| Vorname | Nachname | Beruf | Dauer |
|---|---|---|---|
| Marten | Meißner | Fachinformatiker AE | 07.22-07.24 |

# Wochenbericht KW 36. (2023.09.04. - 10.)

Nachdem nun eine Lösung gefunden wurde, um die Hintergrundprozesse zu steuern, konnte ich mich mehr auf die eigentlichen Prozesse konzentrieren.
Durch meinen Stream kann ich jetzt minütlich den State abfragen und verschiedene Event aktivieren.
Im häufigsten Intervall wird das kleinste Event CheckNotificationEvent().
Dieses Event vergleicht die Anfangszeiten der heutigen Termine und vergleicht bei jedem Durchlauf, wie viel Zeit bis zum nächsten Event ist, wenn die Grenze von fünf Minuten erreicht wird, wird einmal dieses Event aktiviert und weist auf ein Termin hin.
Da diese Funktion, mit einer Asynchron Markierung markiert wurde, wird der Main Thread nicht blockiert, wenn die Funktion eine längere Rechenzeit benötigt und währenddessen weiter Prozesse abarbeiten.
Wie z. B. der nächste Hintergrund Aktualisierung CreateAppointmentVM(), die alle fünf Minuten die Liste von Appointments Viewmodels neu berechnet.
Dadurch werden alle fünf Minuten bereits begonnene Termine automatisch aus dem Speicher gelöscht und verwirren den Nutzer nicht beim nächsten Nutzen der App.
Dies kann Sinn ergeben, da im nicht größeren Intervall, LaodCalenderData() die Datenmodelle der Termine aktualisiert werden. Dies war auch ein Hauptgrund, weshalb wir diese App entwickeln, da viele andere Produkte wie von Google oder Microsoft ihre Kalender und Termine nur einmal am Tag erneuern.
Da wir schon für eine spätere Bezahlversion, planen auch mehrere Kalender verwalten zu können, haben wir den Intervallen recht klein konfiguriert, damit wenn später z. B. sechs Kalender verwaltet werden, einmal je Stunde alle Kalender einmal aktualisiert werden.
Sehr Speicher und Daten intensiv wird dies aber nicht, da eine ganze ICalender Datei(.ics) meist nicht mehr als 15 Kbit(ca.100 Termineinträge) umfasst, dies sich auf einen Tag auf 2,16Mbit’s addiert und auf ein Jahr hochgerechnet 780 Mbit’s oder 98 MByte’s (93 Mebi/Bytes).
Da wir dies nicht zu statisch gestalten wollen, haben wir dafür auch eingeplant, diese Werte später selbständig in den Einstellungen, einstellen zu können.
Die Daten bekommen wir über eine Get abfrage des jeweiligen Kalenders Betreiber.
Die ankommende ICalender Datei, wird dann über eine Regular Expression ausgefiltert und anschließend zu einem unserer Datenmodelle umformatiert.
Eine größere Aufgabe war dann die Erstellung neuer eigener Termine auf Basis der ICSTermin Vorlage, da diese in einem recht kryptischen String alle Informationen verpackt, die beschreiben, wie ein wiederkehrender Termin aufgebaut wird.
Da es dafür viele Möglichkeiten gibt, hat diese Aufgabe auch viel Zeit in Anspruch genommen.
Am Einfachsten war dann eine Logik zu Bauen für Termine die täglich oder wöchentlich Stattfinden, da diese für jeden Tag oder für den jeweiligen Wochentag gebaut werden.
Schwieriger wurde es dann, bei monatlichen Terminen.
Weil diese entweder am selben Tag stattfinden wie z.B. immer am 28 des Monats überweise ich meine Miete oder die Miete wird immer am Letzten des Monats überwiesen, dies kann mal am 30, 31, aber manchmal am 28, 29 stattfinden.

Oder ein Termin soll immer am dritten Mittwoch des Monats stattfinden, nach einigem überlegen und versuchen, stellte ich dann fest, dass es am einfachsten ist eine Liste mit Monatstag und dem dazugehörigen Wochentag anzulegen. Damit ich auch sicher gehen konnte, dass die Funktion diese Informationen auch richtig ausfiltert und mir ein Termin erstellt, musste ich dies mit Unittests überprüfen.
Da ich es sonst nicht unabhängig kontrollieren kann, was die Funktion ausführt und ob am Ende auch die richtigen Ergebnisse auftauchen

Kontrolliert am: _____ Unterschrift :_____