Autonomes Smartes Gewächshaus

Verfasser: Gruppe 4

9220827, 4379580, 6181218, 8466584, 4585051

Kurs: WWI21SEB

Kernkonzepte und Methoden des Software-Engineering

Studiengangsleitung: Prof. Dr. Sebastian Ritterbusch

Dozent: Michael Binzen

Abgabe: 31.07.2023

Inhalt

1 Einführung und Ziele

Das Hauptziel dieses Projekts ist es, ein intelligentes und autonomes Gewächshaus zu schaffen, das eine optimale Umgebung für das Wachstum von Pflanzen bietet, ohne dass manuell eingegriffen werden muss. Durch die intelligente Pflanzenpflegesoftware FloraGPT werden die Bedürfnisse der Pflanzen regelmäßig ermittelt und es werden die verschiedenen Zonen des Gewächshauses unabhängig voneinander gesteuert, um die Bedürfnisse der Pflanzen individuell zu erfüllen.

Das Hauptziel dieses autonomen Gewächshauses besteht darin, eine ideale Umgebung für das Wachstum von Pflanzen zu schaffen, indem verschiedene Klimaparameter wie Temperatur, Luftfeuchtigkeit und Lichteinfall optimal gesteuert werden. Das Gewächshaus soll so konzipiert sein, dass es ohne manuelle Eingriffe arbeiten kann, wodurch der Pflegeaufwand minimiert wird.

Das Gewächshaus wird in mehrere Zonen unterteilt, wobei jede Zone über eigene Klimasteuerungsgeräte wie Heizungen, Lüfter und Luftbefeuchter verfügt. Diese Zonen werden individuell überwacht und gesteuert, um die spezifischen Anforderungen der Pflanzen in jeder Zone zu erfüllen. Durch diese individuelle Steuerung kann eine maßgeschneiderte Umgebung für unterschiedliche Pflanzenarten geschaffen werden.

Das Gewächshaus bietet die Möglichkeit, eine Vielzahl von Pflanzenarten anzubauen. Jede Zone kann beliebig viele Pflanzen beherbergen, und die Anzahl der Pflanzen in einer Zone kann je nach Bedarf variieren. Jede Pflanze wird mit einem Düngesystem, einer automatischen Bewässerung und einer UV-Lampe ausgestattet, um eine optimale Versorgung und ein gesundes Wachstum zu gewährleisten.

Die Bewässerung der Pflanzen erfolgt mit gesammeltem Regenwasser, das in einem speziellen Wasserspeichersystem aufbewahrt wird. Wenn ausreichend Reserven vorhanden sind, wird das Regenwasser automatisch für die Bewässerung verwendet, wodurch der Wasserverbrauch optimiert wird.

2 Aufgabenstellung

Fachliche Aufgaben

Lfd. Nr.	Use Case	Beschreibung
UC 1	Zonen steuern	Sobald eine Zone hinzugefügt wurde, wird sie autonom verwaltet. Es werden Optimalwerte definiert und die Geräte so gesteuert, dass diese erreicht werden.
UC 2	Pflanzen pflegen	Die Pflege aller Pflanzen in allen Zonen wird im Hintergrund durchgeführt. Dies umfasst die Bewässerung, das Düngen und die Kontrolle der UV-Lampe, um sicherzustellen, dass die Pflanze optimale Wachstumsbedingungen erhält.
UC 3	Wasserversorgung verwalten	Das Gewächshaus verfügt über einen Wassertank zur Sammlung von Regenwasser. Jeder Bewässerungsvorgang prüft, ob genug Regenwasser zum Wässern vorhanden ist. Falls nicht, wird autonom Leitungswasser verwendet.
UC 4	Überwachung	Über eine GUI kann der Benutzer jederzeit den Status aller Zonen, Geräte und Pflanzen einsehen
UC 5	Dokumentation	Das System schreibt eine Log-Datei, in der alle relevanten Handlungen dokumentiert werden. Es wird zwischen Informationen, Warnungen und Fehlern unterschieden
UC 6	Konfiguration	Änderungen am Layout können zur Laufzeit in der Config-Datei bestimmt werden und werden innerhalb einer Minute im System registriert

Technische Aufgaben

Lfd. Nr.	Technische Aufgabe	Beschreibung
TA1	TDD (Test Driven Development)	Die Entwicklung erfolgt nach dem Prinzip des Testgetriebenen Entwicklungsansatzes. Zuerst werden Tests geschrieben, dann wird der Code entwickelt, um diese Tests zu bestehen.
TA2	Einhaltung von SOLID-Prinzipien	Die Anwendung orientiert sich an den SOLID-Prinzipien der objektorientierten Programmierung (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion).

TA3	Verwendung von Design Pattern	Bei der Entwicklung werden bewährte Design Patterns eingesetzt, um die Architektur und Struktur der Anwendung zu verbessern und die Wiederverwendbarkeit zu fördern.
TA4	Objektorientierung	Die Anwendung wird objektorientiert entwickelt, um die verschiedenen Komponenten und Funktionen der Gewächshaussteuerung in eigenständige und wiederverwendbare Objekte zu strukturieren.

3 Qualitätsziele

In diesem Kapitel werden die Qualitätsziele beschrieben, die für das autonome Gewächshaus angestrebt werden. Jedes Ziel repräsentiert einen Aspekt, den das System erfüllen soll, um eine zuverlässige und effiziente Funktionsweise sicherzustellen.

Lfd. Nr.	**Qualitätsziel**	**Beschreibung**
Q 1	Observability	Das Gewächshaus soll eine hohe Observability aufweisen, sodass der Benutzer alle relevanten Systeminformationen und Statusmeldungen leicht einsehen kann. Eine übersichtliche Darstellung der Klimadaten, Wasserversorgung und Gerätestatus ermöglicht eine einfache Überwachung des Systems.
Q 2	Responsiveness	Das Gewächshaus soll eine hohe Responsiveness aufweisen, um schnell auf Änderungen der Umgebungsbedingungen oder Benutzerinteraktionen zu reagieren. Eine schnelle Anpassung der Klimasteuerung und Pflegeprozesse gewährleistet ein optimales Wachstum der Pflanzen.
Q 3	Efficiency	Die Effizienz ist ein entscheidendes Qualitätsziel. Das Gewächshaus soll ressourceneffizient arbeiten und die Bewässerung, Düngung und Beleuchtung optimiert steuern. Dadurch wird eine sparsame Nutzung von Wasser und Energie erreicht, um die Betriebskosten zu minimieren.
Q 4	Reproducability / Correctness	Die Reproduzierbarkeit ist von Bedeutung, um konsistente Ergebnisse zu erzielen. Das Gewächshaus soll sich konsistent verhalten und auf gleiche Pflegebedürfnisse gleich reagieren. So ist sichergestellt, dass einmalig korrektes Verhalten dauerhaft ist.
Q 5	Extensibility	Die Erweiterbarkeit des Systems ist ein wichtiges Ziel. Das Gewächshaus soll leicht um neue Zonen, Geräte oder

		Funktionen erweitert werden können, um auf zukünftige Anforderungen und Technologien reagieren zu können.
Q 6	Availability	Die Verfügbarkeit ist entscheidend, um die kontinuierliche Funktion des Gewächshauses zu gewährleisten. Das System soll robust und ausfallsicher sein, um potenzielle Störungen zu minimieren und eine unterbrechungsfreie Versorgung der Pflanzen zu gewährleisten.

4 Stakeholder

Lfd. Nr.	Stakeholder	Beschreibung	Ziel
SH1	Entwickler	Der Entwickler ist ein Mitglied der Entwicklerfirma und arbeitet aktiv an der Implementierung und Wartung des Gewächshaus-Systems.	Ziel ist die leicht verständliche und effiziente Anpassung und Erweiterung des Systems.
SH2	Kunde	Der Kunde ist der Endanwender des autonomen Gewächshauses und hat eine eigene Zone oder mehrere Zonen mit Pflanzen zu pflegen.	Ziel ist eine übersichtliche Benutzeroberfläche und optimale Pflegebedingungen.

5 Randbedingungen

Lfd. Nr.	Qualitätsziel	Beschreibung
R1	Programmiersprache	Python wird als Programmiersprache verwendet.
R2	Lokale Anwendung	Das autonome Gewächshaus ist eine lokale Anwendung.
R3	Log-Führung	Ein detailliertes Logbuch muss geführt werden.
R4	Anbindung an FloraGPT	Das Gewächshaus muss mit FloraGPT verbunden werden.
R5	Zeitrahmen	Das Projekt hat einen Entwicklungszeitraum von ca. 3 Monaten.
R6	Ressourcenbeschränkung	Das Projekt kann nur von 5 Mitarbeitern bearbeitet werden, wobei jeder Mitarbeiter aufgrund anderer Verpflichtungen nur ca. 15% seiner Arbeitszeit für das Projekt aufbringen kann.

6 Kontextabgrenzung

Fachliche Kontextabgrenzung

Innerhalb des Projekts (im Scope):

Im Rahmen des Gewächshausprojekts umfasst die Kontextabgrenzung folgende Aufgaben:

Entwicklung und Implementierung der Gewächshaussteuerung: Die Hauptaufgabe des Projekts besteht darin, eine intelligente und autonome Gewächshaussteuerung zu entwickeln und zu implementieren. Diese Steuerung ermöglicht es, verschiedene Parameter des Gewächshauses zu überwachen und zu regeln, um optimale Bedingungen für das Pflanzenwachstum zu gewährleisten. Außerdem werden die Grundlagen für eine Smart Home Architektur geschaffen.

Außerhalb des Projekts (out of Scope):

Im Rahmen der Kontextabgrenzung liegen bestimmte Funktionen außerhalb des Projekts, die nicht in den Aufgabenbereich der Gewächshaussteuerung fallen:

1. Funktionen zur weiteren Verarbeitung des erstellten Logs
2. Detaillierte Entwicklung einer weiteren Smart Home Steuerung

Technischer Kontext

Innerhalb des Projekts (im Scope):

1. Die Gewächshaussteuerung wird als lokale Anwendung entwickelt, die auf einem Computer oder einem speziellen Steuerungsgerät im Gewächshaus läuft. Die Anwendung wird in der Programmiersprache Python entwickelt.
2. Im Rahmen des Projekts werden spezielle Geräte entwickelt, die für die Steuerung und Regelung des Gewächshauses relevant sind. Dazu gehören Sensoren zur Messung von Temperatur, Luftfeuchtigkeit, Bodenfeuchte usw. sowie Aktoren wie Heizungen, Lüfter, Luftbefeuchter, Düngesysteme, Bewässerungssysteme und UV-Lampen.
3. Die Gewächshaussteuerung wird mit FloraGPT, einer KI-basierten Plattform zur Pflanzenpflege und -analyse, verbunden.

Außerhalb des Projekts (out of Scope):

1. Die Entwicklung und Bereitstellung der physischen Hardware
2. Die Implementierung von Netzwerkfunktionen, Sicherheitsmaßnahmen und Zugriffskontrollen

7 Lösungsstrategie

Um eine effiziente und gut strukturierte Gewächshaussteuerung zu entwickeln, wurden folgende Kernentscheidungen getroffen:

1. Verwendung von MVC (Model-View-Controller) Designmuster: Durch die klare Trennung von GUI (View), Logik (Controller) und Daten (Model) wird der Code übersichtlicher und leichter wartbar. Das MVC-Muster ermöglicht eine flexible Entwicklung, da Änderungen an einem Teil der Anwendung nur minimale Auswirkungen auf andere Teile haben. Die Benutzeroberfläche kann unabhängig von der darunterliegenden Logik entwickelt und getestet werden, was die Entwicklungszeit verkürzt und die Wiederverwendbarkeit erhöht.

2. Einhaltung von SOLID-Prinzipien: Wir werden die SOLID-Prinzipien, insbesondere Single Responsibility Principle (SRP), Open/Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP) und Dependency Inversion Principle (DIP), beachten. Dies führt zu einer flexiblen und erweiterbaren Architektur, die gut gewartet und getestet werden kann.

3. Verwendung folgender Entwurfsmuster: Die Entwurfsmuster Singleton und Factory finden Anwendung, um die Erstellung und Verwaltung der Objekte in unserer Anwendung zu vereinfachen. Das Singleton-Muster gewährleistet, dass bestimmte Klassen nur eine einzige Instanz haben, während das Factory-Muster die Erstellung von Objekten abstrahiert.

4. Nutzung des Python Logging Moduls: Wir verwenden das Python Logging Modul, um umfangreiche und detaillierte Protokolldateien zu erstellen. Dies ermöglicht eine effiziente Fehlerbehebung und Analyse von Ereignissen während der Laufzeit der Anwendung.

5. Konfiguration mittels JSON: Die Konfiguration der Gewächshaussteuerung erfolgt über JSON-Dateien, die einfach zu lesen und zu bearbeiten sind. Dies ermöglicht eine einfache Anpassung der Einstellungen und Parameter, ohne den Quellcode ändern zu müssen.

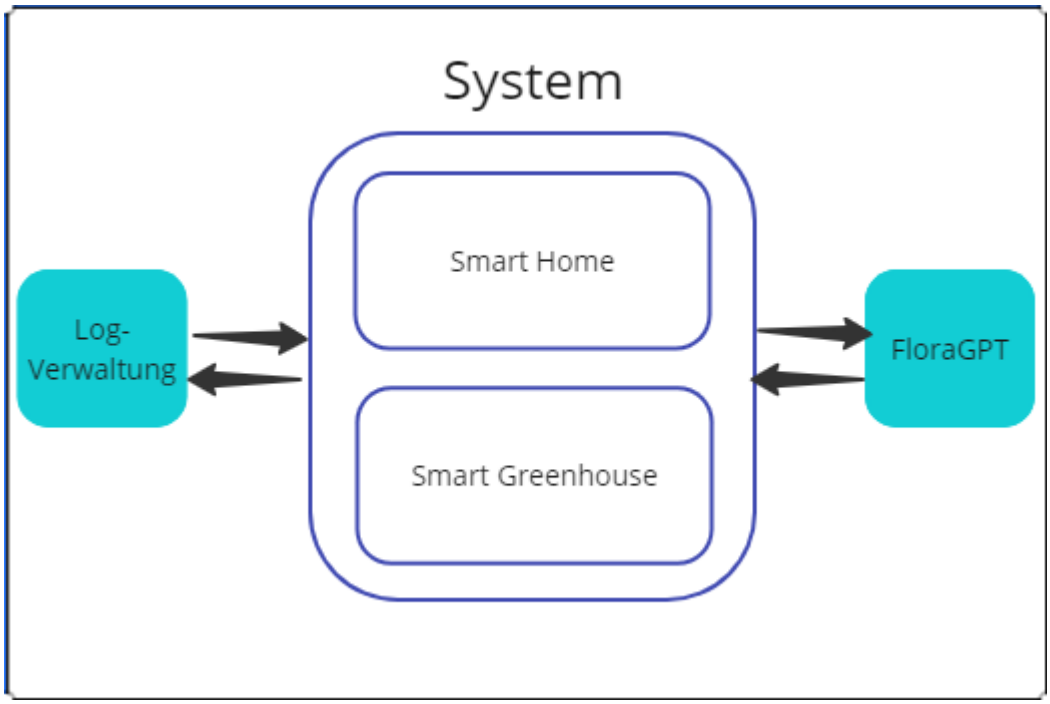
6. Parallelität durch Threading der Controller: Um die Reaktionsfähigkeit der Gewächshaussteuerung zu verbessern, werden bestimmte Aufgaben in separaten Threads ausgeführt. Dadurch können Prozesse parallel ablaufen, was insbesondere bei der gleichzeitigen Steuerung mehrerer Gewächshauszonen von Vorteil ist.

7. Exception Handling: Wir implementieren ein robustes Exception Handling, um Fehler und Ausnahmen gezielt zu behandeln. Dadurch können wir unvorhergesehene Situationen abfangen und angemessen darauf reagieren, um einen reibungslosen Betrieb der Anwendung sicherzustellen.

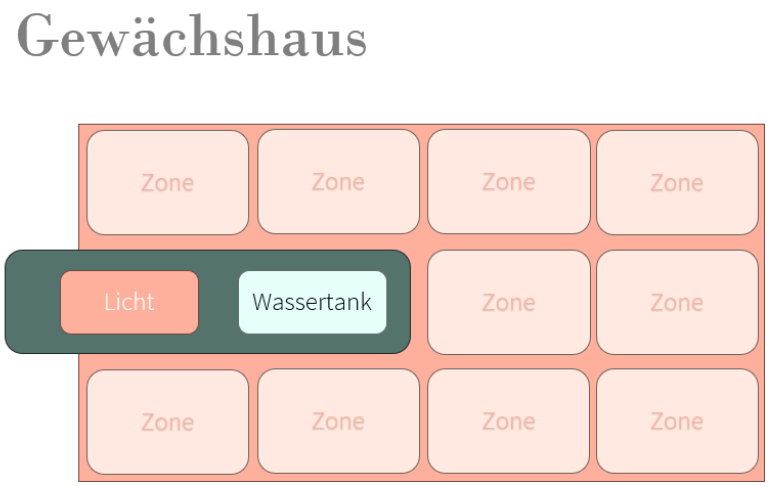
8. Testing mittels Unit Testing Framework: Wir setzen ein Unit Testing Framework ein, um unsere Funktionen und Komponenten isoliert zu testen. Dies ermöglicht eine gründliche Überprüfung der einzelnen Teile der Anwendung und stellt sicher, dass sie wie erwartet funktionieren.

Durch die Umsetzung dieser Lösungsstrategie streben wir an, eine zuverlässige, flexible und gut strukturierte Gewächshaussteuerung zu entwickeln, die den Anforderungen unserer Stakeholder gerecht wird und ein autonomes und optimal versorgtes Gewächshaus ermöglicht.

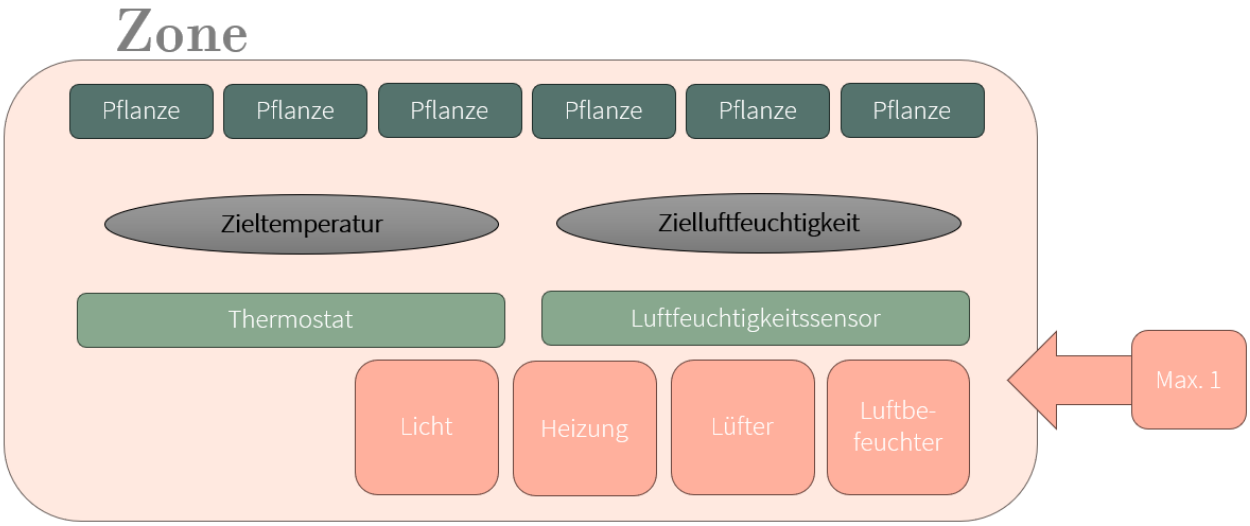
8 Bausteinsicht



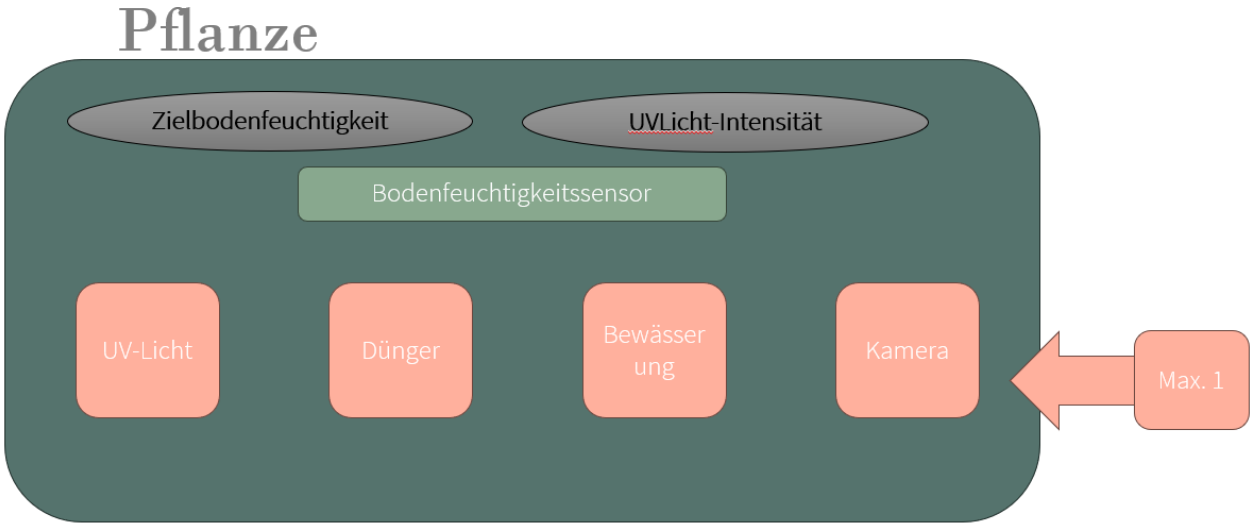
Unser System umfasst in der ersten Iteration im Wesentlichen die Gewächshausfunktionen sowie eine rudimentäre Implementierung eines Smart Home. Als Schnittstellen sind dabei die Anbindung an Flora-GPT sowie an ein Logging-Tool vorhanden.



Das Gewächshaus selbst verfügt über Zonen, die vollständig unabhängig voneinander laufen. Außerdem können im Gewächshaus Lampen angebracht werden und es verfügt über einen Tank zur Speicherung von Regenwasser.

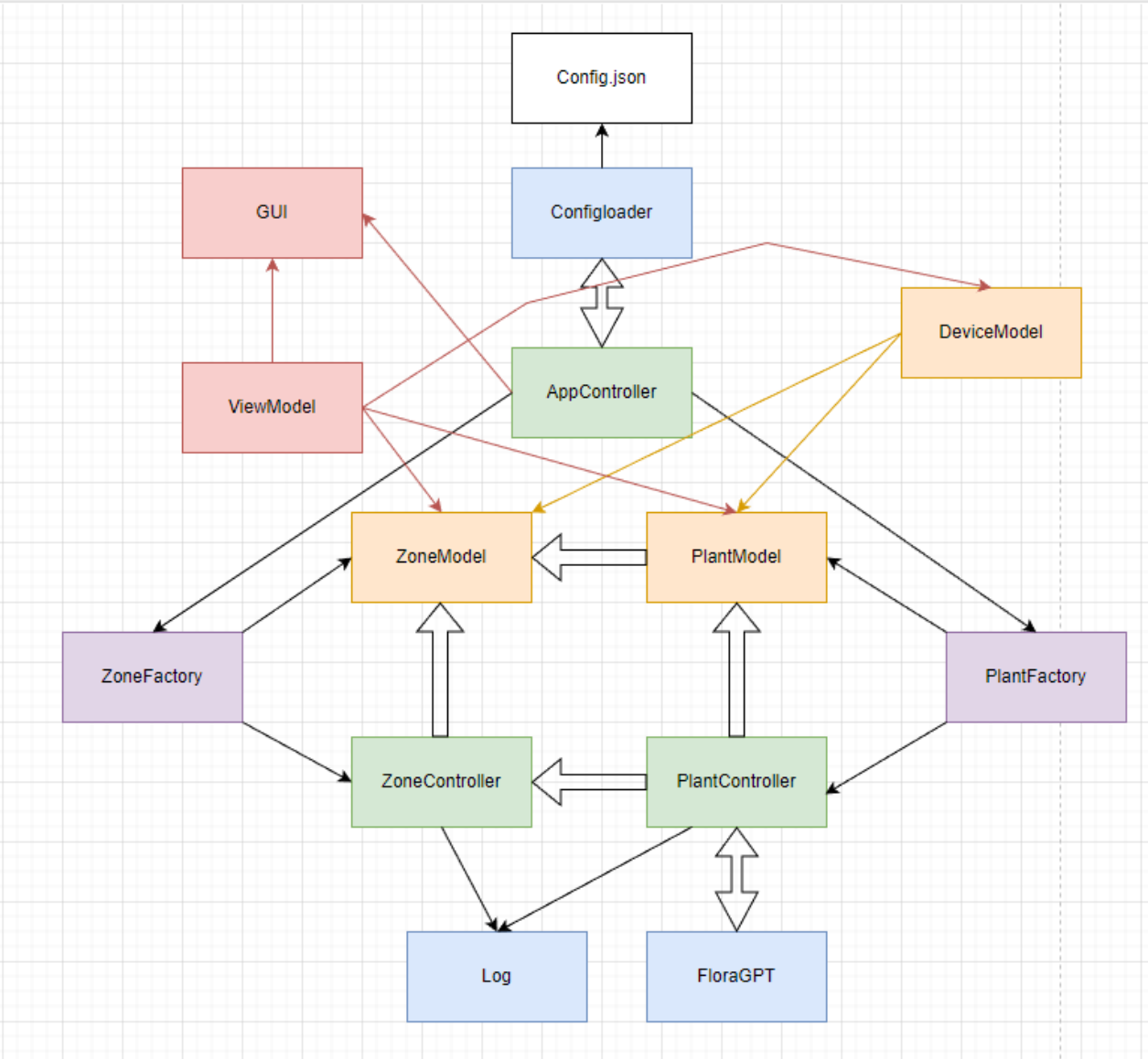


Die Zonen können über Beleuchtung, Heizung, Lüfter und Luftbefeuchter verfügen. Außerdem halten sie die Pflanzenobjekte und haben eine einstellbare Zieltemperatur und Zielluftfeuchtigkeit. Sensoren für Temperatur und Luftfeuchtigkeit sind obligatorisch. Es ist zu beachten, dass pro Zone nur ein Gerät eines Typs angebracht werden kann.



Pflanzen verfügen wie Zonen über Zielwerte, in diesem Fall Bodenfeuchtigkeit und UV-Licht-Intensität. Auch der Bodenfeuchtigkeitssensor ist obligatorisch. Zudem können Pflanzen über maximal je eines der folgenden Geräte verfügen: UV-Licht, Dünger, Bewässerung und Kamera.

Whitebox Gesamtsystem



Lfd. Nr.	Stakeholder	Beschreibung	Ziel
1	Configloader	Der Configloader ist für das Lesen der Konfiguration aus der Datei config.Json verantwortlich. Er lädt die Konfigurationsdaten, die für die Steuerung des Gewächshauses benötigt werden, und stellt sie anderen Komponenten zur Verfügung. Dabei kann er die gelesenen Konfigurationsdaten sowohl lesen als auch schreiben.	Lesen/Schreiben von Konfigurationsdaten

2	App-Controller	Der App-Controller übernimmt die zentrale Steuerung der Anwendung. Er startet die GUI, verknüpft das ViewModel mit der GUI und greift auf die ZoneFactory und PlantFactory zu, um die benötigten Zonen und Pflanzen zu erstellen. Der App-Controller koordiniert die Interaktion zwischen den verschiedenen Komponenten und stellt sicher, dass die Anwendung reibungslos funktioniert.	Zugriff auf ZoneFactory und PlantFactory
3	GUI	Die GUI (Graphical User Interface) ist für die Anzeige des ViewModels verantwortlich. Sie bietet dem Benutzer eine Oberfläche, in welcher die aktuellen Daten angezeigt werden.	Anzeige und Benutzerinteraktion
4	ViewModel	Das ViewModel ist für die Bereitstellung der Daten für die GUI verantwortlich. Es liest Daten aus den DeviceModels, PlantModels und ZoneModel und stellt sie in einem geeigneten Format für die Anzeige in der GUI bereit. Das ViewModel kümmert sich um die Darstellung der Daten und ermöglicht es der GUI, die aktuellen Informationen anzuzeigen.	Datenbereitstellung für die GUI
5	ZoneFactory	Die ZoneFactory ist für die Erstellung der ZoneModel- und ZoneController-Komponenten verantwortlich. Sie erzeugt neue Zonenkomponenten und initialisiert sie mit den erforderlichen Daten und Geräten für die zonale Steuerung des Klimas im Gewächshaus.	Erzeugen von Zonenkomponenten
6	ZoneModel	Das ZoneModel repräsentiert eine Gewächshauszone und enthält ein Array von DeviceModels und PlantModels, die in dieser Zone vorhanden sind. Es bietet eine Schnittstelle zur Datenbereitstellung für den ZoneController, damit dieser die Zone entsprechend steuern kann.	Datenbereitstellung für ZoneController
7	ZoneController	Der ZoneController ist für die Verwaltung des ZoneModels und die Steuerung der Gewächshauszone verantwortlich. Er steuert die Geräte in der Zone, wie Heizung, Lüfter und Luftbefeuchter, basierend auf den Daten im ZoneModel. Zudem schreibt er relevante Informationen in das Log, um die Aktivitäten der Zone zu protokollieren.	Steuerung und Logging für die Zone

8	PlantFactory	Die PlantFactory ist für die Erstellung der PlantModel- und PlantController-Komponenten verantwortlich. Sie erzeugt neue Pflanzenkomponenten und initialisiert sie mit den erforderlichen Daten und Geräten.	Erzeugen von Pflanzenkomponenten
9	PlantModel	Das PlantModel repräsentiert eine Pflanze im Gewächshaus und enthält ein Array von DeviceModels, die für diese Pflanze relevant sind, wie das Düngesystem, die Bewässerung und die UV-Lampe. Es bietet eine Schnittstelle zur Datenbereitstellung für den PlantController, damit dieser die Pflanze entsprechend steuern kann.	Datenbereitstellung für PlantController
10	PlantController	Der PlantController ist für die Verwaltung des PlantModels und die Steuerung der Pflanzen im Gewächshaus verantwortlich. Er steuert die Geräte und Prozesse, die für die optimale Versorgung der Pflanze notwendig sind, wie das Düngesystem, die Bewässerung und die UV-Lampe. Zudem schreibt er relevante Informationen in das Log und ruft FloraGPT auf, um die Pflegehinweise für die Pflanze zu erhalten.	Steuerung, Logging und FloraGPT-Zugriff für die Pflanze

9 Laufzeitsicht

Der generelle Ablauf der Controller lässt sich als zyklisch beschreiben. Sie laufen in einem Thread in Dauerschleife. Im Folgenden wird der Ablauf einzelner Objekte näher erläutert.

Ablauf der Pflanzensteuerung PlantController

1. Alle 10 Sekunden überprüft der PlantController die aktuelle Bodenfeuchtigkeit mithilfe des SoilHumiditySensors.
2. Ist die Bodenfeuchtigkeit geringer als die Zielfeuchtigkeit, startet der Controller die Bewässerung, indem er das Device "Irrigation" aktiviert und mit dem Wassertank kommuniziert, um die Pflanze zu bewässern.
3. Der PlantController nimmt ein Bild der Pflanze mithilfe der Kamera auf.
4. Das aufgenommene Bild wird an FloraGPT gesendet, um Pflegehinweise zu erhalten.
5. FloraGPT gibt als Antwort den Hinweis "Zu geringe Luftfeuchtigkeit".

6. Der PlantController manipuliert das ZoneModel und erhöht die Zielluftfeuchtigkeit, um die Pflegeempfehlung umzusetzen.
7. Zur Simulation eines realen Umfelds wird die Bodenfeuchtigkeit über die Zeit gesenkt, und eine zufällige Veränderung der Bodenfeuchtigkeit wird simuliert.
8. 10 Sekunden Pause, dann weiter bei 1.

10 Verteilungssicht

Da es sich um ein lokales Programm ohne weitere Anbindung handelt, gibt es an dieser Stelle nichts zu erläutern.

11 Querschnittliche Konzepte

Entwurfsmuster

Im Projekt wurden die Entwurfsmuster Singleton und Factory verwendet. Ursprünglich wurde außerdem geplant, das Observer-Pattern zu verwenden, diese Entscheidung wurde jedoch verworfen. Näheres dazu befindet sich im Abschnitt 12 Architekturentscheidungen.

Factory-Pattern

Das Factory Pattern wurde erfolgreich eingesetzt, um die Erstellung von Zonen und Pflanzen zu vereinfachen und zu standardisieren. Durch die Verwendung dieses Entwurfsmusters können Zonen und Pflanzen vorkonfiguriert und einheitlich erstellt werden, was zu einer effizienten und konsistenten Verwaltung führt. Die Factory erstellt Instanzen dieser Objekte und sorgt dafür, dass sie bereits mit den erforderlichen Eigenschaften und Parametern ausgestattet sind, wodurch die Komplexität der Erstellung reduziert wird und eine klare Trennung von Erstellung und Nutzung erreicht wird.

Singleton-Pattern

Im Programm wurde das Singleton-Entwurfsmuster angewendet, um sicherzustellen, dass nur ein einziges Gewächshaus existiert. Eine abgewandelte Version des Singleton-Musters wurde auch für Zonen und Pflanzen verwendet, wodurch sichergestellt wird, dass sie jeweils nur über ein Gerät pro Typ verfügen können. Dadurch wird eine konsistente und eindeutige Verwaltung der Gewächshausressourcen gewährleistet.

SOLID-Prinzipien

In der gesamten Projektarchitektur wurde die Einhaltung von SOLID-Prinzipien forciert. Dies wurde in der Veranstaltung mehrfach thematisiert und in Präsentationen festgehalten, sodass es hier aufgrund der Platzbegrenzung nicht näher thematisiert wird.

Testing

Es wurde kein TDD im herkömmlichen Sinne eingesetzt, die Tests der Komponenten wurden erst nach ihrer Entwicklung geschrieben. Jedoch wurden vor der Integration alle Komponenten mit Tests versehen, um sicherzustellen, dass sie korrekt funktionieren und die Fehler höchstens noch bei der Verbindung zu suchen sind.

Im Projekt wurden umfassende Unit Tests implementiert, da sie eine Vielzahl von Vorteilen bieten. Durch die Verwendung einer Mock-Objekt-Bibliothek können Abhängigkeiten zu anderen Komponenten isoliert und simuliert werden, was eine präzise Prüfung der Funktionalität ermöglicht. Die strikte Modularisierung des gesamten Projekts hat es uns erleichtert, einzelne Komponenten unabhängig voneinander zu testen und sicherzustellen, dass sie gemäß den Spezifikationen arbeiten. Darüber hinaus ermöglichen Unit Tests auch die Durchführung von Integrationstests, bei denen die Interaktion zwischen verschiedenen Komponenten überprüft wird. Dadurch kann die Gesamtfunktionalität der Anwendung getestet und sichergestellt werden, dass die Komponenten reibungslos zusammenarbeiten.

12 Architekturentscheidungen

Observer Pattern

Wir haben uns gegen die Verwendung des Observer-Pattern entschieden, da in unserem Szenario keine realen Sensoren existieren, die die Umwelt überwachen. Das Observer-Pattern hätte dazu geführt, dass wir Umweltveränderungen in weiteren Threads hätten simulieren müssen, um dann die Observer zu benachrichtigen. Dies hätte eine unnötige Komplexität und Rechenlast bei der Ausführung des Programms verursacht.

Usersteuerung

Da unser Ziel war, ein durch KI verwaltetes Gewächshaus zu erstellen, haben wir in der ersten Iteration auf die Möglichkeit von Userinteraktion verzichtet. Sämtliche Steuerung erfolgt autonom anhand der Hinweise von FloraGPT. Der Benutzer kann lediglich die Konfigurationsdatei pflegen.

Konkretisierung der Geräte

In erster Instanz hatten wir geplant, Geräte abstrakt zu behandeln und erst bei Ausführung näher zu bestimmen. Da das Gewächshaus jedoch sehr konkrete Anforderungen an die Komposition der Geräte hat, haben wir darauf verzichtet und Geräten konkrete Plätze (Attribute) an Zonen und Pflanzen zugewiesen.

13 Qualitätsanforderungen

Die Qualitätsziele wurden bereits genau beschrieben, sodass an dieser Stelle lediglich auf die Qualitätsszenarien im Zusammenhang mit den Zielen eingegangen wird. Efficiency und

Correctness sind zudem als übergeordnete Qualitätsziele zu sehen, die in allen Szenarien zu gewährleisten sind.

Qualitätsszenarien

Lfd. Nr.	Bezeichnung	Beschreibung	Qualitätsziele
QS1	Venusfliegenfalle	Der Kunde möchte in seinem Gewächshaus Venusfliegenfallen züchten. Für diese müssen Duftsensoren installiert werden, die registrieren, wann die Pflanzen Lockdüfte produzieren. Werden Duftstoffe registriert, müssen durch ein Gerät Fliegen freigelassen werden.	Durch die Extensibility(Q5) des Systems ist diese Kundenanforderung leicht und ohne Modifikationen umsetzbar. Es müssen nur eine neue Art von Sensor und ein neuer Gerätetyp implementiert werden.
QS2	Urlaub	Der Kunde möchte sein Gewächshaus im Urlaub überwachen.	Die Observability ermöglicht das Überwachen aller Funktionen. Durch die Extensibility muss nur eine GUI hinzugefügt werden, die über das Internet auf das Programm zugreifen kann. Availability ermöglicht, dass er zu jeder Zeit darauf zugreifen kann.
QS3	Mehr Pflanzen	Der Kunde hat neue Pflanzen gekauft und mit den entsprechenden Geräten und Sensoren im Gewächshaus platziert. Er möchte diese möglichst schnell und einfach anbinden.	Durch die Extensibility muss lediglich pro Pflanze ein Eintrag in der Config-Datei hinzugefügt werden. Die Responsiveness ermöglicht ein Laden der zusätzlichen Pflanzen zur Laufzeit.
QS4	Verbesserte KI	FloraGPT hat einen Nachfolger erhalten. Die Schnittstelle soll ersetzt werden.	Durch Einhaltung der SOLID-Prinzipien muss lediglich der Aufruf im PlantController angepasst werden.

14 Risiken und technische Schulden

Die folgende Tabelle enthält eine Liste von technischen Schulden und Risiken im aktuellen Zustand des Projekts. Für jedes Problem wird ein Lösungsansatz vorgeschlagen, um die Herausforderungen zu bewältigen und das System zu verbessern. Die Umsetzung dieser Lösungsansätze trägt dazu bei, die Gesamtqualität des Projekts zu steigern und die Anwendung robuster und erweiterbarer zu machen.

Lfd. Nr.	Bezeichnung	Beschreibung	Lösungsansatz
TS1	Userkontrolle	Derzeit kein Usereingriff möglich, nur Programmbeendigung	Abschaltung einzelner Zonen ermöglichen; Manuelle Einstellung von Zielwerten ermöglichen
TS2	Neustart des Systems	Neustart setzt alle Werte auf Default-Einstellung	Konfigurationsdatei zur Laufzeit schreiben; Letzten Stand in der Konfiguration speichern
TS3	Gerätevollständigkeit	Vollständige Belegung mit allen Geräten wird vorausgesetzt	Vorhandene Geräte in der Konfiguration definieren; Flexibilität für fehlende Geräte gewährleisten
TS4	Remotezugriff	Das System läuft nur lokal	Hosting der Logik auf einem Server; Bereitstellung einer API-Schnittstelle; GUI greift auf die Schnittstelle zu
TS5	Hinzufügen zur Laufzeit	Derzeit ist kein Hinzufügen zu der Konfiguration zur Laufzeit möglich. Durch einen unidentifizierten Bug lässt sich die Konfigurationsdatei zur Laufzeit nicht erweitern. Löschen ist jedoch möglich.	Workaround: Neustart Der Bug muss identifiziert und behoben werden, das ist jedoch nicht bis zum Abgabetermin möglich gewesen

Glossar

Auf ein Glossar verzichten wir an dieser Stelle, da alle verwendeten Begriffe in der Veranstaltung umfänglich aufgegriffen und besprochen wurden.

Aufgabenverteilung

Zuletzt gehen wir auf die Aufgabenverteilung ein. Es wird unterschieden in Konzeptionierung, Programmierung und Dokumentation der einzelnen Aspekte.

Konzeptionierung

1. Erstellung der Package-Struktur sowie Entscheidung für Architekturaspekte: Alle

2. Modellierung der Model-Klassen: 9220827, 4379580, 6181218
3. Planung der Controller und Schnittstellen: 8466584, 4585051
4. Planung der GUI: 4585051, 9220827, 4379580

Programmierung

1. Entwicklung der Model-Klassen: 9220827, 4379580, 6181218
2. Entwicklung Steuerung: 8466584, 4585051, 6181218
3. Entwicklung der GUI: 4585051, 9220827, 4379580
4. Schnittstellen Logger, FloraGPT und Config: 8466584

Dokumentation

1. 9220827
2. 9220827
3. 4585051
4. 4585051
5. 4379580
6. 4379580
7. 8466584
8. 6181218
9. 8466584
10. –
11. 9220827, 4379580, 6181218
12. 8466584
13. 4585051
14. Alle (in Besprechung erstellt)