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

# Wochenbericht KW 25. (2024.06.17 - 23.)

Für eine allgemeine Kundenübersicht und deren Projekte wurde vor Wochen schon ein GUI-Design umgesetzt, aber noch nicht an eine API angebunden. Momentan sind alle angezeigten Daten Hardcoded in das File geschrieben.
Die API habe ich wie üblich über meine API Adapter Klasse implementiert. Hierfür nutze ich private Attribute für die unterschiedlichen URLs der API. Für die Internetabfrage nutzte ich das Flutter Dio-Package, da diese nützliche Erweiterungen, wie vorkonfigurierte Request Methoden, die Übergabeparameter könne hier Standard mäßig mehr als Strings sein wie z. B. „FormDatas“. Ein weiterer Vorteil ist das ich bei der Konstruktor Überladung ich den „DioInterceptor“ konfigurieren kann. Wie der Name schon vermuten lässt, kann ich so vor, während und nach der API anfrage verschiedene Parameter anpassen. Z.B. vor der Anfrage kann die URL bearbeitet werden oder den URL Header updaten. Dies wird benötigt, da die Datenbank bei Schreib, Änderung oder Lösch anfragen einen Bearer Authentifikation Token verlangt. Durch die Überladung des Adapter Konstruktor kann ich vor der Abfrage, den HTTP-Header konfigurieren.
Durch die Kapselung der Logik für das Internet anfragen in der API Adapter Klasse, benötige ich für den Aufruf lediglich eine API Instanz über deren Schnittstellen ich meine Datenbankabfragen aufrufen kann.
Nachdem ich alle Daten aus der Datenbank erhalte und in die GUI übergeben habe, stieß ich auf ein weiteres Web Problem. Da wir eine Webapp bauen und die Anfragen über verschiedenen System(z.B. Google Chrome Browser und Microsoft Datenbank) vorgenommen werden, konnten meine Bilder nicht richtig angezeigt. Nach sehr kurzer recherche sties ich auf die Ursache weshalb die Bilder nicht angezeigt über das Internet angezeigt werden konnte.
Der Flutter Standard Web Kompilierer nutzt das sogenannte Canvas Render-Kit. Dies ist die erste Wahl für anspruchsvolle Web Anwendungen mit Animation und vielen Daten und Pixel genauer UI. Da es viele UI Elemente vorlädt und zur Laufzeit nur kleine Änderungen durchführt, ist es schneller für Web und Mobile Webseiten als das HTML-Render-Kit.
Das HTML-Render-Kit ist schneller bei dem Laden der Webseite, da die UI-Elemente nicht vorgeladen werden. Deshalb sind Webapps, die mit dem HTML-Render-Kit gebaut worden, schneller geladen, aber nicht so effizient wie im Canvas-Renderkit gebaut.
Mein Fehler entstand dadurch, dass das Canvas Render-Kit von der API die Bilder als Bytes pro Pixel benötigt, da durch die CORS Regelung jedoch die Bilddaten aus Sicherheitsgründen nicht in dem Detail freigeben werden, konnten die Bilder mit dem Canvas-Renderkit nicht anzeigt werden. Nach Absprache mit dem Projektleiter und Abwiegen der Anwendungsansprüche, entschieden wir uns dafür, das Render-Kit zu ändern. Weil keine Komplexen Animaionen oder Pixelgenaue GUI Elemente angestrebt werden.

Kontrolliert am: _____ Unterschrift :_____