Hier ist meine Doku von meinen Programmteil

Der theorieteil fehlt hier noch, also alles was wir geplant hatten rollen konzept und

**1. Namespace- und Klassen-Deklaration**

Der Code beginnt mit der Definition von Namespaces und der Hauptklasse. Innerhalb dieser Klasse werden mehrere statische Felder deklariert:

* users: Eine Liste, die alle Benutzer speichert.
* currentUser: Ein Benutzerobjekt, das den aktuell angemeldeten Benutzer repräsentiert.
* rootUser: Ein Benutzerobjekt, das den Root-Benutzer repräsentiert.
* lists: Eine Liste, die verschiedene Listenelemente enthält.

**2. BeforeRun Methode**

Die BeforeRun-Methode wird aufgerufen, bevor das System in den Hauptlaufmodus wechselt. In dieser Methode wird eine einfache Nachricht auf der Konsole ausgegeben, um anzuzeigen, dass das Cosmos-System erfolgreich gestartet wurde. Diese Methode dient als Initialisierungsroutine, um sicherzustellen, dass das System bereit ist, bevor es zur Hauptverarbeitung übergeht.

**3. LoginRootUser Methode**

Die LoginRootUser-Methode verwaltet den Anmeldeprozess für den Root-Benutzer. Zunächst wird der Benutzer aufgefordert, das Passwort für den Root-Benutzer einzugeben. Das eingegebene Passwort wird dann mit dem gespeicherten Passwort des Root-Benutzers verglichen. Wenn das Passwort korrekt ist, wird currentUser auf rootUser gesetzt und eine Erfolgsmeldung wird ausgegeben. Wenn das Passwort falsch ist, wird die Methode rekursiv erneut aufgerufen, um den Benutzer zur erneuten Eingabe des Passworts aufzufordern. Diese Methode stellt sicher, dass nur der autorisierte Benutzer Zugriff auf Root-Privilegien erhält.

**4. Run Methode**

Die Methode Run ist die Hauptschleife des Systems, die kontinuierlich läuft, bis ein Befehl gegeben wird, das System zu beenden. Diese Methode fordert den Benutzer auf, einen Befehl einzugeben und führt je nach Eingabe unterschiedliche Aktionen aus. Die möglichen Befehle umfassen das Erstellen eines Benutzers, das Löschen eines Benutzers, das Anzeigen der Liste der Benutzer, das Ändern einer Rolle, das Ändern eines Passworts, das Wechseln des aktuellen Benutzers, das Neustarten des Systems und das Verwalten von Listen. Jeder dieser Befehle wird durch eine entsprechende Methode umgesetzt, die spezifische Aufgaben ausführt.

**5. CreateUser Methode**

Die Methode CreateUser ermöglicht es einem autorisierten Benutzer, einen neuen Benutzer zu erstellen. Der Benutzer wird aufgefordert, einen Benutzernamen, ein Passwort und eine Rolle für den neuen Benutzer einzugeben. Die Methode überprüft, ob der eingegebene Benutzername bereits existiert, um Duplikate zu vermeiden. Anschließend wird der neue Benutzer zur users-Liste hinzugefügt, sofern der aktuelle Benutzer über die notwendigen Berechtigungen verfügt. Diese Methode stellt sicher, dass nur autorisierte Benutzer neue Benutzerkonten erstellen können.

**6. DeleteUser Methode**

Die Methode DeleteUser ermöglicht es einem autorisierten Benutzer, einen bestehenden Benutzer zu löschen. Der Benutzer wird aufgefordert, den Benutzernamen des zu löschenden Benutzers einzugeben. Die Methode sucht in der users-Liste nach dem angegebenen Benutzernamen und entfernt den entsprechenden Benutzer, sofern der aktuelle Benutzer über die notwendigen Berechtigungen verfügt und der zu löschende Benutzer nicht der Root-Benutzer ist. Diese Methode stellt sicher, dass nur autorisierte Benutzer Konten löschen können und der Root-Benutzer nicht entfernt werden kann.

**7. ListUsers Methode**

Die Methode ListUsers zeigt alle Benutzer im System an. Sie durchläuft die users-Liste und gibt für jeden Benutzer den Benutzernamen und die zugehörige Rolle aus. Diese Methode dient dazu, dem aktuellen Benutzer einen Überblick über alle existierenden Benutzer und deren Rollen zu verschaffen.

**8. ShowCommands Methode**

Die Methode ShowCommands listet alle verfügbaren Befehle auf, die der Benutzer eingeben kann. Diese Befehle umfassen das Erstellen, Löschen und Verwalten von Benutzern, das Verwalten von Listen, sowie Systembefehle wie Neustart und Beenden. Die Methode zeigt die Befehle auf der Konsole an, um dem Benutzer eine Referenz zu bieten, welche Aktionen er ausführen kann.

**9. ChangeRole Methode**

Die Methode ChangeRole ermöglicht es einem autorisierten Benutzer, die Rolle eines anderen Benutzers zu ändern. Der Benutzer wird aufgefordert, den Benutzernamen des zu ändernden Benutzers und die neue Rolle einzugeben. Die Methode überprüft, ob der aktuelle Benutzer die notwendigen Berechtigungen hat, um die Rolle zu ändern, und aktualisiert dann die Rolle des angegebenen Benutzers. Diese Methode stellt sicher, dass nur autorisierte Benutzer die Rollen von anderen Benutzern ändern können und dass die Rollenänderung den festgelegten Berechtigungen entspricht.

**10. ChangePassword Methode**

Die Methode ChangePassword ermöglicht es einem Benutzer, sein Passwort zu ändern. Der Benutzer wird aufgefordert, den Benutzernamen, das alte Passwort und das neue Passwort einzugeben. Die Methode überprüft, ob das alte Passwort korrekt ist und aktualisiert dann das Passwort des Benutzers, falls die Überprüfung erfolgreich ist. Diese Methode stellt sicher, dass Benutzer ihre Passwörter ändern können, um ihre Konten zu sichern.

**11. ChangeUser Methode**

Die Methode ChangeUser ermöglicht es einem Benutzer, sich als ein anderer Benutzer anzumelden. Der Benutzer wird aufgefordert, den Benutzernamen und das Passwort des gewünschten Benutzerkontos einzugeben. Die Methode überprüft, ob das Passwort korrekt ist und setzt currentUser auf den gewünschten Benutzer, falls die Überprüfung erfolgreich ist. Diese Methode erlaubt es Benutzern, zwischen verschiedenen Konten zu wechseln, sofern sie die richtigen Anmeldeinformationen haben.

**12. RebootSystem Methode**

Die Methode RebootSystem führt einen Neustart des Systems durch. Dabei werden die users-Liste und die lists-Liste geleert. Eine Nachricht wird ausgegeben, um anzuzeigen, dass das System neu gestartet wird. Diese Methode stellt sicher, dass das System in einen sauberen Zustand zurückgesetzt werden kann.

**13. CreateList Methode**

Die Methode CreateList ermöglicht es einem Benutzer mit entsprechenden Berechtigungen, eine neue Liste zu erstellen. Der Benutzer wird aufgefordert, einen Listennamen und die Rollen für den Zugriff, die Bearbeitung, das Verschieben und das Löschen der Liste anzugeben. Die Methode überprüft, ob der aktuelle Benutzer die notwendigen Berechtigungen hat, um eine Liste zu erstellen. Anschließend wird die neue Liste zur lists-Liste hinzugefügt. Diese Methode stellt sicher, dass nur autorisierte Benutzer Listen erstellen können und dass die Zugriffsrechte korrekt festgelegt sind.

**14. DeleteList Methode**

Die Methode DeleteList ermöglicht es einem Benutzer mit entsprechenden Berechtigungen, eine vorhandene Liste zu löschen. Der Benutzer wird aufgefordert, den Namen der zu löschenden Liste einzugeben. Die Methode sucht in der lists-Liste nach dem angegebenen Listennamen und entfernt die entsprechende Liste, sofern der aktuelle Benutzer über die notwendigen Berechtigungen verfügt. Diese Methode stellt sicher, dass nur autorisierte Benutzer Listen löschen können und dass die Löschung den festgelegten Berechtigungen entspricht.

**15. AccessList Methode**

Die Methode AccessList ermöglicht es einem Benutzer, auf eine vorhandene Liste zuzugreifen. Der Benutzer wird aufgefordert, den Namen der gewünschten Liste einzugeben. Die Methode überprüft, ob der aktuelle Benutzer die notwendigen Berechtigungen hat, um auf die Liste zuzugreifen. Wenn der Benutzer berechtigt ist, wird eine entsprechende Nachricht ausgegeben. Andernfalls wird der Zugriff verweigert. Diese Methode stellt sicher, dass nur autorisierte Benutzer auf Listen zugreifen können.

**16. MoveList Methode**

Die Methode MoveList ermöglicht es einem Benutzer mit entsprechenden Berechtigungen, eine Liste zu verschieben. Der Benutzer wird aufgefordert, den Namen der zu verschiebenden Liste einzugeben. Die Methode überprüft, ob der aktuelle Benutzer die notwendigen Berechtigungen hat, um die Liste zu verschieben. Wenn der Benutzer berechtigt ist, wird eine entsprechende Nachricht ausgegeben. Andernfalls wird das Verschieben verweigert. Diese Methode stellt sicher, dass nur autorisierte Benutzer Listen verschieben können.

**17. EditList Methode**

Die Methode EditList ermöglicht es einem Benutzer mit entsprechenden Berechtigungen, eine Liste zu bearbeiten. Der Benutzer wird aufgefordert, den Namen der zu bearbeitenden Liste einzugeben. Die Methode überprüft, ob der aktuelle Benutzer die notwendigen Berechtigungen hat, um die Liste zu bearbeiten. Wenn der Benutzer berechtigt ist, wird eine entsprechende Nachricht ausgegeben. Andernfalls wird die Bearbeitung verweigert. Diese Methode stellt sicher, dass nur autorisierte Benutzer Listen bearbeiten können.

**18. ShowLists Methode**

Die Methode ShowLists zeigt alle verfügbaren Listen im System an. Sie durchläuft die lists-Liste und gibt für jede Liste den Namen sowie die Rollen für den Zugriff, die Bearbeitung, das Verschieben und das Löschen aus. Diese Methode dient dazu, dem aktuellen Benutzer einen Überblick über alle existierenden Listen und deren Berechtigungen zu verschaffen.

**19. IsValidRole Methode**

Die Methode IsValidRole überprüft, ob eine gegebene Rolle gültig ist. Sie akzeptiert einen String als Eingabe und gibt einen booleschen Wert zurück, der angibt, ob die Rolle "admin", "fachanwender" oder "anwender" ist. Diese Methode wird verwendet, um sicherzustellen, dass nur zulässige Rollen für Benutzer und Listen vergeben werden.

**20. User Klasse**

Die User-Klasse repräsentiert einen Benutzer im System. Sie enthält die folgenden Eigenschaften:

* Username: Der Benutzername des Benutzers.
* Password: Das Passwort des Benutzers.
* Role: Die Rolle des Benutzers. Die Klasse hat einen Konstruktor, der die Werte für diese Eigenschaften festlegt. Diese Klasse wird verwendet, um Benutzerobjekte im System zu erstellen und zu verwalten.

**21. List Klasse**

Die List-Klasse repräsentiert eine Liste im System. Sie enthält die folgenden Eigenschaften:

* Name: Der Name der Liste.
* OpenRole: Die Rolle, die erforderlich ist, um die Liste zu öffnen.
* EditRole: Die Rolle, die erforderlich ist, um die Liste zu bearbeiten.
* MoveRole: Die Rolle, die erforderlich ist, um die Liste zu verschieben.
* DeleteRole: Die Rolle, die erforderlich ist, um die Liste zu löschen. Die Klasse hat einen Konstruktor, der die Werte für diese Eigenschaften festlegt. Diese Klasse wird verwendet, um Listenobjekte im System zu erstellen und zu verwalten.

**22. User Klasse**

Die User-Klasse repräsentiert einen Benutzer im System. Diese Klasse enthält folgende Eigenschaften:

* **Username**: Der Benutzername des Benutzers.
* **Password**: Das Passwort des Benutzers.
* **Role**: Die Rolle des Benutzers im System. Die Klasse hat einen Konstruktor, der die Werte für diese Eigenschaften festlegt. Diese Klasse wird verwendet, um Benutzerobjekte im System zu erstellen und zu verwalten.

**23. List Klasse**

Die List-Klasse repräsentiert eine Liste im System. Diese Klasse enthält folgende Eigenschaften:

* **Name**: Der Name der Liste.
* **OpenRole**: Die Rolle, die erforderlich ist, um die Liste zu öffnen.
* **EditRole**: Die Rolle, die erforderlich ist, um die Liste zu bearbeiten.
* **MoveRole**: Die Rolle, die erforderlich ist, um die Liste zu verschieben.
* **DeleteRole**: Die Rolle, die erforderlich ist, um die Liste zu löschen. Die Klasse hat einen Konstruktor, der die Werte für diese Eigenschaften festlegt. Diese Klasse wird verwendet, um Listenobjekte im System zu erstellen und zu verwalten.

**24. IsValidRole Methode**

Die IsValidRole-Methode überprüft, ob eine gegebene Rolle gültig ist. Sie akzeptiert einen String als Eingabe und gibt einen booleschen Wert zurück, der angibt, ob die Rolle "admin", "fachanwender" oder "anwender" ist. Diese Methode wird verwendet, um sicherzustellen, dass nur zulässige Rollen für Benutzer und Listen vergeben werden.

Zusammenfassend lässt sich sagen, dass nahezu alle gesetzten Ziele erreicht wurden. Das Projekt konnte die meisten Anforderungen und Erwartungen erfüllen und es wurden viele erfolgreiche Meilensteine erreicht. Die Zusammenarbeit im Team war produktiv und die entwickelten Lösungen haben sich als effektiv und zuverlässig erwiesen.

Jedoch gibt es einen Aspekt, der nicht vollständig umgesetzt werden konnte: die Implementierung des Speichers (Memory). Trotz intensiver Bemühungen und verschiedener Ansätze ist es uns leider nicht gelungen, diese Funktionalität vollständig zu integrieren. Dieser Punkt bleibt somit eine Aufgabe für zukünftige Weiterentwicklungen und Verbesserungen.

\*Die Rechtschreibung des Textes wurde durch DeepL Write überprüft und berichtigt\*