UML Diagramm zur App MK Ablesedienst

Das UML Diagramm zur App stellt alle Klassen im Server-Paket sowie die im App-Paket dar. Aufgrund der vielen Verbindungen sind die Verbindungen oft nicht ganz klar.

Allgemein gilt, dass eine Fragment-Klasse nicht ohne die MainActivity existieren kann uns deshalb hier Assoziationen bestehen. Viele Fragmentklassen sowie die Adapterklassen haben private Unterklassen. Obwohl diese Klassen oft gleich heißen und annähernd die gleichen Funktionen ausführen, ist es sinnvoller diese mehrfach als private Klassen zu deklarieren als einmal als package-private.

Das liegt daran, dass viele dieser Klassen (z.B. ConnectServerTask) einen weiteren asynchronen Thread erstellen (im Vergleich zum Haupthread ist es hier Praxis, diesen privat anzulegen) und somit Daten über das Netzwerk zu kommunizieren und Ergebnisse des Servers zu erwarten. Da diese Threads als potentiell fehleranfällig sind (keine Internetverbindung, ungültige Ergebnisse, andere Fehler) sind diese nicht im Main Thread und damit im eigentliche Fragment erlaubt und müssen ausgelagert werden.

Allgemein gilt zudem, dass fast jede Klasse die Klasse CustomerAppointment seitens der App kennt. Diese Klasse ist eine Hilfsklasse auf der App Seite und spiegelt ein Objekt wieder, welches durch einen Join der Tabellen Customer und Appointment entstehen würde.

Da weder das SQL Ergebnis noch ein in der Serveranwendung definierter Datentyp problemlos über das Netzwerk transportiert werden kann (Serialisierung), bereitet diese Hilfsklasse die übermittelten Daten auf der App Seite erneut auf um sie besser verwenden zu können und um Casting-Fehler zu vermeiden (besonders bei LocalDate und LocalTime).

Auf der Server Seite stellt die Klasse Appointment eine ähnliche Hilfsklasse dar, hauptsächlich für Daten die von der App übermittelt werden. Auch hier werden die Daten aufbereitet um Sie besser nutzen zu können.