Vorname	Nachname	Beruf	Dauer
Marten	Meißner	Fachinformatiker AE	07.22-07.24

Wochenbericht KW 24. (2023.06.12. - 18.)

Für unser aktuelles Projekt „Meeting_Alert“ haben wir uns mit Regular Expression gekümmert. Unsere App akzeptiert nun eine URL, auf diese URL fragen wir eine get Abfrage und erhalten eine ICS Datei zurück. In dieser Datei stehen alle Informationen, die wir für die einzelnen Termine benötigen. Um die Informationen daraus zu erhalten, nutzen wir jetzt Regular Expression, dies erlaubt uns, die ICS Datei so zu zergliedern, dass die Termine gruppiert auszulesen. Mit weiteren Reg Ex aufrufen haben wir anschließend die Terminpakete weiter aufgegliedert und als Map in eine Liste gespeichert. Diese Liste, Map, String, Dynamik speichern wir als String(JSON) lokal auf unserem PC, dafür nutzen wir eine Dart Funktion getApplicationDocumentsDirectory(), die einen Ablageort und dessen Pfad definiert, dazu müssen wir nur noch den Dateinamen definieren. Regular Expression kommt aus der theoretischen Informatik und ist daher in so ziemlich jeder Programmiersprache nutzbar. Reg Ex wird in Dart wie folgt geschrieben RegEx('r/xxxxx/); , Reg Ex hat in Dart seine eigene Klasse und wird in einem String definiert und steht deswegen in Anführungszeichen und beginnt mit einem kleinen r gefolgt bzw. abgeschlossen mit je einem Slash, die sogenannten delimiter und dienen als Begrenzung. Reg Ex hat auch eine recht komplizierte Syntax, da bestimmte Zeichen schon vordefiniert sind, wie z.B. ein Plus-Zeichen, dieses kann als quantifier benutzt werden und verbindet zwei ausdrücke oder es kann eine Reihe von einem oder mehren Zeichen gruppieren und ist dadurch auch ein sogenannter Meta-Charakter. Meta Charaktere wurden bestimmt, um verschiedene aussagen zu bestimmen, klassisch hier ist der Ausdruck \w, \w symbolisiert jeden aphlanummerischwert also a-z und 0–9 und schließt dazu noch Unterstriche ein, dies kann auch als Gegenteil verwendet werden z.B. \W, hier wird jetzt jeder Alphanummerischerwert ausgeschlossen. Hier sind die bekanntesten Werte“ \d (Zahlen), \w (Buchstaben), \s(whitespace alle nicht lesbaren Charaktere wie Leerzeichen oder Zeilenumbruch) „. Zu den Meta Charakteren kommen dann sogenannte quantifier diese könne genutzt werden, um die gefundenen Treffer zu modifizieren, so kann mit einem Punkt der nächste Charakter nach jedem Treffer mit markiert, dies schließt auch nicht lesbare Treffer ein wie ein Zeilenumbruch oder Leerzeichen. Ein sehr unsauber Befehl aus „greety“ bezeichnet, ist ein .* dies sagt aus das es alle folgende oder nichts auswählt und dann keine Ausnahmen macht. Ein solcher Befehl könnte so aus sehen „r/(.*)/“, hier wird kein Zeichen ausgewählt oder so viele es findet bis hin zum gesamten Text mit allen Zeilenumbrüchen und Leerzeichen, dies ist sehr unsauber, weil es schlecht zu regulieren ist. Um die Termine als eine Information zu sammeln haben wir wie folgt in der ICS Datei gesucht und gefiltert, „r/?<=BEGINN:VEVENT\r(.|\r\n)+?(?=END:VEVENT)/“ In Worte hieß hier der Befehl (?<=) Markiere nach den Charakteren BEGINN:VEVENT und einem Zeilenumbruch markiert (lookbehind). Als Nächstes wird gesagt „(.|\n)+?“ die Klammer markiert eine Gruppierung von Charakteren, der Punkt steht für das nächste Zeichen, | steht für ein oder und erlaubt in der Gruppierung auch Zeilenumbruch, das Folgende Plus sagt wieder aus, Markiere ein Zeichen oder so viele wie du findest. Mit dem Fragezeichen wird aus dem noch der Gruppierung Ausdruck verfeinert und sagt gruppiere so häufig wie es möglich ist, ohne diese Fragezeichen würde unser Befehl nur eine Gruppe Markieren, die von dem Ersten BEGIN:VEVENT bis zum letzten END:VEVENT markiert wird. (?=END:VEVENT) heißt lookahead und schließt vor dieser Gruppe die Markierung ab.

Kontrolliert am: _____ Unterschrift :_____