Vorname	Nachname	Beruf	Dauer
Marten	Meißner	Fachinformatiker AE	07.22-07.24

Wochenbericht KW 27. (2023.07.03. - 09.)

Für unser aktuelles Projekt »Meeting-Alert« habe ich, das Datenmodell vorerst fertiggestellt. Für die Umformulierung, der Outlook Kalenderdaten, entwarf ich, eine Datenklasse, die alle Daten entgegennimmt und ihren eigenen Attributen zuweist, da diese Umformulierung nicht ganz so einfach ist, konnte ich die Services-packages für die JSON Formulierung und den Code Generator nicht nutzen. Die erste Klasse kann alle Daten entgegennehmen und für uns Verwendung in eigene Attribute speichern. Z.B. hat jeder Termin eine einzigartige ID, um später genau diesen Termin zu finden, kann diese Eigenschaft verglichen werden, ob es sich um den gesuchten Termin handelt. Da wir nicht alle Eigenschaften einer ICalender.ics Datei verwenden oder bearbeiten, sammeln wir diese Informationen trotzdem in einem allgemeinen Attribute. Des Weiteren bekommen wir unsere Informationen immer als Typ String. Wenn unsere Datenklasse nur Strings enthalten würde, wäre das sehr einfach für mit ihr zu arbeiten, aber auch sehr Fehler anfällig. Da unsere erste Damenklasse schon sehr komplex ist, um ICalender Dateien passend zu formatieren und deswegen viel Logik enthält, entschließ ich mich nach dem Prinzip „do one thing and do in well", eine zweite Datenklasse für die Verwendungen in der App zu erstellt. Diese hat eine sehr große Ähnlichkeit zum anderen Datenmodell, aber diese enthält nicht nur String, sondern auch komplexere Datentypen wie DateTime(Uhrzeit). Die zweite Klasse hat die Aufgabe, die Attribute des ersten Datenobjekts entgegenzunehmen und diese in ihre richtigen Typen zu formatieren und kann so leicht als JSON gespeichert oder geladen werden. Des Weiteren diskutierten wir über weitere Architekturentscheidungen. Z.B. nutzen wir klassische Konstruktoren für unsere Viewmodel oder nutzen wir das factory-design-pattern, das hilft uns die Logik in dem Viewmodel zu halten und nicht überall im Code zu entscheiden, welches Objekt erzeugt werden soll. Wir entschieden uns aber für eine Sealed class, dieser Klassentyp ist neu in Dart, Dart wurde mit dem Update Dart-3. um verschiedene Neuerungen erweitert unter anderen weitere Klassentypen. Versiegelte Klassen können nur in der Datei der Oberklasse erstellt und bearbeitet werden. Des Weiteren können Unterklassen nur extendiert werden, also es müssen alle Attribute der Oberklasse übernommen werden. Eine weitere Eigenschaft von versiegelten Klassen ist, dass es leichter ist unterschiedliche Objekte zu erzeugen, weil in der Datei alle Unterklassen definiert sein müssen, fällt der default-fall in einem Switch Statement weg, deshalb kann genau bestimmt werden, unter welchen Kriterien das Objekt erstellt wird.

Kontrolliert am: _____ Unterschrift :_____