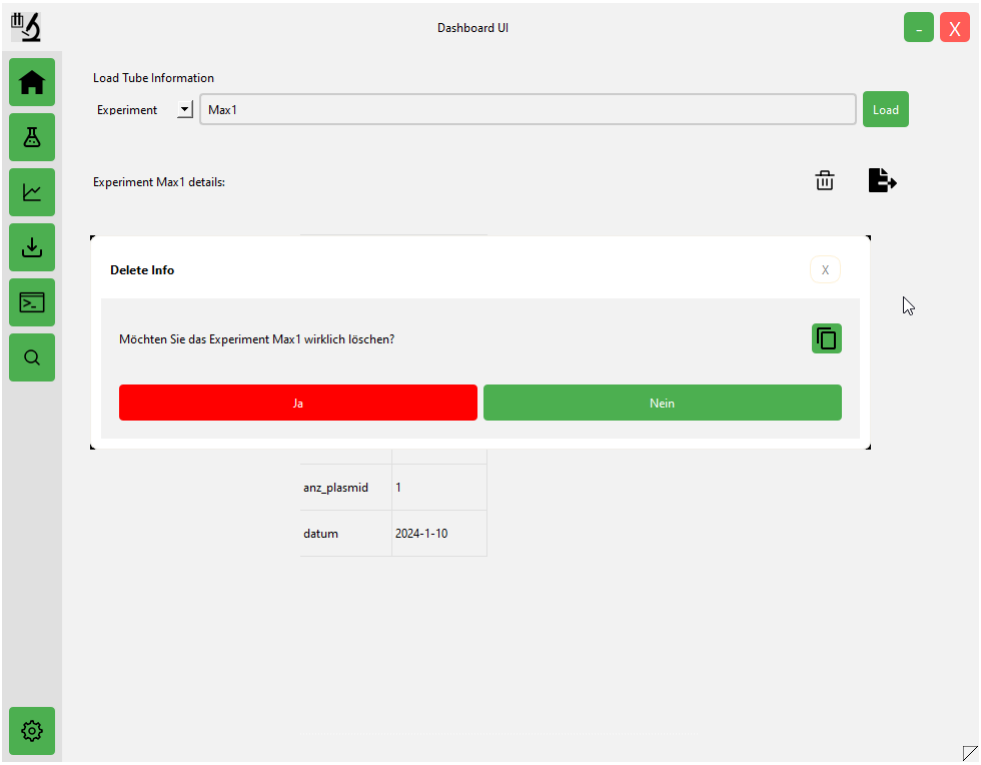


Abbildung 2.11: Experiment Löschen bestätigen

2.1.1.7 Starten und Navigieren der Anwendung

Die GUI Dashboard UI wurde entwickelt, um die Überwachung und Verwaltung von automatisierten Laborprozessen zu erleichtern. Die Oberfläche ist so strukturiert, dass sie eine intuitive und benutzerfreundliche Erfahrung bietet, die es dem Benutzer ermöglicht, durch verschiedene Funktionen wie die Vorbereitung von Experimenten, den Import von Metadaten, die Prozesssteuerung und die Systemeinstellungen zu navigieren.

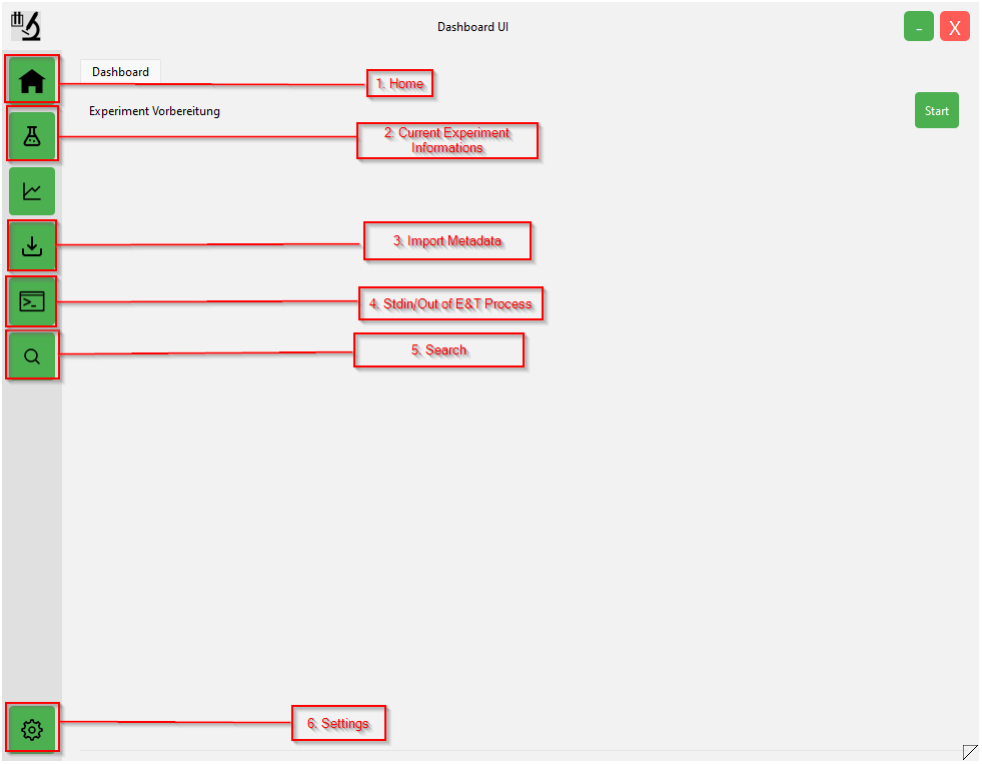


Abbildung 2.12: Navigation Informationen

1. **Heim (Haus-Symbol)**
   - **Experimentvorbereitung**: Erstellen Sie Experimentparameter.
   - **Erfassung und Tracking starten**: Starten Sie das Experiment, indem Sie die Datenerfassung einleiten.
   - **Live-Überwachung**: Verwenden Sie die Funktionen "Live-Ansicht", "QR-Codes anzeigenünd Älle Tubes des aktuellen Experiments", um den Fortschritt und die Details des Experiments zu überwachen.
   - **Navigation**: Bewegen Sie sich zwischen den verschiedenen Abschnitten des Dashboards, um verschiedene Funktionen zu nutzen.
2. **Aktuelle Experimentinformationen (Flask-Symbol)**
   - **Zweck:** Um detaillierte Informationen über die Experimente und ihre Komponenten wie Mikro-QR-Codes für jeden Tube und Tracking-Protokoll des derzeit durchgeführten Experiments zu überprüfen.
   - **Funktion:** Zeigt detaillierte Daten für jedes aktive Experiment an, z. B. Experiment-Tube-Informationen, verantwortliche Personen und den aktuellen Status.

- **Verwendung:** Verwenden Sie diese Funktion, um den Fortschritt zu überwachen, Mikro-QR-Codes zu exportieren oder aktuelle Experimente zu analysieren.

3. **Metadaten importieren (Pfeil nach unten auf dem Tray-Symbol)**
   - **Zweck:** Um Plasmid-Metadaten, die für Experimente wichtig sind, in das System zu integrieren.
   - **Funktion:** Ermöglicht den Import experimenteller Daten, z. B. Plasmid-Metadaten.
   - **Verwendung:** Um neue Plasmiddaten einzugeben, klicken Sie auf diese Schaltfläche und folgen Sie den Anweisungen, um Dateien von Ihrem Computer oder Netzwerk hochzuladen.
4. **Stdin/Out des E&T-Prozesses (zweiseitige horizontale Pfeil-Symbol)**
   - **Zweck:** Um direkte Befehlseingabe für das System zu ermöglichen und die Ausgabe des Ausführungs- und Tracking-Prozesses zu verfolgen.
   - **Funktion:** Bietet eine konsolenartige Schnittstelle zum Senden von Befehlen an das System und zeigt die Echtzeitantwort des Systems an.
   - **Verwendung:** Umgeleitete Ausgabe der Erfassungs- und Tracking-Anwendung.
5. **Suchen (Lupensymbol)**
   - **Zweck:** Um bestimmte Daten schnell im System zu finden.
   - **Funktion:** Bietet eine leistungsstarke Suchfunktion, um Experimente, Tubes oder Plasmide in der Datenbank des Systems zu finden.
   - **Verwendung:** Geben Sie die Suchkriterien wie Experiment-ID oder Datum ein und das System wird relevante Ergebnisse abrufen und anzeigen.

2.1.1.8 Verwendung der wichtigsten Funktionen

2.1.2 Anleitung System-Admin

Lieber Systemadministratoren,
In diesem Teil des Handbuchs wird Ihnen der Umgang mit dem **DB-Browser für SQLite** erläutert. Diese Anleitung hilft Ihnen beim Durchsuchen der Daten, beim Überprüfen der Datenbankstruktur und beim Ausführen von SQL-Abfragen. Wir setzen voraus, dass Sie bereits über Grundkenntnisse in der Datenverwaltung verfügen. Der DB-Browser für SQLite bietet eine intuitive Oberfläche, die die Datenverwaltung vereinfacht. Sie können damit Daten abfragen, bearbeiten und exportieren, was zur Integrität und Funktionalität der Laborstraßen-Datenbank beiträgt.

Laden Sie zunächst den **DB Browser für SQLite** von https://sqlitebrowser.org/dl/ herunter und installieren Sie ihn. Um eine Datenbank zu öffnen, starten Sie das Programm und wählen Sie im Hauptmenü *Datei* die Option *Datenbank öffnen*. Navigieren Sie zu der gewünschten Datenbankdatei und öffnen Sie diese. Nun können Sie die Datenbank bearbeiten, durchsuchen und SQL-Abfragen ausführen.

2.2 Anleitung für Systemadministratoren

Löschen eines Experiments: Sie können ein Experiment löschen, indem Sie zunächst im DB Browser für SQLite die Tabelle *"Experiment"* unter *"Browse Data"* auswählen. Anschließend wählen sie das zu löschende Experiment aus der Liste aus. Mit einem Klick auf den Button *"Delete Record"* wird das Experiment als auch alle zugehörigen Tubes automatisch aus der Datenbank entfernt.
Dies geschieht durch die *"ON DELETE CASCADE"*-Option in der Fremdschlüsselbeziehung zwischen den Tabellen *Experiment* und *"Tubes"*. Sie müssen lediglich das Experiment unter *"Browse Data"* auswählen und auf *"Delete Record"* klicken, um das Experiment und alle verbundenen Tubes zu löschen.

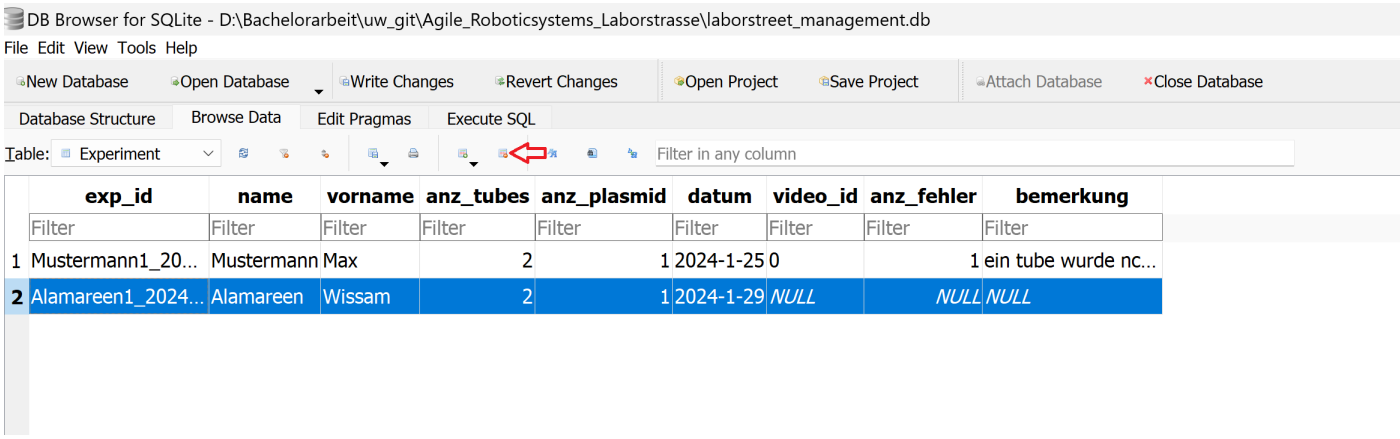


Abbildung 2.13: Löschen eines Experiments im DB Browser

2.2.1 Zugriff auf die Funktionen des aktuellen Experiments

Um das aktuelle Experiment zu überprüfen und die Funktionen zu nutzen, die nur das aktuelle Experiment hat, wie zum Beispiel das Starten der E&T-Anwendung, gehen Sie wie folgt vor:

1. Geben Sie die Experimentvorbereitung ein: Klicken Sie auf den Abschnitt 'Experimentvorbereitung'.

2. Geben Sie die Experiment ID ein: Geben Sie in das vorgesehene Feld die ID des Experiments ein, das Sie überprüfen möchten.

3. Füllen Sie andere Felder mit vorhandenen Daten: Klicken Sie auf die anderen Felder, um sie automatisch mit vorhandenen Daten zu füllen, die sich auf die eingegebene Experiment-ID beziehen.

4. Navigieren Sie zur ursprünglichen Seite: Klicken Sie zweimal auf 'Weiter'. Dadurch gelangen Sie zurück zur ursprünglichen Seite.

5. Starten Sie die E&T-Anwendung : Von der ursprünglichen Seite aus können Sie nun die E&T-Anwendung starten, die eine Funktion des aktuellen Experiments ist.

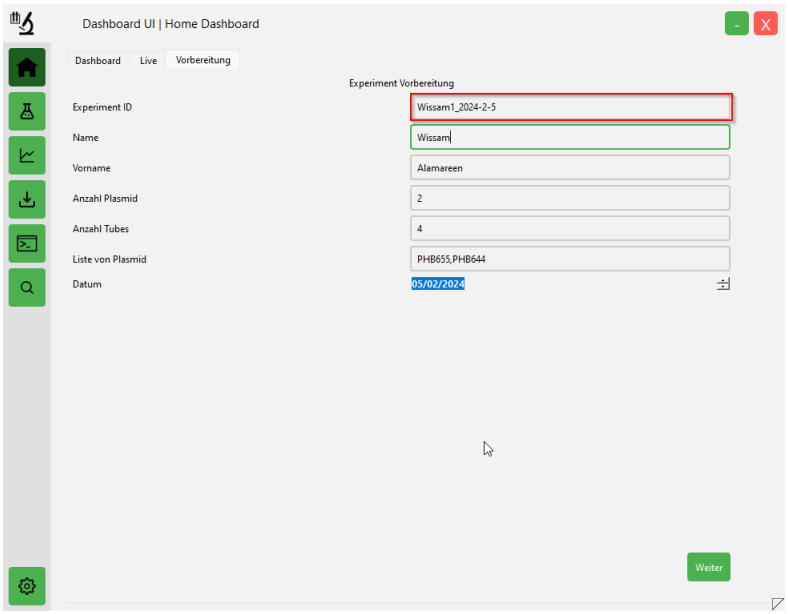


Abbildung 2.14: Eingabe der Experiment-ID zum Laden des aktuellen Experiments

2.2.2 Finden des aktuellen Experiments

Um das aktuelle Experiment zu finden, verwenden Sie das 'Experiment-Suchfeld'. Sie können darauf zugreifen, indem Sie auf das Lupensymbol im linken Navigationsmenü klicken. Geben Sie aktuelle Datum in `"YYYY-MM-DD"`Format ein, um das aktuelle Experiment zu finden.

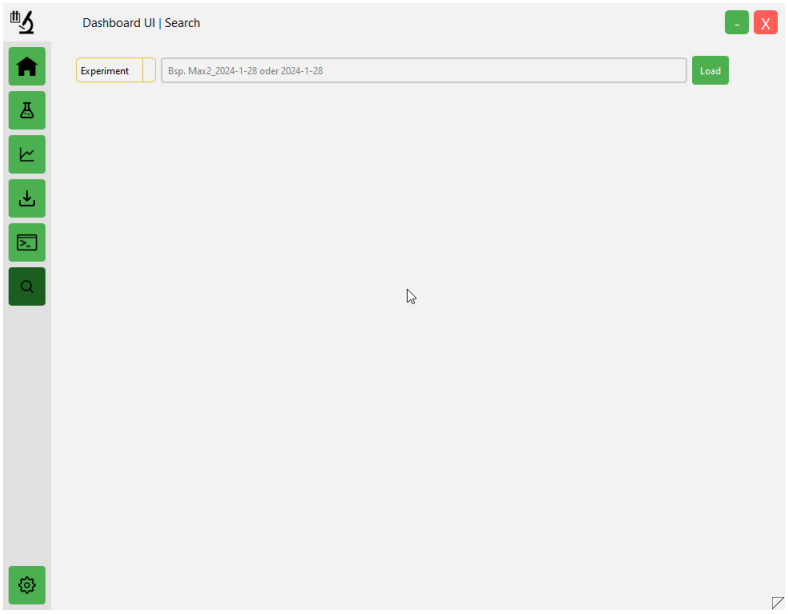


Abbildung 2.15: Experiment Suche

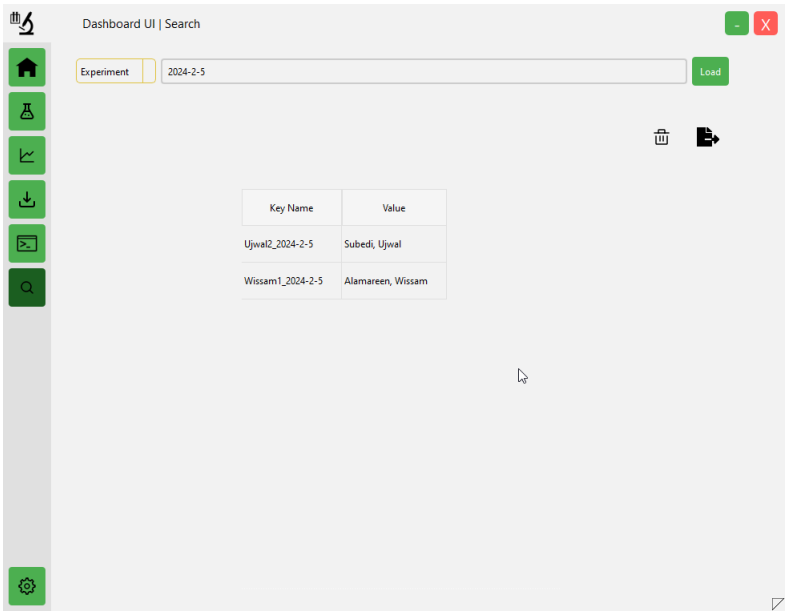


Abbildung 2.16: Experiment Ergebnis des Datums

Abrufen der aktuellen Experiment-ID aus einer Datei

Alternativ können Sie die aktuelle Experiment-ID auch direkt aus einer Datei abrufen:

- Navigieren Sie zum Projektordner:
  `C:\Users\Fujitsu\IdeaProjects\Agile_Roboticsystems_Laborstrasse`.
- Finden Sie die Datei '`application_cache.json`'.
- Öffnen Sie die Datei '`application_cache.json`' mit einem Texteditor.
- Suchen Sie nach der '*experiment_id*'.
- Verwenden Sie den Wert der '*experiment_id*'.

```
{
    "user_preferences": {
        "experiment_id": "Wissam1_2024-2-5",
        "language": "en"
    }
}
```

Listing 2.1: application_cache.json content

Video-Encoder-Funktion

Die Version sollte festgelegt werden, um die Video-Encoder-Funktion zu nutzen. Die benötigten Pakete sind:

- opencv_contrib_python==4.8.0.74
- openh264-1.8.0-win64.dll

Falls es einen Fehler bezüglich openh264 gibt, stellen Sie sicher, dass die openh264-1.8.0-win64.dll-Datei in den Umgebungsvariablen von Windows (Beispiel [1]) eingerichtet ist.

2.3 Anleitung für Developer

2.3.1 Repository-Struktur

- **DBService:** Python-Skripte für das Datenbankmanagement, einschließlich Adapter für Datenbank und Experimente, Importeure für Excel-Daten und Modelle für Experimente und Plasmide.
- **Entladen:** Skripte für den Entladeprozess, einschließlich Steuerung eines DoBot-Arms und Kalibrierung.
- **Erkennen:** Skript für das Lesen von Micro-QR-Codes.
- **GUI:** Python-Skripte für die grafische Benutzeroberfläche des Projekts, einschließlich benutzerdefinierter Widgets, Utility-Skripte, Modelle für verschiedene UI-Komponenten und Ressourcen wie Icons und Stylesheets.
- **Main:** Haupt-Python-Skripte für das Projekt, einschließlich Client-Skripte, ein Skript für die DoBot-Steuerung und eine Implementierung des Beobachtermusters. Es enthält auch eine DLL-Datei für DoBot und einige Video- und Modell-Dateien.
- **ModuleTest:** Skript für die Überwachung der Stationsfunktionalität.
- **Monitoring:** Python-Skripte und Konfigurationsdateien für Überwachungszwecke, einschließlich YOLO-Modell-Dateien und einer benutzerdefinierten YAML-Konfiguration.
- **Report:** Dokumentation und Notizen zum Fortschritt des Projekts, wie Prozessdokumentation, Glossar und To-Do-Listen.
- **SimulationData:** CSV-Dateien für die Protokollierung von Simulationsdaten, einschließlich detaillierter Logs für spezifische Fälle und fehlgeschlagene Simulationen.
- **Tracker_Config:** Skripte, Konfigurationen und Bilder für die Kamerakalibrierung und -verfolgung.

Die Dateien `Navigation.ui` und `resource.py`, die sich im GUI-Ordner befinden, sind integrale Bestandteile der Benutzeroberfläche für das Agile Roboticsystems Laborstrasse-Projekt, die für die Ausführung unter dem Qt-Framework, insbesondere unter Verwendung von PyQt oder PySide, entwickelt wurden.

- **Navigation.ui**: Eine Qt Designer UI-Datei, die das Layout und die Struktur der Navigationschnittstelle definiert. Es spezifiziert das Layout des MainWindow, einschließlich Widgets wie QPushButton für verschiedene Navigationsknöpfe (z.B. Home, Experimentinfo, Statistiken, Datei importieren, Kommandozeilenschnittstelle, Suche und Einstellungen) mit entsprechenden Icons, die aus einer Ressourcendatei geladen werden. Die Navigation ist darauf ausgelegt, minimal zu sein, mit einem vertikalen Layout auf der linken Seite des Hauptfensters, das einen effizienten Zugang zu verschiedenen Bereichen der Anwendung bietet.
- **resource.py**: Eine Python-Datei, die aus einer Qt-Ressourcendatei (‘.qrc‘) generiert wird und Definitionen für in der Anwendung verwendete Ressourcen wie Icons, Bilder und andere statische Dateien enthält. Diese Ressourcen werden in eine Python-Datei kompiliert, um innerhalb des Anwendungscodes verwendet zu werden, was den Zugriff auf und die Verwaltung von UI-Assets erleichtert.

2.3.2 PyQt Designer Einführung

Dieses Tutorial beschreibt die grundlegenden Schritte zur Verwendung von Qt Designer und PyQt6 für die Entwicklung einer Navigationsinterface für die Anwendung [3]. Dieses Tutorial bietet eine Einführung in die Nutzung von Qt Designer und PyQt6 zur Erstellung einer benutzerfreundlichen Navigationsinterface.

Schritt-für-Schritt-Anleitung

1. **Vorbereitung**: Stellen Sie sicher, dass PyQt6 und Qt Designer installiert sind. PyQt6 kann über pip installiert werden mittels dem Befehl `pip install PyQt6`.

2. **Qt Designer starten**: Öffnen Sie Qt Designer und wählen Sie `Main Window` als Formular für die neue Benutzeroberfläche.

3. **Hauptfenster entwerfen**: Fügen Sie ein `Frame` Widget hinzu, um die Navigationsknöpfe zu enthalten. Verwenden Sie `Vertical Layout`, um die Knöpfe vertikal anzuordnen.

4. **Navigationsleiste erstellen**: Platzieren Sie `Push Button` Widgets im Frame für jede Navigationsfunktion. Setzen Sie Eigenschaften wie Icon und Tooltip über den `Property Editor`.

5. **Ressourcen verwalten**: Erstellen Sie eine Ressourcendatei (.qrc) mit Qt Designer's Ressourcen-Editor, um Ihre Icons und Bilder zu verwalten.

6. **UI-Datei speichern**: Speichern Sie Ihre Designarbeit als '.ui' Datei, z.B. 'Navigation.ui'.

7. **UI in Python-Code umwandeln**: Konvertieren Sie die '.ui' Datei in eine '.py' Datei mit dem Befehl `pyuic6 Navigation.ui -o Navigation.py`.

8. **Resource in Python-Code umwandeln:** Bitte beachten Sie, dass nach Änderungen an der Ressourcendatei (.qrc) diese in eine Python-Datei kompiliert werden muss, damit die Ressourcen in Ihrer PyQt6-Anwendung verwendet werden können. Verwenden Sie hierfür das pyrcc6-Werkzeug. Führen Sie den folgenden Befehl aus: `pyrcc6 resources.qrc -o resources.py`

   Einbindung kompilierter Ressourcen in die Anwendung

   Nachdem Sie die Python-Ressourcendatei generiert haben, müssen Sie sie in Ihre Hauptanwendungsdatei importieren oder dorthin, wo Sie auf diese Ressourcen zugreifen müssen. Zum Beispiel: `import resources`

9. **Integration in Ihre Anwendung**: Importieren Sie die generierte '`Navigation.py`' in Ihr Python-Projekt und verknüpfen Sie die Logik Ihrer Anwendung mit der Benutzeroberfläche.

2.3.2.1 Wie kann man Schaltflächen zur linken Navigation hinzufügen?

Die linke Navigationsleiste ist ein wesentliches Element für die Benutzerführung in vielen Anwendungen. Dieses Dokument beschreibt, wie Sie eine solche Leiste in einer PyQt6-Anwendung implementieren können, basierend auf der Klasse `LeftNavigation`.

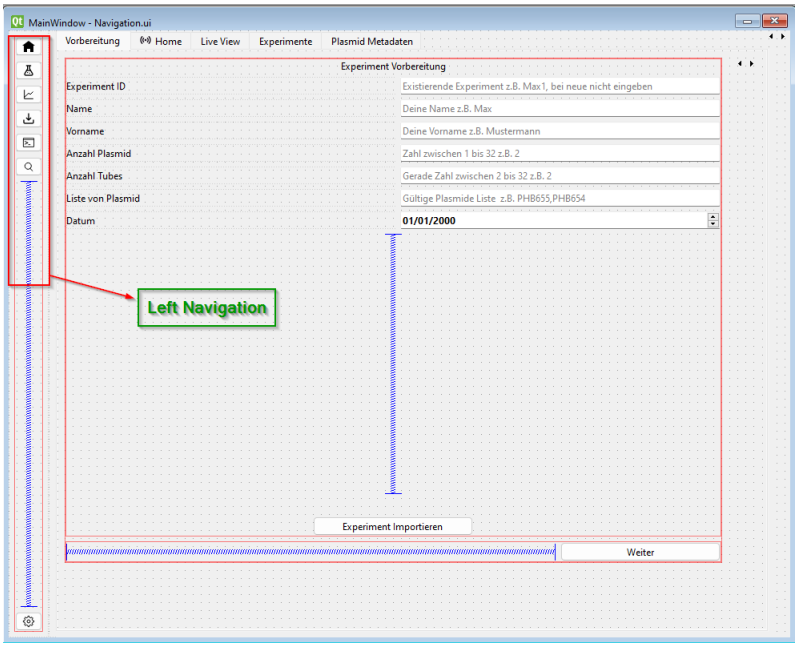


Abbildung 2.17: Fensteransicht der Gestaltung des linken Navigationslayouts im PyQt-Designer

Implementierung

Die `LeftNavigation` Klasse verwaltet die Erstellung und das Verhalten der Navigationsbuttons. Folgende Schritte sind notwendig:

1. Erstellen Sie eine neue Klasse `LeftNavigation`, die die Initialisierung der Buttons und deren Verhalten verwaltet.
2. Definieren Sie eine Liste von Buttons, die zur Navigationsleiste gehören sollen.
3. Verwenden Sie die Methode `map_buttons_to_pages()` um jeden Button mit einer Seite zu verknüpfen.
4. Implementieren Sie die Methode `highlight_button()`, um den aktuell ausgewählten Button hervorzuheben.

Code-Beispiel

Ein vereinfachtes Beispiel für die Initialisierung und Verknüpfung von Buttons:

```
class LeftNavigation:
    def __init__(self, ui, main_window):
        self.ui = ui
        self.main_window = main_window
```

```
        # Add new buttons here
        self.buttons = [self.ui.home_btn_dashboard, ...]

    def map_buttons_to_pages(self):
        self.ui.home_btn_dashboard.clicked.connect(
            lambda: self.ui.stackedWidget.setCurrentIndex(
                self.ui.stackedWidget.indexOf(self.ui.test_page_home)))
        ...
```

Listing 2.2: Code zum Zuordnen der linken Navigationsschaltflächen

Button-Zuordnung

Für die Zuordnung verwenden wir die Methode `map_buttons_to_pages()` innerhalb der `LeftNavigation`-Klasse. Jeder Button wird mit einem Signal verbunden, das beim Klicken ausgelöst wird. Dieses Signal ruft eine Methode auf, die die `setCurrentIndex`-Methode des `QStackedWidget` verwendet, um die gewünschte Seite anzuzeigen.

Implementierung

Hier ist ein Beispiel für die Implementierung der Button-Zuordnung in der `LeftNavigation`-Klasse:

```
def map_buttons_to_pages(self):
  self.ui.home_btn_dashboard.clicked.connect(
    lambda: self.ui.stackedWidget.setCurrentIndex(
      self.ui.stackedWidget.indexOf(self.ui.homePage)))
  self.ui.settingsBtn.clicked.connect(
    lambda: self.ui.stackedWidget.setCurrentIndex(
      self.ui.stackedWidget.indexOf(self.ui.settingsPage)))
  # Weitere Buttons hier zuordnen
```

Hervorhebung des aktiven Buttons

Um den aktuell ausgewählten Button hervorzuheben, implementieren Sie die Methode `highlight_button(button)`:

```
def highlight_button(self, button):
  # Zuruecksetzen der Button-Styles
  for btn in self.buttons:
    btn.setStyleSheet("")
  # Hervorheben des aktiven Buttons
  button.setStyleSheet("background-color: #1B5E20;")
```

Zusammenfassung

Die Klasse `LeftNavigation` bietet eine strukturierte Methode zur Verwaltung einer linken Navigationsleiste in PyQt6-Anwendungen. Durch die Zuordnung von Buttons zu Seiten und die visuelle Hervorhebung des aktiven Buttons verbessert sie die Benutzererfahrung significant.

2.3.3 Hinzufügen von Widgets zu einem QTabWidget in PyQt6

Ein `QTabWidget` ermöglicht es, mehrere Tabs für die Organisation der Benutzeroberfläche in einer PyQt6-Anwendung zu erstellen. Dieses Dokument beschreibt, wie Sie Widgets dynamisch zu einem `QTabWidget` hinzufügen, basierend auf der Methode `setupExperimentView` in der `MainWindow`-Klasse.

Schritte zum Hinzufügen von Widgets

Um Widgets zu einem `QTabWidget` hinzuzufügen, folgen Sie diesen Schritten:

1. Initialisieren Sie das `QTabWidget` in Ihrer Hauptfensterklasse.
2. Erstellen Sie für jeden Tab ein Widget, das die Inhalte des Tabs enthält.
3. Fügen Sie das Widget als neuen Tab zum `QTabWidget` hinzu, indem Sie die Methode `addTab()` verwenden.

Beispiel-Implementierung

Hier ist ein vereinfachtes Beispiel basierend auf der `setupExperimentView`-Methode:

```
def setupExperimentView(self):
    # Erstellen des QTabWidgets
    self.tab_widget_experiment_qr = QTabWidget(self.ui.
    experiment_info_view)

    # Creating the widget for the first tab
    experiment_tubes_widget = QWidget()
    experiment_tubes_layout = QVBoxLayout(experiment_tubes_widget)

    # Here you add the specific widgets and layouts to the tab
    ...

    # Adding the tab to the QTabWidget
    self.tab_widget_experiment_qr.addTab(experiment_tubes_widget, "
    Uebersicht")

    # Repeat the above steps for further tabs
```

Listing 2.3: Code zum Hinzufügen eines Widgets zu einem QTabWidget

Zusammenfassung

Die Methode `setupExperimentView` demonstriert, wie Sie ein `QTabWidget` in Ihrer PyQt6-Anwendung verwenden können, um verschiedene Ansichten oder Inhalte organisiert in Tabs darzustellen. Durch das dynamische Hinzufügen von Widgets können Sie eine flexible und benutzerfreundliche Schnittstelle erstellen.

2.3.4 Anwendung von Stylesheets in PyQt6

Die Anwendung von Stylesheets in PyQt6 ermöglicht es Entwicklern, das Erscheinungsbild ihrer Anwendungen anzupassen. Diese Dokumentation beschreibt, wie ein Stylesheet aus einer externen ‘.qss‘-Datei geladen und auf eine PyQt6-Anwendung angewendet wird.

Stylesheet laden und anwenden

Verwenden Sie die Methode ‘`apply_stylesheet`‘ von MainWindow.py, um das Stylesheet zu laden und auf die Anwendung anzuwenden.

Beispiel-Implementierung

Hier ist ein Beispiel, wie das Stylesheet geladen und angewendet wird:

```
def apply_stylesheet(self):
    if os.path.isfile('GUI/stylesheet/stylen.qss') and os.access('GUI/
    stylesheet/stylen.qss', os.R_OK):
        with open('GUI/stylesheet/stylen.qss', 'r') as file:
            stylesheet = file.read()
        self.setStyleSheet(stylesheet)
    else:
        print("Stylesheet-Datei fehlt oder ist nicht lesbar.")
```

Zusammenfassung

Die Methode ‘`apply_stylesheet`‘ lädt das Stylesheet aus der ‘`stylen.qss`‘-Datei und wendet es auf die gesamte Anwendung an. Dieser Ansatz ermöglicht eine zentrale Verwaltung des Erscheinungsbilds der Anwendung und erleichtert das Anpassen und Aktualisieren des Styles.

2.3.5 Interaktion mit anderen Klassen und Komponenten

Die Entwicklung komplexer Anwendungen erfordert oft die Interaktion zwischen verschiedenen Klassen und Komponenten. In PyQt6-Anwendungen ermöglicht dies eine flexible und modulare Architektur, bei der jede Klasse spezifische Aufgaben übernimmt. Die Interaktion zwischen diesen Klassen erfolgt typischerweise über Signale und Slots, Methodenaufrufe oder den Austausch von Datenobjekten.

Signale und Slots

Eine der Hauptmethoden für die Interaktion in PyQt6 ist das Signal- und Slot-System. Signale werden von einer Klasse ausgesendet, um auf ein Ereignis hinzuweisen. Slots sind Methoden, die auf diese Signale reagieren. Durch das Verbinden von Signalen mit Slots können Klassen ohne direkte Referenzen aufeinander kommunizieren.

```
# Beispiel fuer die Verbindung eines Signals mit einem Slot
self.button.clicked.connect(self.handleButtonClick)

def handleButtonClick(self):
    # Logik, die ausgefuehrt wird, wenn das Signal ausgeloest wird
```

Methodenaufrufe

Klassen können auch direkt interagieren, indem sie Methoden anderer Klassen aufrufen. Dies erfordert eine Referenz auf das Objekt der anderen Klasse. Methodenaufrufe sind nützlich für direkte Aktionen und Datenmanipulationen zwischen Klassen.

```
# Beispiel fuer einen Methodenaufruf
self.anotherClass.doSomething()
```

Datenobjekte

Für komplexere Interaktionen, insbesondere wenn Daten zwischen Komponenten ausgetauscht werden müssen, können Datenobjekte verwendet werden. Diese Objekte können Daten kapseln und zwischen Klassen übergeben werden, um Informationen auszutauschen.

```
# Beispiel fuer die Ubergabe eines Datenobjekts
data = DataObject()
data.attribute = "Wert"
self.anotherClass.processData(data)
```

Zusammenfassung

Die Interaktion zwischen Klassen und Komponenten in PyQt6-Anwendungen ist essentiell für die Erstellung modularer und wartbarer Software. Durch die Verwendung von Signalen und Slots, direkten Methodenaufrufen und dem Austausch von Datenobjekten können Entwickler komplexe Anwendungslogiken effektiv implementieren.

2.3.6 Die `CustomDialog`-Klasse

Die `CustomDialog`-Klasse ist eine erweiterte Implementierung der `QDialog`-Klasse von PyQt6, die für das Erstellen benutzerdefinierter Dialogfenster verwendet wird.

Funktionalitäten

Die `CustomDialog`-Klasse bietet folgende Funktionalitäten:

- Anpassbare Titelleiste mit Drag-and-Drop-Funktionalität und Schließknopf.
- Dynamisches Hinzufügen von Widgets wie Textfeldern und Schaltflächen.
- Unterstützt verschiedene Inhaltsarten wie Information, Warnung und Fehler.
- Implementiert Signale zum Senden von Benutzereingaben.

Anwendungsbeispiel

Hier ist ein Beispiel, das die Verwendung der `CustomDialog`-Klasse zeigt:

```
# Initialisierung des CustomDialog
dialog = CustomDialog(parent)

# Hinzufuegen eines Titels
dialog.add_titlebar_name("Meine Dialogueberschrift")

# Inhalt zum Dialog hinzufuegen
content_widget = dialog.addContent("Dies ist eine Benachrichtigung.",
    ContentType.OUTPUT)

# Dialog anzeigen
dialog.show()
```

Signale

Die `CustomDialog`-Klasse definiert das Signal `sendButtonClicked`, das ausgelöst wird, wenn der Senden-Button geklickt wird.

Styling

Der Dialog unterstützt benutzerdefiniertes Styling über Stylesheets, was eine visuelle Anpassung an die Bedürfnisse der Anwendung ermöglicht.

Zusammenfassung

Mit der `CustomDialog`-Klasse können Entwickler erweiterte und angepasste Dialoge in ihren PyQt6-Anwendungen erstellen, die für eine Vielzahl von Benutzerinteraktionen geeignet sind.

2.3.7 `CustomLiveWidget`-Klasse

Die `CustomLiveWidget`-Klasse ist eine benutzerdefinierte PyQt6-Komponente, die für die Echtzeit-Datenvisualisierung innerhalb der Anwendung konzipiert wurde.

Funktionalitäten

Die Klasse bietet folgende Hauptfunktionen:

- Visualisierung von Echtzeit-Daten in einer anpassbaren Ansicht.
- Interaktion mit Benutzern durch eingebettete Schaltflächen und Informationsanzeigen.
- Flexibles Design, das durch externe Stylesheets gestaltet werden kann.

Implementierungsdetails

Die `CustomLiveWidget`-Klasse verwendet eine Kombination aus QVBoxLayouts und QHBoxLayouts, um eine strukturierte Darstellung von Live-Daten zu ermöglichen. Jedes Widget innerhalb des Layouts repräsentiert eine spezifische Station oder einen Status im Überwachungsprozess.

Beispiel

Ein Beispiel für die Initialisierung und Verwendung der `CustomLiveWidget`-Klasse könnte wie folgt aussehen:

```
# Initialisierung des CustomLiveWidget
customLiveWidget = CustomLiveWidget(parent=parentWidget, main_window=
    mainWindow)

# Einbindung des CustomLiveWidget in das Hauptfenster
mainWindow.setCentralWidget(customLiveWidget)

# Datenaktualisierung ausloesen
customLiveWidget.refresh_data()
```

Datenaktualisierung und Ereignisbehandlung

Die Klasse stellt Methoden bereit, um Daten zu aktualisieren (`refresh_data`) und Benutzerereignisse zu behandeln, indem sie Schaltflächen mit entsprechenden Slots verbindet.

Zusammenfassung

Die `CustomLiveWidget`-Klasse erweitert die Funktionalität von Standard-PyQt-Widgets, um eine dynamische und interaktive Benutzererfahrung für Echtzeit-Datenmonitoring zu bieten.

2.4 Aktivierung des Debug-Modus

Öffnen Sie die Datei `Main/main.py` in Ihrem Code-Editor.

Um den Debug-Modus global für die Capture und Tracking Anwendung zu aktivieren, ändern Sie die `is_debug` Parameter in der Konstruktor-Definition auf `True`:

```
def __init__(self, is_debug=True):
```

Diese Änderung ermöglicht es, detaillierte Debug-Ausgaben zu sehen und die Live-Ansicht entsprechend den bereitgestellten Protokolldateien zu aktualisieren.

2.5 Methode `live_simulation` und Protokolldateien

Die Methode `live_simulation` simuliert den Prozess in Echtzeit, wobei spezifische Protokolldateien verwendet werden, um den Ablauf zu veranschaulichen. Für ein konzeptionelles Verständnis und Details zu den verwendeten Protokolldateien, siehe Abschnitt 6 von Ujwal Subedi [4], "Quality Assurance".

2.5.1 Änderung der Konfiguration bei Änderungen am Station

Die Anpassung der Konfiguration, insbesondere die Änderung der Anzahl der Stationen im Labor, erfordert eine Überarbeitung der *TrackingLog-Tabelle* in der Datenbank, um sicherzustellen, dass alle relevanten Stationen erfasst werden können. Die Klasse *DatabaseConnection* und insbesondere die Methode *create_tracking_log_table* sind dafür verantwortlich, die Struktur dieser Tabelle zu definieren. Änderungen an der Anzahl der Stationen könnten eine Anpassung der Attribute in dieser Tabelle notwendig machen, um zusätzliche Stationen aufzunehmen oder nicht mehr benötigte zu entfernen.

Der *TrackingLogAdapter* spielt eine Schlüsselrolle bei der Interaktion mit der *TrackingLog*-Tabelle. Änderungen in der Tabelle erfordern möglicherweise Anpassungen in Methoden wie *insert_tracking_log*, um Daten entsprechend den neuen oder geänderten Stationen zu erfassen. Zudem könnte es notwendig sein, die Logik innerhalb von *get_tracking_logs_by_probe_nr* und *get_tracking_logs_by_exp_id* anzupassen, um sicherzustellen, dass Abfragen korrekt auf die aktualisierte Tabelle reagieren.

Kapitel 3

Fehlerbehebung und FAQs

3.0.1 keine Experimente

Wenn Sie diese Fenster sehen, sollten Sie ein neues Experiment anlegen. Wahrscheinlich gibt es keine Experimente in der Datenbank.

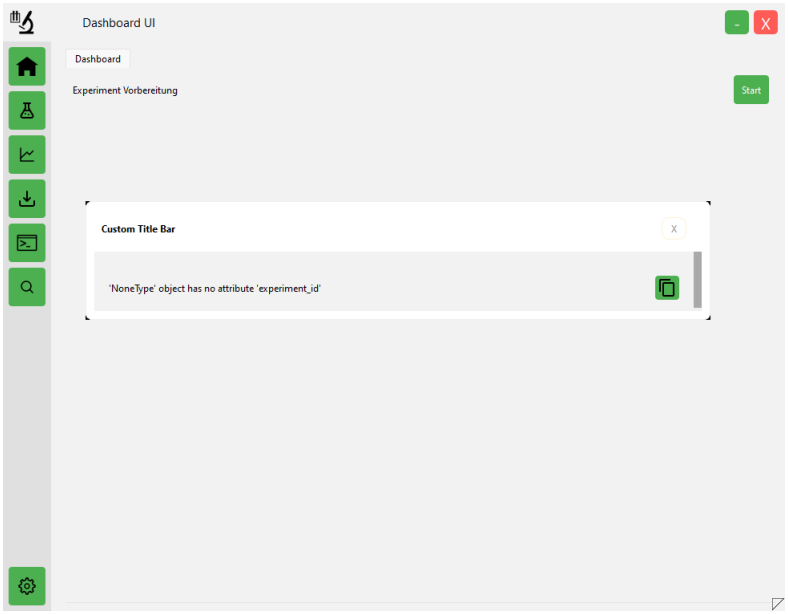


Abbildung 3.1: No Experiment Dialog

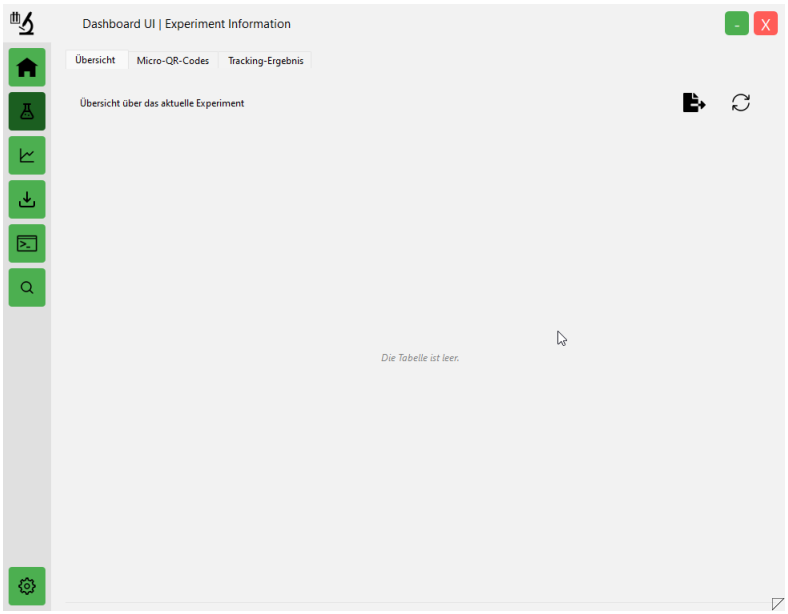


Abbildung 3.2:

3.0.2 Installation

Fehlerbehebung bei ImportError

Wenn ein ImportError auftritt, der auf ein Problem mit der Importierung von QtCore aus PyQt6 hinweist, wie in der folgenden Fehlermeldung dargestellt:

```
Traceback (most recent call last):
File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.2032.0_x64
return _run_code(code, main_globals, None,
File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.2032.0_x64
exec(code, run_globals)
File "C:\Users\User\Documents\bachelorarbeit\Test Delete Agile Robotics Systems Labor
from PyQt6 import QtCore
ImportError: DLL load failed while importing QtCore: The specified procedure could no
Process finished with exit code 1
```

Führen Sie die folgenden Schritte zur Fehlerbehebung durch:

1. Deinstallieren Sie pyqt6 und pyside6.

```
1    pip uninstall PyQt6
2    pip uninstall PySide6
3
```

Listing 3.1: Install PyQt6 & PySide6

2. Installieren Sie pyqt6 und pyside6 erneut.

```
1    pip install PyQt6
2    pip install PySide6
3
```

Listing 3.2: Uninstall PyQt6 & PySide6

3.1 Anhang

3.1.1 Codeausschnitte und Beispiele

3.1.2 Zusätzliche Ressourcen

Literatur

[1] How-To Geek. *How to Edit Environment Variables on Windows 10 or 11*. 17. Nov. 2022. URL: https://www.howtogeek.com/787217/how-to-edit-environment-variables-on-windows-10-or-11/.

[2] Hessisches Kultusministerium. *Landesabitur Informatik Glossar*. 2021. URL: https://kultusministerium.hessen.de/sites/kultusministerium.hessen.de/files/2021-10/la-informatik-glossar.pdf.

[3] Riverbank Computing Limited. *PyQt6 Documentation*. Accessed on February 6, 2024. Riverbank Computing Limited. 2024. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/.

[4] Ujwal Subedi. *Optimization of Laboratory Processes through a GUI Dashboard: Real-Time Monitoring and Management of the Automated Laboratory Line*. Bachelorarbeit. Bachelor's Thesis. Germany, Feb. 2024.

Glossar

E&T

Erfassung und Tracking bezieht sich auf eine Anwendung, die zur Überwachung der Laborstraße mit 2 Kameras entwickelt wurde. Diese Anwendung sammelt Daten von den Laborprozessen, speichert sie in einer Protokolldatei und sendet diese Informationen auch an das GUI Dashboard. Dies ermöglicht die Überwachung und Verfolgung von Laboraktivitäten und Experimenten in Echtzeit.. 1, 22

Echtzeitüberwachung, Live

In diesem Zusammenhang bezieht sich die Echtzeitüberwachung auf die kontinuierliche Beobachtung und Analyse des Status und des Fortschritts der Röhrchen innerhalb der Laborstraße, wobei verfolgt wird, welches Röhrchen welche Station erreicht hat. Bei diesem Prozess werden die Leistung und die Abläufe des Systems in Echtzeit bewertet, so dass genaue und aktuelle Informationen über die Bewegung und die Prüfphasen der einzelnen Röhrchen im Experiment gewährleistet sind.. 1

Experiment

Im Zusammenhang mit diesem Projekt bezieht sich ein "Experiment" auf einen Prozess, bei dem Plasmide zu Testzwecken in Tubes gefüllt werden. Jedes Experiment kann bis zu 32 Röhrchen umfassen. Diese Röhrchen werden systematisch über verschiedene Stationen innerhalb einer Laborstraße bewegt, wo sie verschiedenen Tests und Verfahren unterzogen werden.. 1, 14–16, 18, 19

Experiment ID

In diesem Kontext ist die Experiment-ID ein eindeutiger Identifikator oder Code, der einem Experiment zugewiesen wird. Sie wird aus dem Nachnamen des Laboranten, der Anzahl der von dieser Person durchgeführten Experimente und dem aktuellen Datum gebildet. Die Formel für die Erstellung einer Experiment-ID lautet: Nachname + Anzahl der Experimente (Nachname) + Datum. Diese Kennung ist für die eindeutige Identifizierung, Verfolgung und Referenzierung des Experiments innerhalb der Laborinformations- und -verwaltungssysteme von entscheidender Bedeutung. . 1, 22

GUI Dashboard

In diesem Zusammenhang ist das GUI Dashboard eine grafische Benutzeroberfläche, die es den Benutzern ermöglicht, automatisierte Systeme wie Laborprozesse in Echtzeit zu steuern und zu überwachen. Es erleichtert die Vorbereitung von Experimenten, einschließlich der Anordnung und Verfolgung von Experimentkomponenten, und bietet Funktionalitäten für die effiziente Suche und Verwaltung dieser Komponenten.. 1

Glossar für das Fach Informatik [2]

Glossar für das Fach Informatik

In der Wissenschaft Informatik werden die Fachbegriffe nicht immer einheitlich verwendet. Zudem gibt es Unterschiede in den Darstellungsformen von Diagrammen. Das folgende Glossar soll diesem Umstand abhelfen und die für die Schulinformatik und das Landesabitur relevanten Fachbegriffe und Darstellungsformen festlegen.

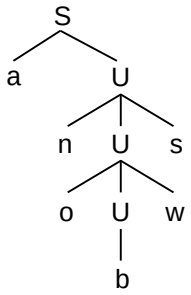
Ableitung

Als Ableitung bezeichnet man für eine gegebene Grammatik die schrittweise Bildung eines Wortes aus Terminalzeichen. Ausgehend vom Startsymbol wird bei jedem Schritt eine Produktion angewendet.

Für die unten angegebene Grammatik ist Folgendes eine Ableitung:
S → aU → anUs → anoUws → anobws

Ableitungsbaum

Für eine kontextfreie Grammatik kann eine Ableitung strukturiert als Ableitungsbaum dargestellt werden. Die Wurzel des Ableitungsbaumes ist das Startsymbol S, die inneren Knoten bestehen aus Nicht-Terminalen und die Blätter aus Terminalen. Die Anwendung einer Produktion wird im Ableitungsbaum durch den Produktionskopf als Elternknoten und den Produktionsrumpf als zugehörige Kindknoten dargestellt. Das obige Ableitungsbeispiel ergibt den dargestellten Ableitungsbaum.

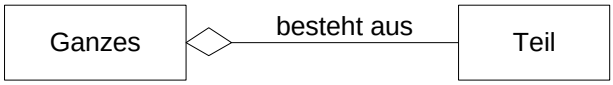


abstrakte Klasse/Methode

Mit einer abstrakten Klasse kann in einer Klassenhierarchie eine Oberklasse modelliert werden, welche abstrakte Methoden enthält, die erst in abgeleiteten Unterklassen implementiert werden. Die abstrakte Oberklasse *GeometrischeFigur* kann beispielsweise die abstrakte Methode *berechneFläche()* definieren, die jeweils in den beiden Unterklassen *Rechteck* und *Kreis* implementiert wird.

Aggregation – die *besteht aus*-Beziehung

Die Aggregation ist eine Sonderform der Assoziation zwischen zwei Klassen. Sie liegt dann vor, wenn zwischen den Objekten der beteiligten Klassen eine Beziehung existiert, die sich als „besteht aus“ oder „ist Teil von“ beschreiben lässt. In der UML-Darstellung wird die Aggregatklasse mit einer Raute versehen. Die Raute symbolisiert das Behälterobjekt, in dem die Teile gesammelt werden.



Akzeptor

Ein Akzeptor besteht aus fünf Bestandteilen:
- dem Eingabealphabet ∑, einer endlichen Menge von Zeichen
- der Zustandsmenge Z, einer endlichen Menge von Zuständen
- dem Startzustand z_0, einem Element aus Z
- der Menge der Endzustände E, einer Teilmenge von Z
- der Übergangsfunktion δ: Z x ∑ → Z, die für Paare aus Zustand und Zeichen festlegt, in welchen Folgezustand der Akzeptor übergeht.

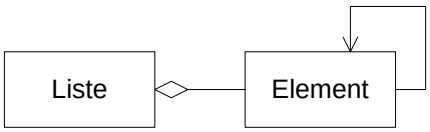
Ein Wort w wird akzeptiert, wenn ausgehend vom Startzustand z_0 schrittweise für jedes Zeichen des Wortes ein Übergang in einen Folgezustand erfolgt und nach Abarbeitung aller Zeichen ein Endzustand erreicht ist.

Assoziation – die *kennt*-Beziehung

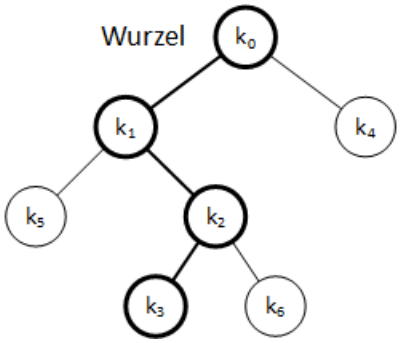
Eine Assoziation beschreibt eine Beziehung zwischen zwei Klassen. Mit Hilfe einer gerichteten Assoziation kann dargestellt werden, dass diese Beziehung nur in einer Richtung existiert. Grafisch wird die ungerichtete Assoziation als Strecke und die gerichtete Assoziation als Pfeil dargestellt. Im Unterschied zur bidirektionalen Datenmodellierung im ER-Modell wird bei der objektorientierten Modellierung in der Regel mit gerichteten Assoziationen gearbeitet.

Eine Assoziation heißt **rekursiv**, wenn die beiden beteiligten Klassen gleich sind.

Beispiel: Eine lineare Liste besteht aus Elementen (Aggregation), wobei jedes Element mit Ausnahme des letzten auf das nachfolgende Element verweist (rekursive Assoziation).



Baum



Ein Baum besteht aus Knoten und Kanten. Ein einziger Knoten ist als Wurzel des Baumes dadurch ausgezeichnet, dass er keinen Elternknoten hat. Alle anderen Knoten sind Kindknoten und direkt durch eine Kante mit ihrem Elternknoten verbunden. Eine Folge $k_0, k_1, ..., k_n$ von Knoten eines Baumes derart, dass stets k_{i+1} Kindknoten von k_i ist, wird als Pfad der Länge n bezeichnet. Knoten, die keine Kinder haben, werden als Blätter bezeichnet; alle anderen Knoten heißen innere Knoten. Die Höhe eines Baumes ist die Länge des längsten Pfades von der Wurzel zu einem Blatt. Die Tiefe eines Knotens ist die Länge eines Pfades von der Wurzel zum Knoten.

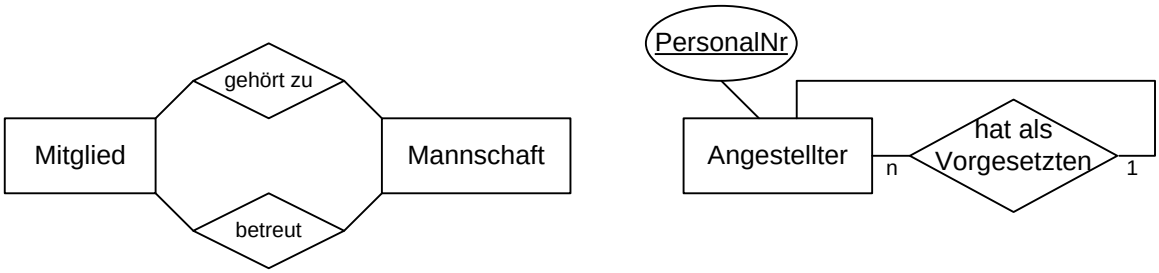
Der im Bild dargestellte Baum besteht aus 7 Knoten und 6 Kanten. Er hat die Wurzel k_0. Fett eingezeichnet ist ein Pfad mit maximaler Länge von der Wurzel bis zum Blatt k_3. Die Höhe des Baumes ist also 3 und die Tiefe des Knotens k_5 ist 2.

Beziehung und Beziehungstyp

Zwei Entitäten können in einer Beziehung stehen, z. B. die Angestellte Müller leitet die Abteilung Leichtathletik.

Ein binärer Beziehungstyp stellt eine Beziehung zwischen zwei Entitätstypen *A* und *B* dar. Die Beziehung besteht in den beiden Richtungen $A \rightarrow B$ und $B \rightarrow A$. Daher werden Kardinalität (1:1, 1:n, n:m) und Optionalität (kann, muss) eines Beziehungstyps durch jeweils zwei Angaben beschrieben. Eigentlich müsste auch die Bezeichnung des Beziehungstyps in beiden Richtungen angegeben werden, doch üblicherweise gibt man sie nur in der Leserichtung von links nach rechts, bzw. von oben nach unten an. Im ER-Diagramm wird ein Beziehungstyp als Raute dargestellt, die den Namen des Beziehungstyps enthält.

Zwischen zwei Entitätstypen können mehrere Beziehungen bestehen. Im Beispiel gehören Mitglieder eines Vereins zu Mannschaften und jede Mannschaft wird auch von einem Vereinsmitglied betreut.



Wenn ein Entitätstyp mit sich selbst in Beziehung steht, spricht man von einer rekursiven Beziehung. Im Beispiel haben Angestellte einer Firma einen anderen Angestellten als Vorgesetzten. Bei der Abbildung ins Relationenmodell entsteht ein Namenskonflikt, weil das Attribut *PersonalNr* sowohl als Primärschlüssel als auch als Fremdschlüssel in der Relation erscheint. Daher wird der Fremdschlüssel gemäß der Beziehung umbenannt.

Relation: Angestellter(<u>PersonalNr</u>, ↑VorgesetztenPersonalNr, …)

Datenkapselung

siehe **Geheimnisprinzip**

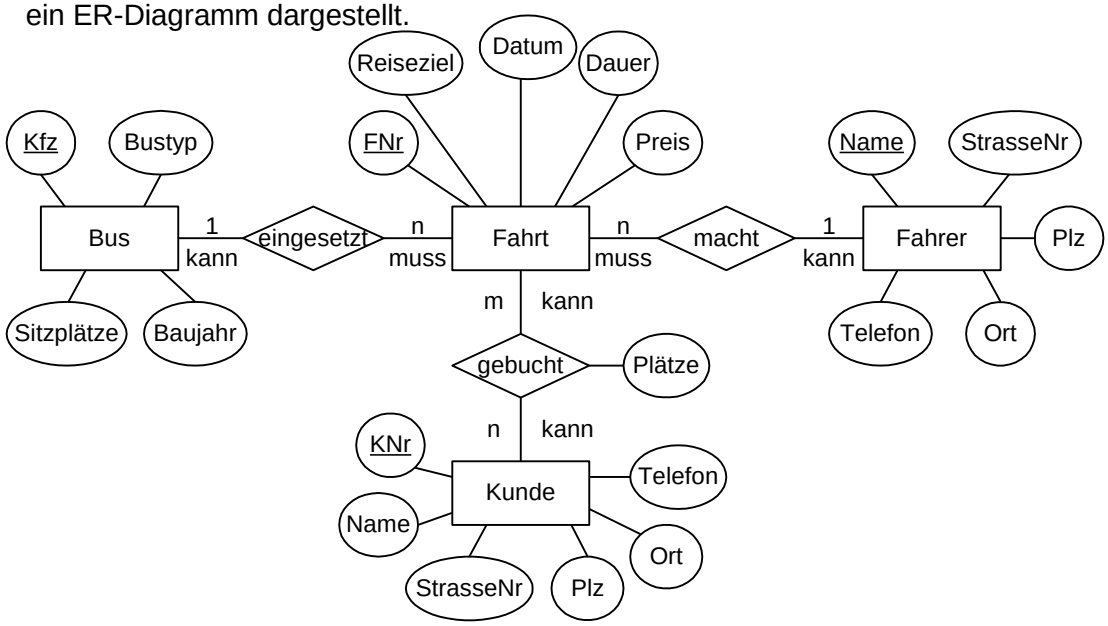
Entität und Entitätstyp

In der Datenmodellierung wird ein Objekt der realen Welt als Entität (engl.: entity) modelliert. Eine Entität kann eine Person, ein Gegenstand, ein Prozess oder auch ein nicht materielles Ding sein.

Gleichartige Entitäten bilden einen Entitätstyp (engl.: entity type) z. B. alle Angestellten, Bücher oder Reservierungen. Der Name eines Entitätstyps ist ein Substantiv im Singular. Die Eigenschaften eines Entitätstyps werden durch Attribute beschrieben.
Im ER-Diagramm wird ein Entitätstyp durch ein Rechteck dargestellt, das den Namen des Entitätstyps enthält.

ER-Diagramm und ER-Modell

Die bei der datenorientierten Modellierung eines Ausschnitts der realen Welt entstehenden Entitäts- und Beziehungstypen bilden das Entity-Relationship-Modell (ER-Modell oder ERM) und werden in einem Entity-Relationship-Diagramm (ER-Diagramm oder ERD) dargestellt. Ein Entitätstyp wird durch ein Rechteck, Attribute durch Ovale und ein Beziehungstyp durch eine Raute dargestellt. Die Kardinalität eines Beziehungstyps wird im ER-Diagramm durch 1:1, 1:n bzw. n:m und die Optionalität durch „kann“ bzw. „muss“ angegeben. Exemplarisch ist nachfolgend ein ER-Diagramm dargestellt.



Fachkonzept

Das Fachkonzept ist eine zusammenfassende Darstellung eines Anwendungssystems aus fachlicher Sicht. Es besteht aus dem im Rahmen der objektorientierten Analyse entstandenen Klassendiagramm, mit Berücksichtigung aller fachlichen Aspekte des zu entwickelnden IT-Systems, ohne grafische Benutzungsoberfläche (GUI) und Datenhaltung in Datenbanken.

Geheimnisprinzip

Das Geheimnisprinzip besagt, dass die Implementierungsdetails einer Klasse verborgen werden sollen. Es wird als Datenkapselung umgesetzt, indem Attribute einer Klasse die Sichtbarkeit *private* oder *protected* erhalten und somit von außen nicht direkt zugreifbar sind. Auch der Zugriff auf Methoden kann so verhindert werden. Die Attributwerte können von außen über Methoden mit der Sichtbarkeit *public,* also über die öffentliche Schnittstelle, oder bei abgeleiteten Klassen mit Methoden der Sichtbarkeit *protected* abgefragt bzw. verändert werden.

Glossar Molekulargenetik

Adenin	organische Purinbase, Bestandteil von Nukleotiden
Adenosin	AMP ohne P, Verbindung von Pentose und Adenin
allgemeine Transduktion	Übertragung beliebiger Wirtsgene durch Viren
Anaphase	3. Phase der Mitose, Auseinanderweichen der Chromatiden
Antibiotika	sind Stoffe, die entweder das Wachstum von Bakterien hemmen oder sie abtöten
Bakteriostatika	Antibiotika die wachstumshemmend sind
Bakterizide	Antibiotika, die Bakterien töten
Basenpaarung	in Nukleinsäuren verbinden sich gegenüberliegende Basen mit Wasserstoffbrücken
Capping	Anhängen einer Basensequenz an das 5´-Ende der mRNA bei der Reifung zur Bindung an das Ribosom
Centromer	Verengung im Chromosom, Spindelfaseransatzstelle
Chromatiden	Chromosomenhälften, in der Metaphase sichtbar
Chromatin	Komplex von DNA und Protein im Zellkern der Eukaryonten
Chromosomen	X-förmige Gebilde, die während der Zellteilung sichtbar werden und aus DNA und Protein bestehen.
Chromosomenmutationen	Mutation, die einen Defekt (Form usw.) am Chromosom betrifft
codogener Strang	der Strang der Doppelhelix, der die genetische Information enthält
Codon	Triplett auf der mRNA mit Info für Aminosäure
Cytosin	organische Pyrimidinbase, Bestandteil von Nukleotiden
degeneriert	der genetische Code ist nicht eindeutig, es gibt mehrere Tripletts für einen Aminosäure
Deletion	Mutation bei der Chromosomenabschnitte fehlen
DNA	doppelsträngiges Riesenmolekül im Zellkern der eukaryontischen Zellen und im Cytoplasma der Bakterienzellen; hat die Erbinformation gespeichert.
DNA-Polymerase	Enzym, das bei der Replikation zu beiden DNA-Strängen einen komplementären Strang synthetisiert
DNA-Fingerprint	Verfahren zum Vergleich von DNA-Fragmenten zur Identifizierung des DNA-Trägers
Down-Syndrom	Erbkrankheit durch Trisomie am 21.Chromosom verursacht
Exon	GenCode enthaltende, durch Introns unterbrochene mRNA-Sequenz
Flavr Savr	genetisch manipulierte Tomate, die länger lagern kann
Gelelektrophorese	Trennmethode für geladene Biomoleküle mit Hilfe von Gelen und Strom.
genetischer Fingerabdruck	Methode zur Analyse und Vergleich Individuum-spezifischer DNA als Merkmal

Genom	Gesamtheit der Erbinformation einer Zelle
Genmutation	Mutation im Gen
Genommutation	Mutation, die die Anzahl der Chromosomen betrifft
Gen-Therapie	ersetzen von defekten Genen bei erbkranken Individuen
Guanin	organische Purinbase, Bestandteil der Nukleotide
Hemmhof	Zone um ein auf Petrischalen aufgebrachtes Antibiotikum, in der keine Bakterien wachsen
Interphase	Phase zwischen zwei Zellteilungen einer eukaryontischen Zelle
Intron	in den fortlaufenden Gencode der mRNA eingestreute Nonsenssequenzen, die bei der Reifung herausgeschnitten werden
klebrige Enden	einsträngiger DNA-Abschnitt, der beim Durchschneiden mit Restriktionsenzymen entsteht
Klon	Gruppe erbgleicher Organismen
Konjugation	Pseudosexualvorgang bei Bakterien mit Plasmid oder Genaustausch
Lysozym	Antibiotikum der Säugetiere in der Tränenflüssigkeit; hemmt Zellwandsynthese
Meselson + Stahl	bewiesen durch ein cleveres Experiment (Dichtegradientenzentrifugation) die semikonservative Replikation
Metaphase	2. Phase der Mitose, Anordnung der Chromosomen in der Äquatorialebene
Mitose	Zellteilung der Eukaryonten wobei 2 genetische identische Zellen entstehen, läuft in Prophase, Metaphase, Anaphase und Telophase, M-Phase des Zellzyklus
Mutationsrate	Häufigkeit der Mutationen in einer Population
mRNA	entsteht als Genabschrift bei der Transkription im Zellkern
mRNA-Reifung	Bearbeitung der abgeschriebenen mRNA zur endgültigen Fassung unter Herausschneiden der Introns
Mutagen	mutationsauslösender Faktor
Mutation	spontane oder induzierte Erbänderung
Nonsens-Mutation	Genmutation, bei der ein Stopp-Codon entsteht
Nukleotide	Bausteine der Nukleinsäuren; sie bestehen aus Phosphorsäure, einer Pentose und einer organ. Base
Okazaki-Fragmente	mehrere kleine Nukleotid-Sequenzen, durch die DNA-Polymerase im 2. Abschnitt der Replikation gebildet werden
Onkogene	Gene, die Krebs hervorrufen
Penicillin	bakteriostatisches Antibiotikum des Schimmelpilzes Penicillium
Phagen	Viren, die Bakterienzellen befallen
Plasmide	kleine DNA-Ringe in Bakterien
Polyploidie	mehrfache Vervielfachung des Chromosomensatzes

Polymerase-Kettenreaktion	Methode zur schnellen Vervielfachung von DNA-Sequenzen
Polysomen	an einer mRNA aufgereihte Ribosomen
Primer	kurze RNA-Sequenz, die bei der Replikation an der Replikationsgabel von der Primase komplementär synthetisiert wird
Prophage	ins Wirtsgenom integrierte Phagen- DNA
Prophase	1. Phase der Mitose, Verkürzung und Verdickung der Chromatinfäden, Teilung des Centriols
Punktmutation	Mutation durch Änderung einer Base in der DNA
Purinbasen	Adenin und Guanin, leiten sich vom Purin ab
Pyrimdinbasen	Cytosin, Thymin und Uracil sind vom Pyrimidin abgeleitet
Raster-Mutation	Genmutation, bei der durch Ausfall oder Einfügen einer Base das Triplettraster verändert wird und ab dieser Stelle ein Protein mit einer anderen Aminosäuresequenz entsteht
R-Faktor	Resistenzgene auf Plasmiden
Restriktionsenzyme	Enzyme, die DNA gezielt in Fragmente schneiden können
Replikation	Verdopplung der DNA in der S-Phase des Zellzyklus (Interphase)
Resistenz	erbliche Fähigkeit der Bakterien und Pilze unempfindlich gegen Antibiotika zu sein
Restriktions-Endonukleasen	Enzyme in Bakterien, die die DNA spezifisch durchschneiden
RNA	Ribonukleinsäure, kommt als mRNA, tRNA und rRNA vor, enthält Uracil statt Thymin, ist meist einsträngig
RNA-Polymerase	schreibt im Zellkern Gene in mRNA ab
RNA-Splicing	Herausschneiden der Intron aus der prä-mRNA
rRNA	Ribosomen bestehen aus rRNA
Sichelzellanämie	Blutkrankheit die homozygot wegen defekten Hämoglobins zum Tod führt (Genmutation)
semikonservative Replikation	die DNA wird dadurch repliziert, dass je ein komplementärer Strang zu einem alten Strang synthetisiert wird, somit ist die neue Doppelhelix halb alt, halb neu.
Starter-Codogen	Triplett am Anfang des Gens, das den Start der Transkription an der DNA bestimmt
Telophase	4. Phase der Mitose, Bildung neuer Kernmembranen, Zellteilung
temperent	Phage, dessen DNA sich ins Wirtsgenom integrieren kann
Terminierung	Endphase der Translation; Ablösen der Ribosomen von der mRNA
Tetrazykline	bakterizide Antibiotika der Streptomyceten, die die Proteinsynthese hemmen
Thymin	organische Pyrimidinbase, Bestandteil der DNA-Nukleotide
Transduktion	Übertragung von Genen durch Viren auf andere Zellen

transgen	Organismus enthält fremde Gene
Transformation	Aufnahme von DNA durch Bakterien aus der Umgebung
Transkription	Abschreiben des Gens in mRNA
tRNA	Trägermolekül für Aminosäuren bei der Translation mit Anticodon für mRNA
universell	der genetische Code gilt praktisch bei allen Organismen
Uracil	Pyrimidinbase aus Ribonukleinsäure
Watson+Crick	klärten 1953 die Doppelhelixstruktur der DNA auf
Zellzyklus	Zyklus von Wachstum, DNA-Verdopplung, Wachstum und Zellteilung bei Eukaryonten