UNIKASSEL
VERSITÄT

Seminararbeit

Von C# nach Python: Software-Konzeptionierung einer robotergestützten Lagerverwaltung

Analyse bestehender Software und Konzeptionierung einer integrierten Python-Anwendung mit kameragestützten Validierungsprozessen in der Industrie 4.0-Plattform Modellfabrik μPlant

Lennart Schink

Matrikelnummer:	33237484
Gutachter:	Univ.-Prof. Dr.-Ing. Andreas Kroll
Betreuer:	Dip.-Ing. Axel Dürrbaum
Tag der Abgabe:	10. Juni 2023
MRT-Nr.:	N.N

mrt

isac

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Symbolverzeichnis

Symbol	Bedeutung	Einheit	Bemerkung
π	Kreiszahl		
c	Lichgeschwindigkeit	m/s^2	im Vakkum
g	Erdbeschleunigung		
g	Erdbeschleunigung	m/s^2	

Index

1 Motivation und Zielsetzung

Das Institut für Mess- und Regelungstechnik an der Universität Kassel hat in den letzten Jahren eine Modellfabrik μPlant gebaut. Aus über 70 Einzelarbeiten ist ein modernes Industrie-4.0 Konzept geschaffen worden. Teil der μPlant ist ein vollautomatisiertes Lager. Das Lager besteht aus einem abgetrennten Raum, dessen Zugang über eine Tür mit einem Türschalter überwacht ist. In diesen Bereich können Turtlebots einfahren. In dem abgetrennten Bereich steht ein Industrieroboter und ein Lagerregal mit ausgewiesenen 18 Lagerplätzen. Außerdem befindet sich neben einer Andockstation für den Turtlebot auch noch eine Werkbank.

Ein pneumatischer Greifer des Industrieroboters kann Paletten, die je mit bis zu zwei Bechern bestückt werden können, zwischen dem mobilen Roboter und dem Lagerregal frei bewegen. Von einem PC-Arbeitsplatz aus können mittels Software die Lagerprozesse überwacht werden. Außerdem kann im Fehlerfall eingeschritten werden und es können manuell Prozesse ausgelöst werden.

Die Software ist derzeit in 3 Teile aufgeteilt: Einerseits gibt es die Lagerverwaltung - die Hauptsoftware. Sie bildet die automatisierten Prozesse ab und verfügt über ein GUI welches u.A. den Bestand visualisiert. Daneben gibt es den Warehouse Controller, der dazu verwendet wird manuell Lagerprozesse auszulösen, und ein RFID-Tool was für manuelle RFID - Prozesse benutzt wird.

Mit dem Wechsel des Betriebssystems von Windows 7 auf Windows 10 ist die Kompatibilität der in C# implementierten Software nicht mehr gegeben. Außerdem laufen Teilfunktionen des Programms nicht fehlerfrei oder tolerieren kaum Fehlbedienungen. Die Dreiteilung der Software ist im Allgemeinen auch nicht mehr erwünscht.

Diese Seminararbeit beschäftigt sich mit der Analyse der bestehenden Software: Es wird ermittelt, aus welchen Programmteilen und Funktionen die Software besteht. Aus den Erkenntnissen wird ein Konzept entwickelt wie die Drei Software Teile zusammengeführt werden könnten um so die Grundlage für eine Migration der Software nach Python zu schaffen.

Erkenntnisse aus der studentische Arbeit von [Hügler] sollen überprüft und vertieft werden um Anforderungen an Kameras und arUco Marker zu ermitteln, die später eine automatisierte Inventur ermöglicehn sollen.

2 Softwarearchitektur der bestehenden Software

Analysemethoden der Informatik für Software sind in der Regel für die verschiedenen Design-Phasen einer Software entwickelt worden. Eine von mir durchgeführte Recherche ergab, dass sich Analyse-Tools und Methoden für bestehende Software vor Allem darauf fokussieren die Performance, Speichermanagement und Benutzererfahrung zu bewerten. Die Architektur einer Software spielt dabei eine untergeordnete Rolle. Für die Neuentwicklung der bestehenden Software werden im Folgenden die Klassen und Objekte sowie ihre Wechselwirkungen dargestellt und anschließend analysiert und bewertet.
In einem C# - Projekt sind UI und Logik in getrennten Dateien implementiert. XAML-Dateien sind eine angepasste Form von XML-Dateien. Sie legen fest wie etwas gerendert wird während XAML.CS- Dateien die dahinterliegende Buisiness-Logik abbilden. Am Beispiel des Haupt-GUI des Programms soll der prinzipielle Aufbau gezeigt werden. Wegen der Komplexizität wird der Umfang jedoch nicht für das gesamte Programm fortgesetzt.

2.0.1. Lagerverwaltung 3.0

- Die Datei `App.xaml.cs` ist der Einstiegspunkt des Programms.
- Die Datei `App.xaml` erzeugt ein Objekt der Apllication Klasse und erzeugt ein Dictionary mit den Ressourcen der Anwendung. Als Startup-URL ist die Datei `MainWindow.xaml` festgelegt.
- In `MainWindow.xaml` wird:
  - Das Hauptfenster gerendert.
  - Objekte und Variablen initialisiert, die dem Lager zugeordnet sind:
    * Ein Objekt `inventory` der Klasse `Inventory` für das Inventar mit `null` initialisiert.
    * Ein Objekt `storageMatrix` von der Klasse `PalletMatrix` erzeugt.
    * Ein Object `commissionMatrix` von der Klasse `PalletMatrix` erzeugt.
    * Ein Objekt `mobileRobot` von der Klasse `MobileRobot` erzeugt.
    * Außerdem eine Variable `lastCupRead` vom Datentyp `ushort` (16-Bit-Ganzzahl, vorzeichenlos) mit 0 initialisiert.
  - Objekte und Variablen initialisiert, die dem ABB Controller zugeordnet sind:
    * Ein Objekt `commands` von der Klasse `controllerCommandList`.
    * Ein Objekt `controllerProperties` von der Klasse `RobotControllerProperties`.

* Ein Objekt `controllerBase` von der Klasse `RobotControllerBas`, mit dem Initialisierungswert `null`.
* Ein Objekt `controllerSim` von der Klasse `RobotSimulator`.

Im Constructor der Klasse werden der ModBus und der Roboter Controller initialisiert, die Produktlistegeladen und gerendert. Daten des Lagers und des turtlebots sowie Comissionsdaten werden geladen und anschließend das Inventar gerendert.

Abb. 1 zeigt einen Screenshot des Hauptbildschirm des Programms. Es ist in 7 wesentliche Teile eingeteilt:

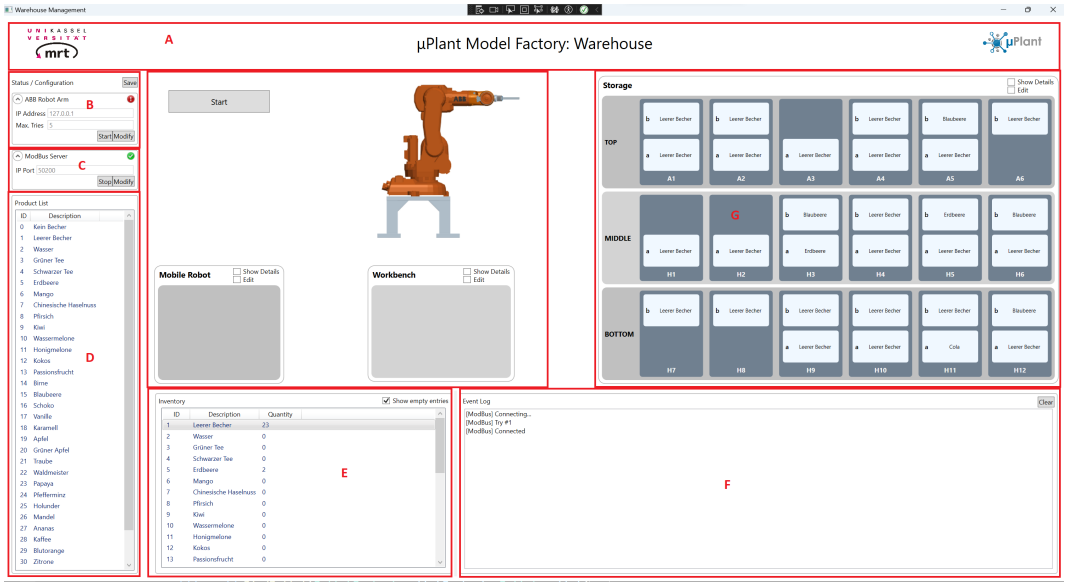


Abbildung 2.1.: Startbildschirm der Anwendung, zur besseren Beschreibung sind die Bedienbereiche rot umrandet und mit Buchstanben gekennzeichnet

- Der rote , mit "1"markierte Bereich im Startbildschirm dient dazu die Verbindungsdaten zum ModBus zu konfigurieren. Über entsprechende Taste kann die Eingabe der beiden Felder aktiviert/deaktiviert werden. Über die Start-Taste wird die VErbindung hergestellt. An der Stelle des sichtbaren roten Ausrufezeichen-Symbol wird ein grüner Haken sichtbar, wenn die Verbindung korrekt hergestellt wurde.
- Der blaue, mit "2"markierte Bereich dient dazu gesondert den ModBus-Port einzustellen.
- Der grüne, mit "3"markierte Bereich ist eine Listenansicht, in der jedes mögliche Produkt mit seiner zugeordneten ID abgebildet ist. Die Liste ermöglicht dem Benutzer einen schnellen Überblick über mögliche Produkte.
- Der gelbe, mit "4"markierte Bereich ist die Übersicht über das aktuelle Geschehen. Im linken, unteren Bereich ist der Lagerplatz des mobilen Roboters symbolisiert. Im rechten, unteren Bereich ist der Lagerplatz der Werkbank symbolisiert. Beide Symboliken verfügen über je einen Lagerplatz, wobei die real doppelte Ausführung der Werkbank nicht umgesetzt wurde.
- die violetten Bereiche "5ünd "6"bilden die Visualisierung des Lagers. In Bereich 6 sind die 18 Lagerslots so dargestellt wie das reale Regal aufgebaut ist. Einzig die Lagerort-Bezeichnung ist anders. Im Bereich "5"werden alle möglichen Produkte

in einer Liste mit der gelagerten Menge angezeigt. Wahlweise können alle Produkte ausgeblendet werden, die keine Bestand haben. Mit Klick auf ein Produkt, egal ob im Bereich "5öder "6", werden alle passenden Einträge farblich hervorgehoben, sodass auf einen Blick erkennbar ist, wo die Lagermenge im Lager wirklich abgebildet ist.

- Im grünen, mit "7"gekennzeichneten Bereich ist ein Eventlogger implementiert. Wann immer ein für den Benutzer relevantes Ereignis eintritt, wird hier eine entsprechende Meldung angezeigt.

Lagerelemente

Die bestehende Software dient dazu Lagerpakete vom mobilen Roboter auf die Werkbank oder ins Lager zu bewegen. Oder alle möglichen Kombinationen davon. Das Konzept der μPlant beschränkt sich dabei auf eine Palette, die so ausgeführt ist, dass der Greifer des Industrieroboters sie sicher bewegen kann. Im Wesentlichen ist es ein Quader mit zwei seitlichen Längsnuten. Eine Palette enthält zwei senkrechte Bohrungen, die es ermöglichen je einen Becher aufzunehmen.
Ein Becher ist ein transparentes, zylindrisches Gefäß aus Acrylglas mit einem Absatz, der etwa mittig in der Höhe angebracht ist. Dadurch können die Becher einzeln vom Greifer bewegt werden.

2.0.2. RFID Server

2.0.3. Controller

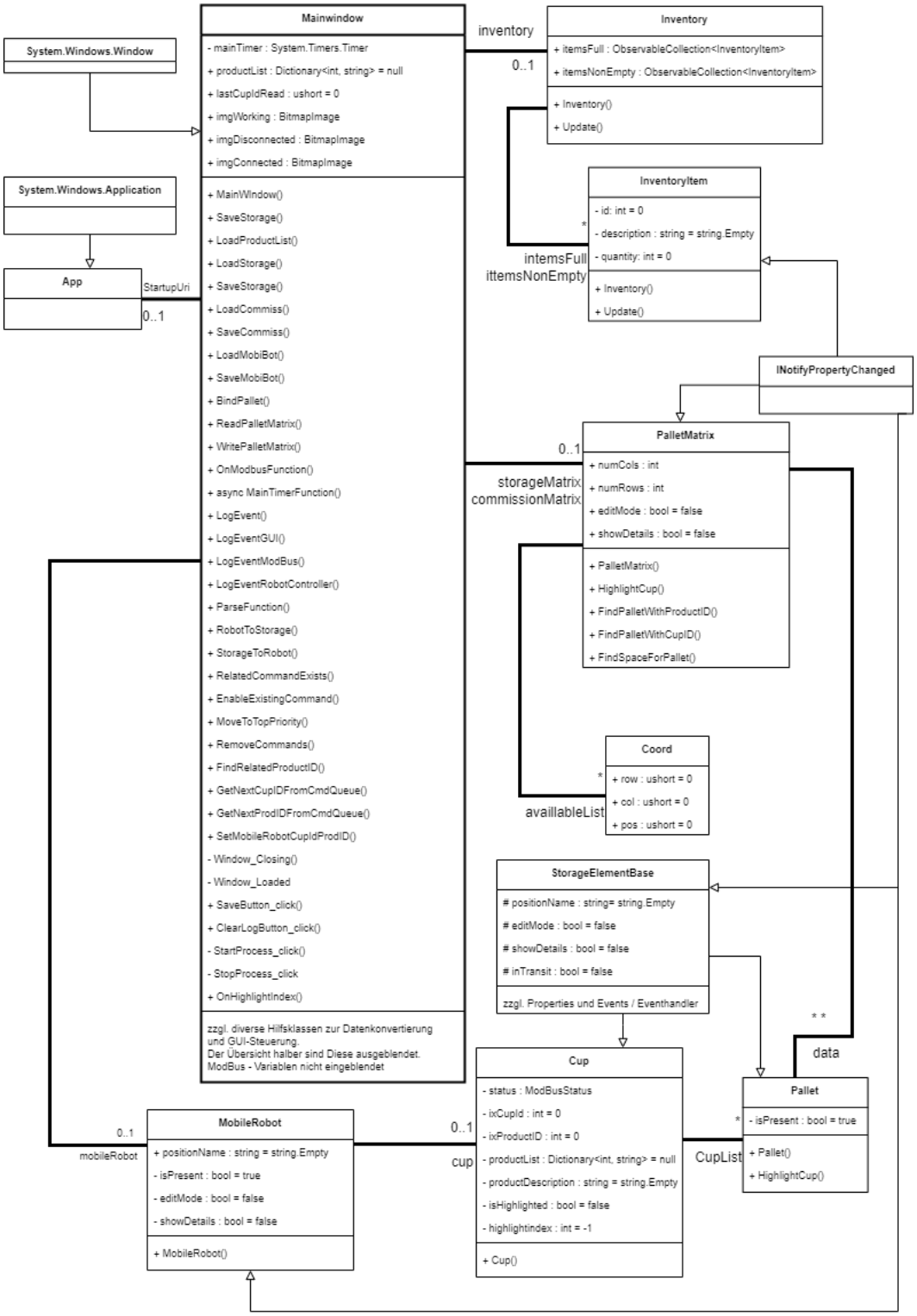


Abbildung 2.2.: Klassendiagramm der MainWindow-Klasse mit vererbenden- und Datenmodell - Klassen

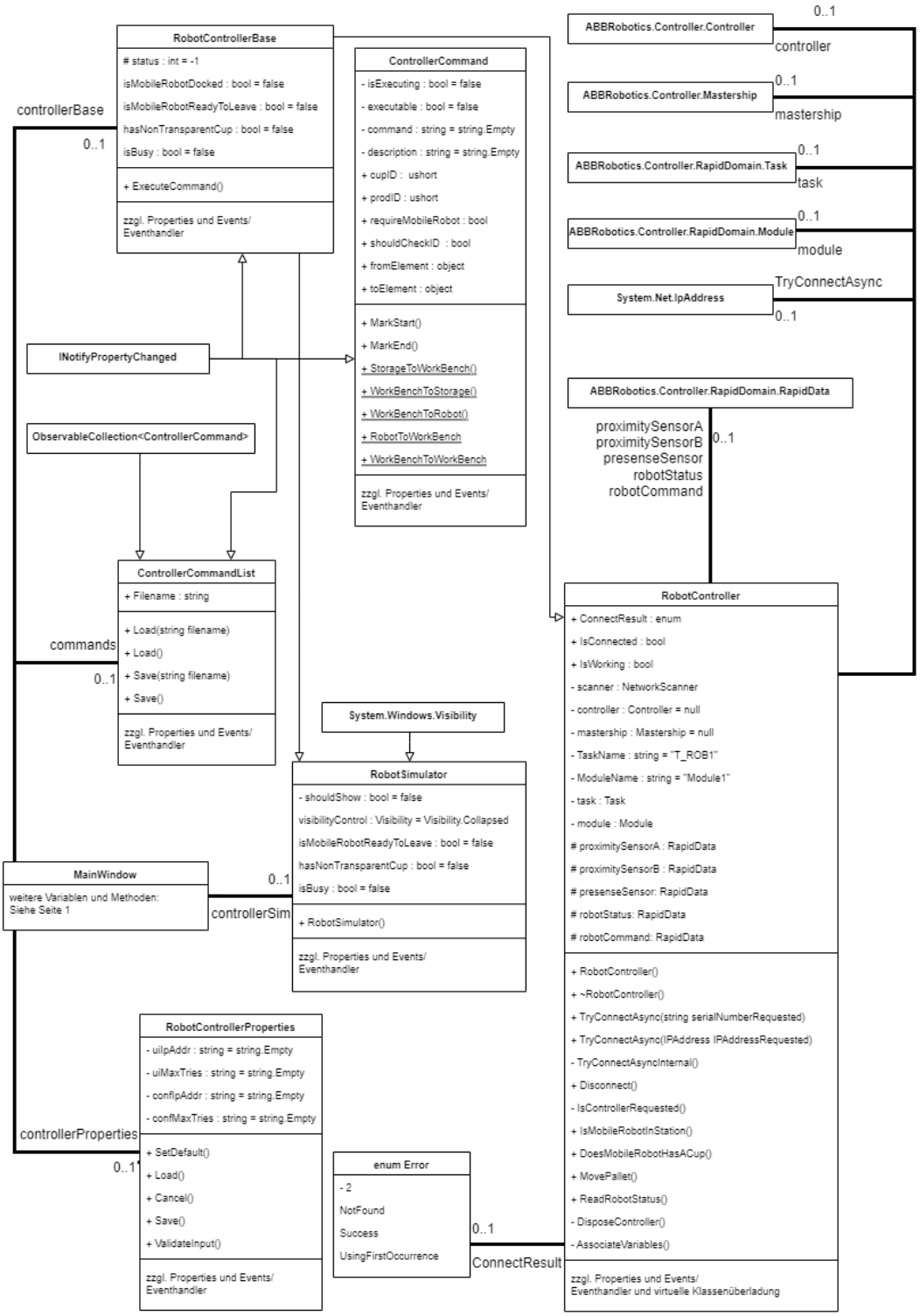


Abbildung 2.3.: Klassendiagramm der MainWindow-Klasse mit Klassen aus dem ABB Controllerframework.

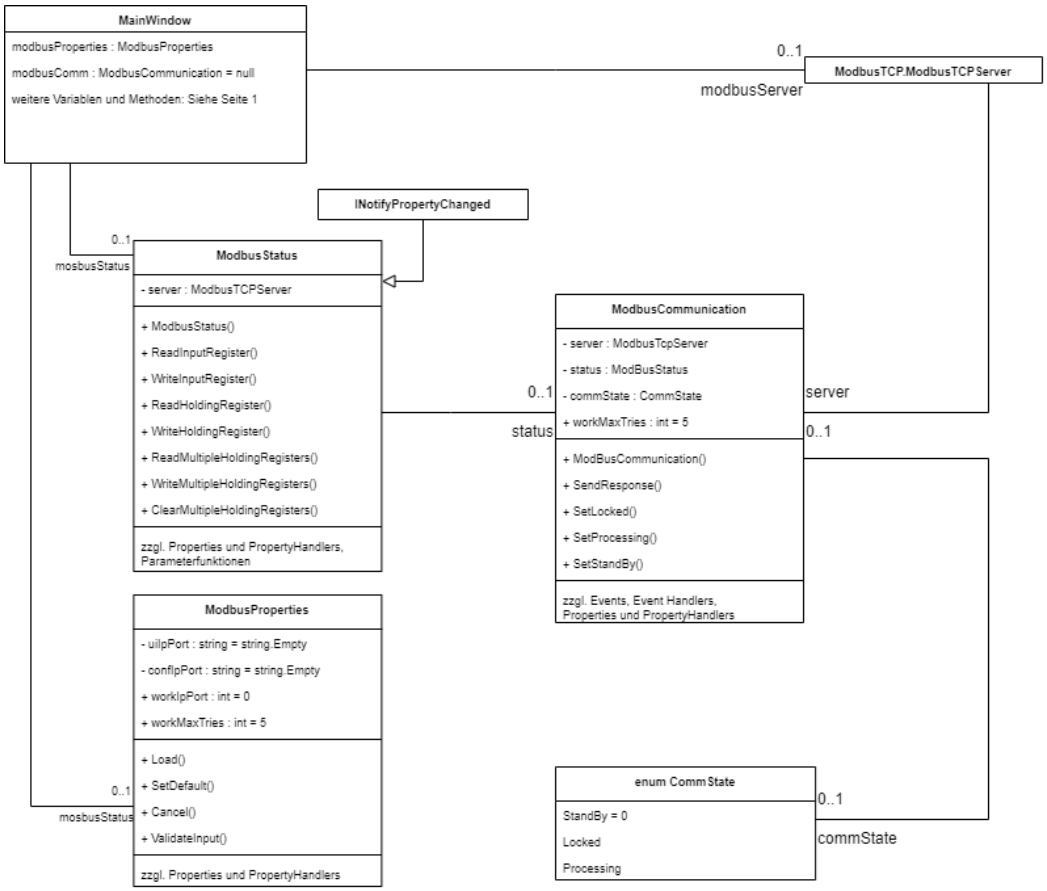
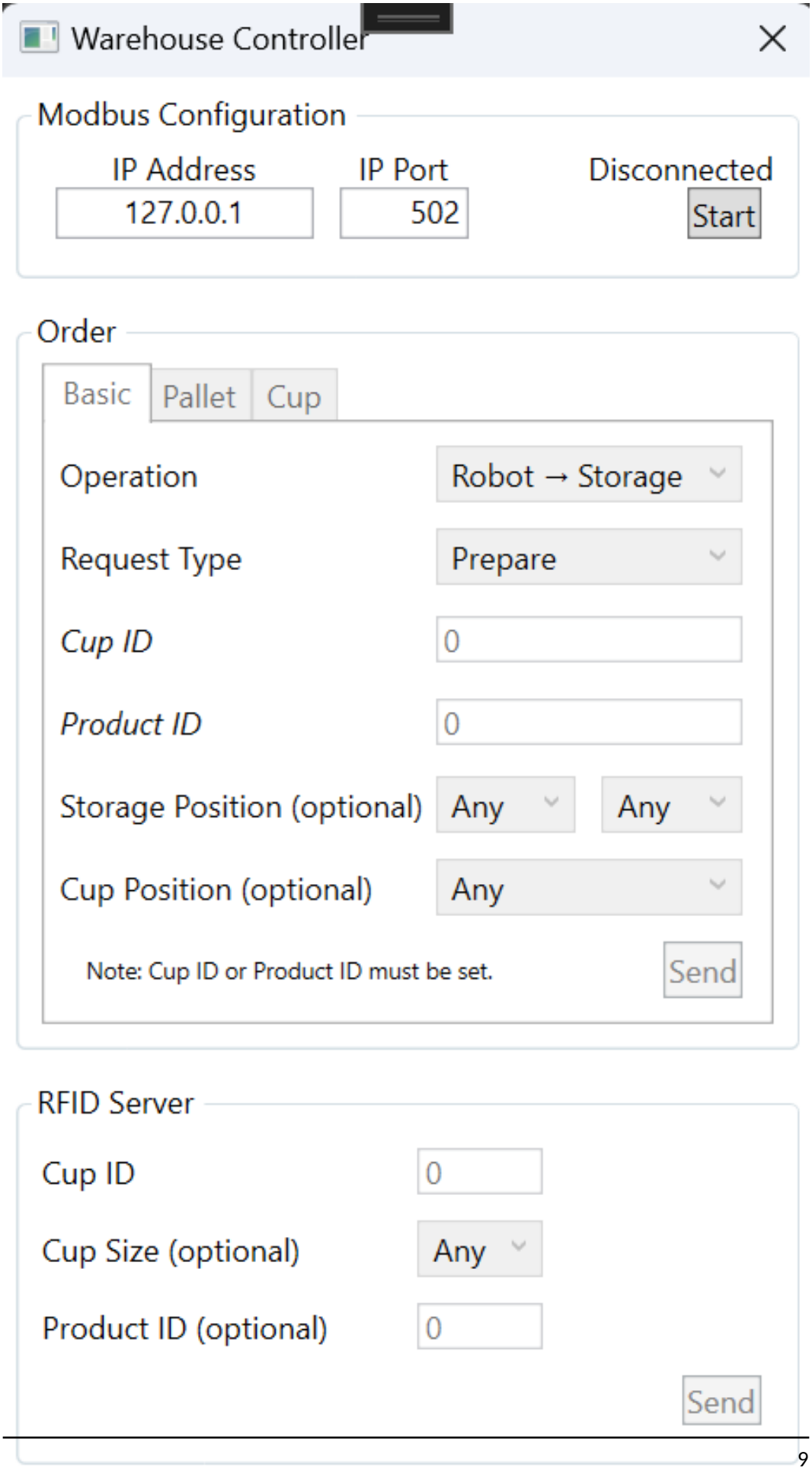


Abbildung 2.4.: Klassendiagramm der MainWindow-Klasse mit Mosbus-Klassen


Warehouse Controller
Modbus Configuration
IP Address
127.0.0.1
IP Port
502
Disconnected
Start
Order
Basic
Pallet
Cup
Operation
Robot → Storage
Request Type
Prepare
Cup ID
0
Product ID
0
Storage Position (optional)
Any
Any
Cup Position (optional)
Any
Note: Cup ID or Product ID must be set.
Send
RFID Server
Cup ID
0
Cup Size (optional)
Any
Product ID (optional)
0
Send
Event Log

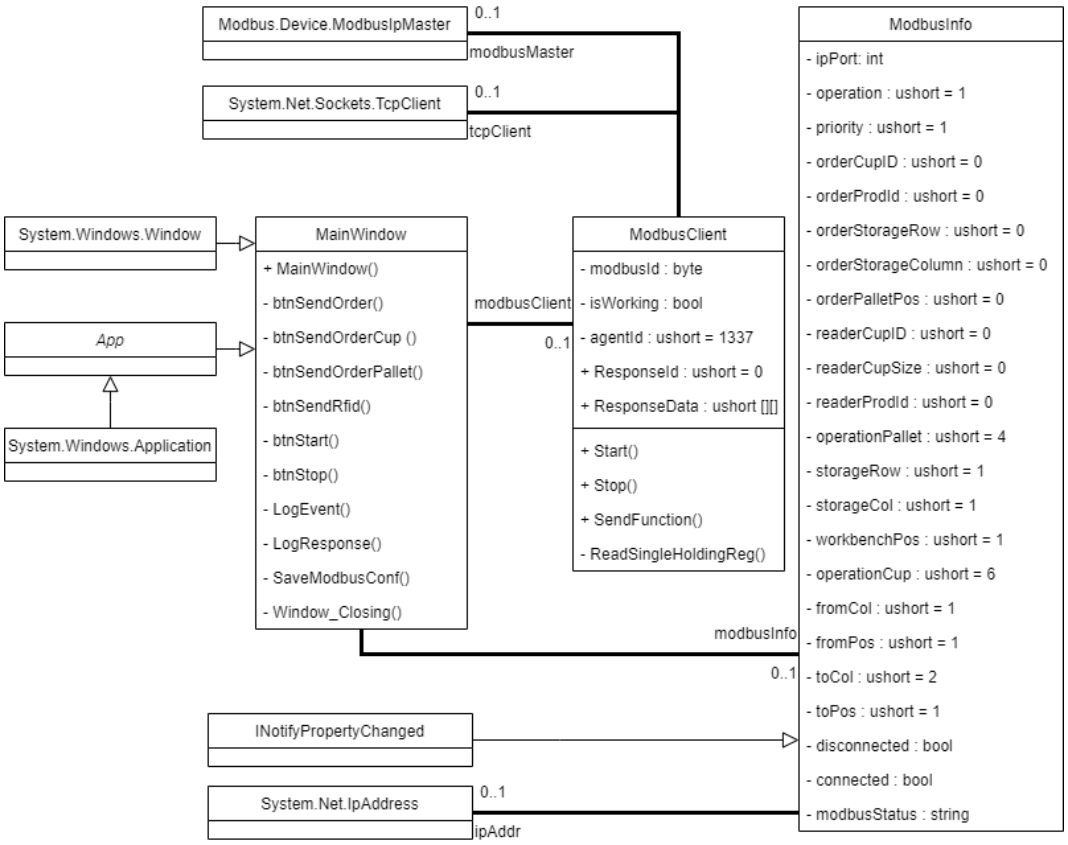


Abbildung 2.6.: Klassendiagramm des Warehouse-Controllers mit allen vererbenden Klassen jedoch ohne Klassen die lediglich Datentypen konvertieren.

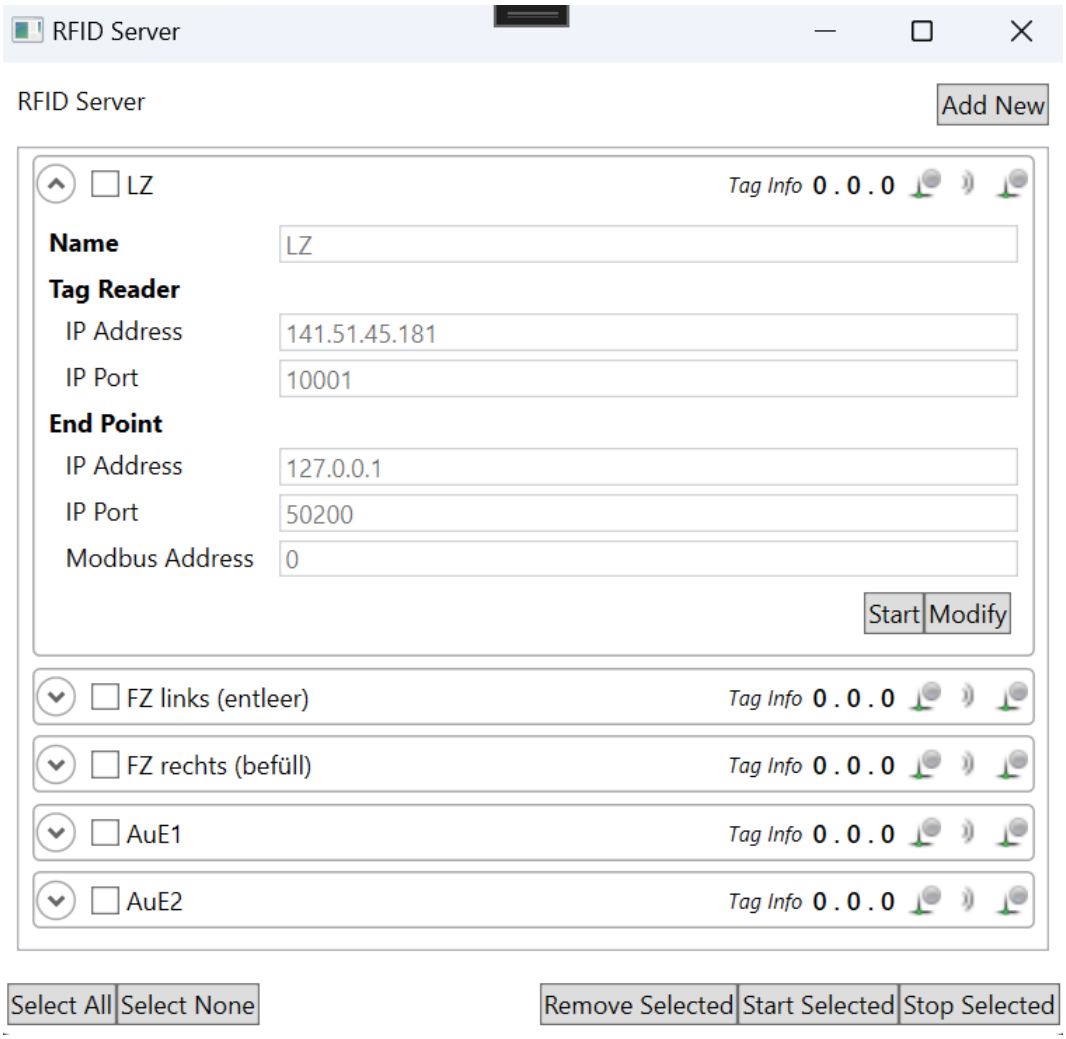


Abbildung 2.7.: Startbildschirm des Programms "RFID-Server"

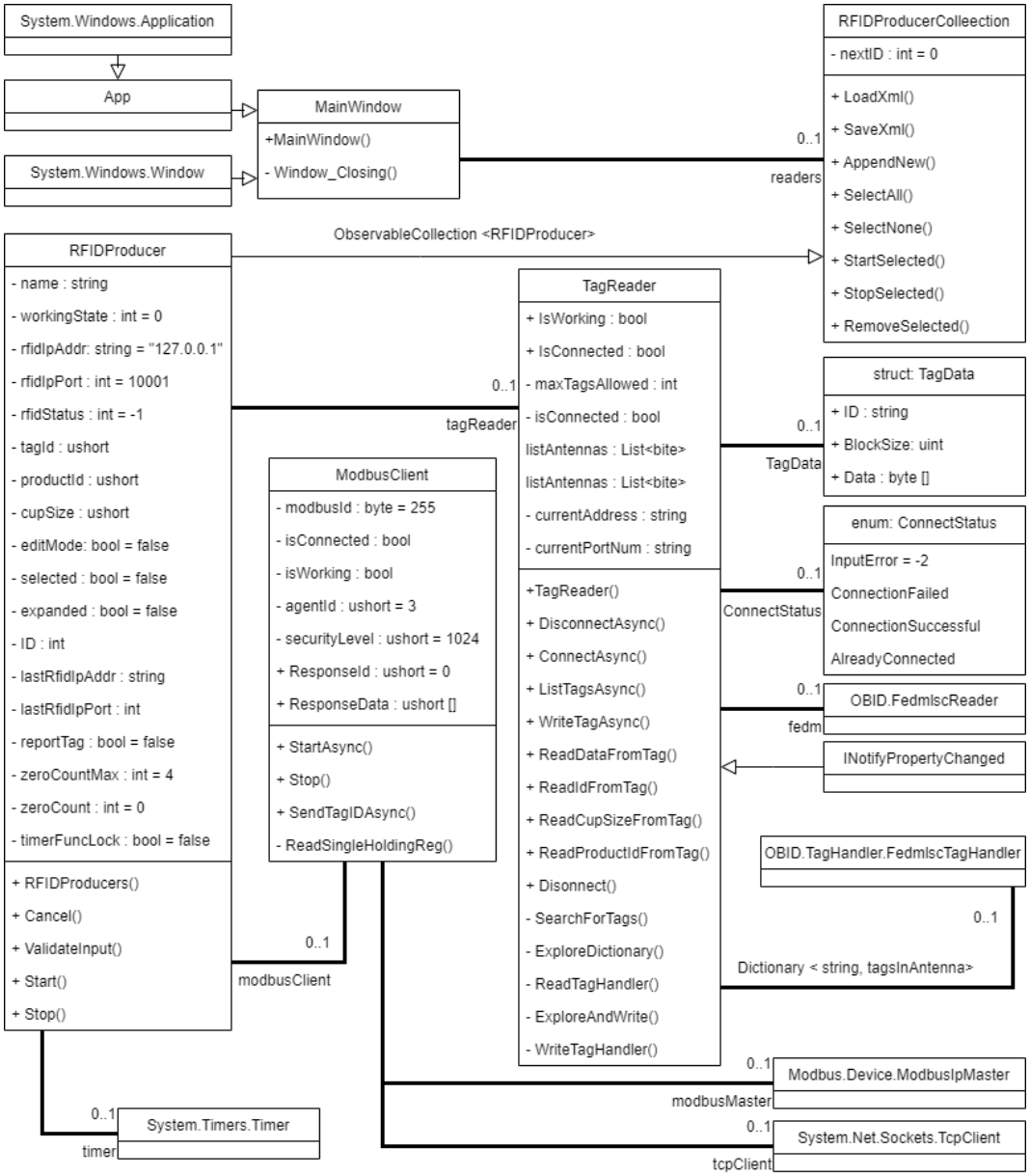


Abbildung 2.8.: Klassendiagramm des Programms "RFID-Server"mit allen vererbenden Klassen jedoch ohne solche die lediglich Typen konvertieren.

3 Konzeptionierung einer integrierten Python Anwendung

3.0.1. GUI - Konzeptionierung

3.0.2. Konzepte zur Datenmodellierung

3.0.3. Konzepte für Controller- und Serviceklassen

3.0.4. Teilautomatisierte Code Dokumentation

4 Analyse zur Fehlerbehandlung

5 Ideensammlung zu kameragestützten Validierungsprozessen in der Lagerverwaltung

5.0.1. Konzepte

5.0.2. Abgeleitete Anforderungen an die Kamera

5.0.3. Kameraauswahl

6 Zusammenfassung und Ausblick

Hier wird die Arbeit zusammengefasst und eun Ausblick auf offene Fragestellungen gebeben.

A Dies ist der erste Anhang

Hier Text einfügen.

Literaturverzeichnis