

Abschlussprüfung Winter 2022

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

[REDACTED]

AAIC API-Mobiler-Client

Mobiler-Client für den Zugriff auf die Posteingangsmanager

API-Schnittstelle der Sage 100 Warenwirtschaft

Prüfungsbewerber:

[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]

Ausbildungsbetrieb:

[REDACTED]
[REDACTED]
[REDACTED]

[REDACTED]



Ausbilder:

[REDACTED]

Inhaltsverzeichnis

1.EINLEITUNG	1
1.1 ABWEICHUNGEN ZUM PROJEKTANTRAG.....	1
1.2 PROJEKTZIEL.....	1
1.3 PROJEKTUMFELD.....	1
1.4 PROJEKTBEGRENZUNG	2
1.5 PROJEKT BETREUUNG.....	2
2 PROJEKTPLANUNG	3
2.1 PROJEKTPHASEN.....	3
2.2 ENTWICKLUNGSPROZESS.....	3
2.3 RESSOURCENPLAN.....	3
3 ANALYSEPHASE	4
3.1 IST-ANALYSE:	4
3.2 SOLL-ANALYSE:	4
3.3 PROJEKTKOSTEN.....	5
3.4 „MAKE OR BUY“ ENTSCHEIDUNG.....	6
3.5 WIRTSCHAFTLICHKEIT DES PROJEKTES.....	6
4 ENTWURFSPHASE	7
4.1 ARCHITEKTUR	7
4.2 ZIELPLATTFORM.....	7
4.3 DATENMODEL.....	7
4.4 GESCHÄFTSLOGIK.....	8
4.5 VERWALTUNG VOM AUFBAU	8
4.6 PFLICHTENHEFT.....	8
5 IMPLEMENTIERUNGSPHASE	9
5.1 IMPLEMENTIERUNG DER DATENSTRUKTUR:.....	9
5.2 IMPLEMENTIERUNG DER GESCHÄFTSLOGIK:	9
5.3 IMPLEMENTIERUNG DER KLASSEN:.....	9
5.4 IMPLEMENTIERUNG DER OBERFLÄCHEN:	10
5.5 IMPLEMENTIERUNG DER METHODEN:.....	10
5.6 GRUNDBAUSTEINMETHODEN:.....	10
<i>Klasse: TokenService:</i>	<i>10</i>
<i>Klasse: SageldToken:</i>	<i>11</i>
<i>MauiProgram-Klasse:.....</i>	<i>11</i>
<i>AppShell.xaml.cs-Klasse:</i>	<i>11</i>
6 QUALITÄTSMANAGEMENT	12
6.1 MANUELLE TESTS	12
6.2 FEHLERBEHEBUNG.....	12
7 FAZIT	13
7.1 ABNAHME.....	13
7.2 SOLL-IST-VERGLEICH.....	13
7.3 ZUKUNFTSORIENTIERUNG	14
ANLAGEN:.....	15
A. ANHANGSVERZEICHNIS.....	15
I. PFLICHTENHEFT	15

II. VERWENDETE NUGET-PAKETE UND BIBLIOTHEKEN.....	16
II.1 Klassenbibliothek:	16
II.2 .NET MAUI Anwendung:.....	16
III. KLASSENDIAGRAMM.....	17
IV. SCREENSHOTS ZUM QUELLCODE DER MANUELLEN TESTS.....	19
IV.1 => TokenService.....	19
IV.2 => Lizenz und Datenbank Auswahl.....	20
V. VERWENDETE RESSOURCEN.....	21
V.1 PERSONAL.....	21
V.2 SOFTWARE.....	21
V.3 HARDWARE	21
VI. USE-CASE DIAGRAMM.....	22
VII. BENUTZERHANDBUCH.....	23
VIII. SCREENSHOTS ZUM QUELLCODE.....	25
B QUELLENANGABEN	47
Webauthentifikator: Fehlerbildbehebung:.....	47
Authentifikationslösung:.....	47
Verwenden von Webauthenticator:	47
Sage 100 API-Schnittstellen:	47

1.Einleitung

1.1 Abweichungen zum Projektantrag

Im vorangegangenen Projektantrag wurde „.NET“ und „Xamarin“ für die Erstellung eines Mobilien Client´s angegeben, durch ein Update von Microsoft wurde die „.MAUI“ Plattform als Nachfolger für „Xamarin“ bereitgestellt. Aus diesem Grund wird „.MAUI“ anstelle von „Xamarin“ verwendet.

Der Vorgang zwischen diesen beiden Plattformen bleibt identisch. In der Projektarbeit wird die Artikelstammliste über eine API-Schnittstelle angesprochen und anstatt vom Posteingangsmanager ausgegeben. Dies bezieht sich auf ein internes Fehlerbild und wird im Dokumentationsablauf als Posteingangsmanager gehandhabt, die Schritte sind dabei bis auf kleine Abweichungen von der grafischen Benutzeroberfläche identisch. Die grafische Benutzeroberfläche wird händisch geschrieben, das DevExpress Framework kommt nach Fertigung der Projektarbeit in Benutzung, um die von der IHK vorgegebenen 80 Stunden einzuhalten.

1.2 Projektziel

Ziel des Projektes ist der Aufbau eines Programms „Mobiler Client“, der Inhalte der Sage 100 Warenwirtschaft aus dem Modul „PEM“ (Posteingangsmanager) zur Verfügung stellt und PDF's ohne der Notwendigkeit eines Downloads anzeigt, ohne dass man eine Sage ID Anwender:innen - Lizenz benötigt. Der Mobile Client soll nativ sein, also auf vielen verschiedenen Betriebssystemen funktionieren (zum Beispiel: Android, Windows). Durch den Zugriff auf das PEM-Modul der API-Schnittstelle, soll die Verwaltung oder Einsicht in die Dokumentenablage dieses Moduls sowohl online als auch lokal gewährleistet werden.

1.3 Projektumfeld

_____ fungiert als Praktikumpartner und gleichzeitig als Auftragsgeber. _____ Fachhändler betreut die _____ über 400 Kund:innen in Mitteldeutschland im Umgang mit Sage Produkten. Darüber hinaus bietet die _____ als Systemhaus auch Dienstleistungen im Bereich Consulting und Schulungen sowie Server-Infrastruktur an. Die Aufgaben der ca. 20 Mitarbeitenden reichen vom Vertrieb, Installationen, Consultings und Beratungen hin bis zu kundenspezifischen Anpassungen der Software und Netzwerk- und Server-Betreuung.

Mitunter die wichtigsten Produkte der [REDACTED] sind:

1.4 Projektbegrenzung

Das Posteingangsmanager Modul befindet sich aktuell in der weiteren Entwicklungs- und Fehlerbehebungsphase. Aus diesem Grund ist eine Fehlerquote möglich. Zudem ist durch die Vorgabe der IHK das Projekt mit 80 Stunden begrenzt. Eine weitere Fehlerbildquelle entspringt dem Betriebssystem Windows Server 2022. Das Betriebssystem ist nicht lange auf dem Markt und enthält noch nicht alle für eine vollständige Windowsumgebung notwendigen Patches und Updates. Aus diesem Grund wird das Projekt auf einem Windows 10 lokalem Host-Gerät geschrieben und später zur Ausführung auf den Windows Server 2022 übertragen.

1.5 Projektbetreuung

Während der Projektzeit wird der Praktikant und Autor vom Ausbilder [REDACTED] der [REDACTED]

2 Projektplanung

2.1 Projektphasen

Die Umsetzung des Projektes erfolgt in den von der IHK vorgegebenen 80 Stunden und ist in verschiedene Phasen aufgeteilt. Die Umsetzung des Projektes erfolgt ausschließlich am Arbeitsplatz der [REDACTED] und wird, bis auf die Projektdokumentation, in den regulären Arbeitszeiten erstellt und getestet. Die Abnahme erfolgt betriebsintern vom Ausbilder [REDACTED] innerhalb der Prüfungsvorbereitungszeit des Autors.

Analysephase	12 Stunden
Entwurfsphase	15 Stunden
Implementierung	40 Stunden
Qualitätsmanagement	3 Stunden
Erstellen der Dokumentation	10 Stunden

2.2 Entwicklungsprozess

Der Autor wählte bereits vor dem Projektantrag die Entwicklungsumgebung.

Zu Beginn des Projektes wird über die Entwicklungsschritte für die Umsetzung unterschieden und entschieden. Zur Orientierung wurde das Wasserfall-Modell für die Softwareentwicklung gewählt, die Schritte werden inkrementell und iterativ behandelt. Für die Methoden die implementiert werden erfolgen Unit-Tests. Damit schreiten die Implementierungs- und die Testphase parallel zu einander voran. Innerhalb dieses Konzeptes werden mögliche Fehlerbilder bereits in der Implementierungsphase der einzelnen Schritte erkannt sowie behoben. Der Implementierungsvorgang kann somit ebenfalls inkrementell dokumentiert werden.

2.3 Ressourcenplan

Die Ressourcen der Entwicklungsumgebung und der Arbeitsumgebung beziehen sich auf die Angaben im Projektantrag. Im Anhang wird eine Auflistung dieser hinzugefügt. [\[Anhang V. Verwendete Ressourcen\]](#)

3 Analysephase

3.1 Ist-Analyse:

Die [REDACTED] ist ein Enterprise Resource Planing (ERP) Programm bestehend aus Warenwirtschaft und Rechnungswesen. Sie ermöglicht das Planen, Verwalten sowie das Steuern von Personal und Ressourcen, wie Kapital, Material, Informationstechnik und Betriebsmittel. Zudem bietet die [REDACTED] ein Rechnungswesen, um die buchhalterischen Prozesse eines Unternehmens abzubilden. Der Zugriff auf die Warenwirtschaft oder das Rechnungswesen erfolgt über eine Client-Installation auf einem Windows-PC und erfordert eine entsprechende kostenpflichtige Nutzer:innen - Lizenz (Named-User-Lizenz).

Der Posteingangsmanager ist eine Eigenentwicklung der [REDACTED]. Dabei handelt es sich um eine digitale Postmappe, diese bietet über Workflows die Möglichkeit die Bearbeitung von Dokumenten bis zur Verbuchung vollständig innerhalb der [REDACTED] durchzuführen. Die digitalisierten Dokumente, die meist als PDF vorliegen, werden auf dem ERP-Server gespeichert und zur Darstellung als Byte-Stream zur Verfügung gestellt.

Für den Zugriff auf den Posteingangsmanager ist eine [REDACTED] Nutzer:innen – Lizenz obligatorisch, jedoch ermöglichte Sage mit dem Update auf die Version 9.0.4 einen externen Zugriff über API-Schnittstellen. Darüber hinaus ist mit dem Update auch möglich, eigene Schnittstellen zu erstellen und Anwender:innen über externe Programme Zugriff zu gewähren.

3.2 Soll-Analyse:

Der Autor ist beauftragt einen Mobilen Client der [REDACTED] zur Ansprache der Posteingangsmanager API-Schnittstelle zu entwickeln. Mithilfe von dem Mobilen Client soll der Zugriff auf den Posteingangsmanager gewährt werden. Dadurch wird es den Mitarbeit:erinnen der [REDACTED] ermöglicht von mehreren verschiedenen Geräten Betriebsdokumente aus der Sage 100 aufzurufen, zu überwachen oder zu bearbeiten. Das Endziel ist es den Mobilen Client [REDACTED] Kund:innen kommerziell als eigene Leistung anzubieten und zu verkaufen. Die [REDACTED] benötigt zur Nutzung der Warenwirtschaft eine Named-User-Lizenz für alle Mitarbeiter:innen. Der Mobile Client der [REDACTED] ermöglicht die Ansprache des Posteingangsmanager-Moduls über eine API-Schnittstelle und erfordert für viele Anwender:innen nur eine [REDACTED] Named-User-Lizenz. Zugleich ist der Lizenz Preis zur Nutzung des [REDACTED] Clients kostengünstiger im Vergleich zur [REDACTED] Lizenz und ermöglicht somit allen Parteien einen wirtschaftlichen

Vorteil. Das Arbeiten von unterwegs bietet sich durch die Plattformunabhängigkeit über ein Smartphone oder ein Tablet mit den Betriebssystemen Android, iOS und Windows an.

3.3 Projektkosten

Im Folgenden werden alle anfallenden Personalkosten anhand eines internen Verrechnungssatzes berechnet. Der Auszubildende wird mit Kosten für Praktikant:innen des Software-Entwickler-Teams angegeben. Zugleich werden entstandene Kosten für Ressourcen und andere Kosten als Gemeinkosten berücksichtigt. In den Lohngemeinkosten werden Kosten der Nutzung von Soft- und Hardware ermittelt. Verwaltungsgemeinkosten stehen entsprechend für den Verwaltungsaufwand. Alle Angaben sind fiktiv.

Folgende Projektkosten entstehen für Stundensätze und Gemeinkosten:

- Auszubildender wie Autor 10 €
- Ausbilder und Entwickler 120 €
- Mitarbeiter der Hardwareabteilung 120 €
- Lohngemeinkosten 70%
- Verwaltungsgemeinkosten 10%

Phase	Angestellte	Aufwand	Summe
Entwicklung	Auszubildender	80 Stunden	800 Euro
Bereitstellung des Test-Servers	Angestellter der Hardwareabteilung	2 Stunden	240 Euro
Unit-Test Abnahme	Ausbilder Auszubildender	2 Stunden	240 Euro
Abnahme des Projektes	Ausbilder	1 Stunde	120 Euro
Summe			1320 Euro

Personalkosten gesamt: **1320 €**

• Lohngemeinkosten: 924 € (70%)

• Verwaltungskosten: 132 € (10%)

Die resultierenden Projektkosten beziehen sich entsprechend auf: **2376 €**

3.4 „Make or Buy“ Entscheidung

Das Unternehmen [REDACTED] GmbH ist wie bereits im [Punkt 1.3 Projektumfeld](#) beschrieben ein eigenständiges Software-Entwicklungs-Unternehmen. Zugleich wird der Mobile Client zu kommerziellen Zwecken erstellt und angeboten. Anhand dieser beiden Punkte resultiert sich eindeutig die „Make“ Entscheidung. Es besteht für das Unternehmen [REDACTED] kein Bedarf eines Auftrages an ein externes Unternehmen, da es in der Lage ist selbstständig Lösungen zu entwickeln und umzusetzen.

3.5 Wirtschaftlichkeit des Projektes

Der Gesamtaufwand des Projektes liegt bei 2376 €. Der Mobile Client kostet in der Anschaffung für Kund:innen 4999€ beim Kauf, inklusive 5 Nutzer: innen-Lizenzen, weitere Nutzer: innen-Lizenzen lassen sich für je 50 € dazu erwerben. Dabei handelt es sich um Life-Long-Lizenzen. Das Angebot einer Lizenz zur monatlichen Nutzung ist geplant und im Errechnungsprozess, zurzeit steht noch kein Angebot dafür fest.

Daraus lässt sich der Break-Even-Point errechnen. Bereits nach einem Verkauf des Mobilen Clients, liegt das Unternehmen [REDACTED] in den schwarzen Zahlen und deckt jegliche Kosten der Entwicklung des Mobilem Clients ab, der Umsatz aus dem ersten Verkauf liegt dabei bei 2623 €.

Da Wartungs- oder Updatearbeiten sowie Support im Lebenszyklus der Software eine Rolle spielen, aber in diesem Fall in der Wirtschaftlichkeitsanalyse eine untergeordnete Position haben, werden jene Punkte im Abschnitt 7 noch einmal diskutiert.


4 Entwurfsphase

4.1 Architektur

Als native Applikation soll das Programm auf nahezu allen Plattformen zur Verfügung stehen.

Mit .NET MAUI über das Visual Studio 2022 IDE wird ermöglicht die Applikation auf beinahe allen Betriebssystemen zur Verfügung zu stellen und zu nutzen, der Autor entschied sich für das Betriebssystem Windows 10. Der Quellcode wird in C# und XAML geschrieben. Es stehen viele kostenfreie NuGet-Bibliotheken für die Erstellung zur Verfügung.

4.2 Zielplattform

Die Zielplattformen sind die Android und Windows Betriebssysteme, sowie MacOS und iOS. Für die beiden letzten genannten Betriebssysteme ist die entsprechende Hardware, wie das iPhone, das iPad, der iMac oder das MacBook zum Testen des Programmes notwendig. Diese Art der Hardware steht dem Autor nicht zur Verfügung. Aus diesem Grund liegt die Konzentration auf dem Betriebssystem Windows 10 mit künftigem Ausbau für Android. Linux unterstützt .NET MAUI und  nicht und wird daher keine Rolle im Projektplan einnehmen.

4.3 Datenmodel

Für die Umsetzung des Projektes werden folgende Projekte in einem leerem Projektordner angelegt:

.NET MAUI- APP	.NET 6.0	<= für die plattformunabhängige Ausführung
Klassenbibliothek	.NET Standard 2.0	<= allgemeine Bibliothek für Klassen
MS-Test – Projekt	.NET 6.0	<= Projekt für Unit-Test-Methoden

Für das User Interface steht die Erweiterung „DevExpress“ im Visual Studio 2022 auf einem firmeninternen Test-Server fertig vorbereitet. Das Framework bietet fertige User Interfaces die zur Implementierung nach der Fertigung der Projektarbeit übernommen werden.


Die verwendeten NuGet - Zusatzbibliotheken sind im Anhangsverzeichnis aufgelistet.

[\[Anhang III. Verwendete NuGet-Pakete und Bibliotheken\]](#)

Anhand der Berechtigung ID´s (Entitlement ID) der Anwender: innen können Zugriffe auf Daten und Datenbanken im Modul verwehrt oder gewährt und aufgelistet werden.

4.4 Geschäftslogik

Der Mobile Client ist ein komplettes Programm, dass neu geplant und aufgebaut werden muss. Der Prozess wird in einzelne Aufgabenbereiche aufgeteilt und findet inkrementell, sowie iterativ statt. Damit wird sichergestellt, dass jeder Schritt fehlerfrei ist bevor die Implementierung des nächsten begonnen wird. Der Autor hat sich für sieben Schritte entschieden, die wie folgt aufgeteilt werden:

Einrichtung einer Projektmappe, Zeichenkettenkonvertierung für den Isolierten Speicher, Anmeldung an der  ID, das User Interface für die Anmeldung, Auswahl der Lizenz und Datenbanken, Ansprechen der API-Schnittstelle, sowie das User Interface für die Ausgabe der Daten aus der API-Schnittstelle.

Es wird sichergestellt werden, dass die Anmeldedaten im isolierten Speicher sicher abgelegt und daraus aufgerufen werden können. Die Anmelderessourcen müssen nach einer Ablaufzeit von acht Stunden aktualisiert werden.

4.5 Verwaltung vom Aufbau

In dem Aufbau ist der Authentifikationsfehler in .NET MAUI zu beachten, aktuell ist eine Standard-MAUI-Lösung nur mit Windows ausführbar, andere Plattformen erfordern zusätzliche Manipulationen. Diese sind von Microsoft auf Ihrer Web Seite beschrieben und Lösungsvorschläge im GitHub angegeben und verlinkt.

Microsoft Authentifikation Fehlerbericht: **Microsoft:** <https://learn.microsoft.com/de-de/dotnet/maui/platform-integration/communication/authentication?tabs=windows>

4.6 Pflichtenheft

Zum Ende der Entwurfsphase, folgt die Erstellung des Pflichtenheftes. Ein Auszug des Pflichtenheftes wird im [\[Anhang II.Pflichtenheft\]](#) mit angegeben.

5 Implementierungsphase

5.1 Implementierung der Datenstruktur:

Bevor die Implementierung der Geschäftslogik stattfinden kann, muss eine Struktur der einzelnen Schritte bedacht und angelegt werden. Die Geschäftslogik wurde somit auf sieben Schritte aufgeteilt um nach dem Wasserfall-Modell für den jeweiligen einzelnen Schritt sukzessiv iterativ voranzuschreiten. Somit befasst sich der Autor direkt zum Abschluss eines Schrittes mit dem Unit-Test und kehrt zur Implementierung zurück, sollten Fehlerbilder aufgetreten sein. Auf diese Art geht er sicher, dass keine objektiven Fehlerbilder in der aktuellen Geschäftslogik entstehen. Erst darauf wird die Implementierung des nächsten Schrittes initiiert. Für eine strukturierte Arbeit und Übersicht, werden in einer Projektmappe, drei Projekte erstellt und mit Verzeichnissen belegt, die dafür da sind, bestimmte Klassen in bestimmte Bereiche abzulegen.

5.2 Implementierung der Geschäftslogik:

Die Implementierung der Geschäftslogik ist das Herzstück der Projektarbeit, daher ist eine richtige Entwicklungsumgebung absolut relevant. Das Visual Studio 2022 IDE der Community Edition ist hierbei die Wahl des Autors gewesen. Die für das Projekt notwendigen und genutzten NuGet-Pakete sind im [\[Anhang-NuGet\]](#) aufgelistet und in ihren Funktionen beschrieben.

Da eine Beschreibung des Oberflächendesigns nicht relevant in der Implementierung ist, wird auf das [\[Benutzerhandbuch\]](#) im Anhang verwiesen. Mit der einfach gehaltenen grafischen Oberfläche, sind die in der Implementierungsphase beschriebenen Punkte nachzuvollziehen.

5.3 Implementierung der Klassen:

Durch das [\[Klassendiagramm\]](#) im Anhang ist die Implementierung der Klassen für die Klassenbibliothek und für das MAUI-Projekt gut dargestellt. Dabei handelt es sich um separate Projekte. Die Klassenbibliothek beinhaltet die Methoden zu der eigentlichen Logik zum Anmelden, Speichern und Aufrufen der Daten zur Anmeldung. Im MAUI-Projekt werden darauf die Klassen zur Nutzung der Geschäftslogik implementiert. Beide Projekte beinhalten zugleich Klassen, die aus JSON durch das Abfangen von Fehlerbildern und fehlerfreien Aufrufen entnommen wurden. Im Programm wird außerdem die Klasse zum Browseraufruf von der durch Microsoft angebotenen Lösung hinzugefügt und genutzt. Jegliche Test-Klassen, sowie die Interfaces, wurden entweder durch die Option der Schnelloptionen erstellt oder durch NuGet-Pakete bereitgestellt.

5.4 Implementierung der Oberflächen:

Der Autor nutzt im Projekt die Model-View-ViewModel Option. Daraus ergibt sich eine einfache Erstellung und Nutzung der Oberflächen, angebunden an die Geschäftslogik der Klassen im Codebehind. Alle Oberflächen müssen für die Funktionalität des Programms über die allgemeine Klasse „MauiProgram.cs“ angegeben werden und mit den dafür entsprechenden Klassen verbunden werden. Jegliche Oberflächenelemente wurden mit .XAML im MAUI-Projekt erstellt und bearbeitet.

5.5 Implementierung der Methoden:

Durch das Klassendiagramm, wie in der Implementierung der Klassen erwähnt, ergibt sich eine gute Übersicht über die eingesetzten Methoden. Dabei wird separiert zwischen den allgemeinen Funktionen, wie Token erhalten, speichern und löschen, als auch die Methoden zur Anmeldung über das MAUI-Projekt zur Anmeldung/Abmeldung mit Zugriff auf die allgemeinen Methoden. Zugleich werden Interfaces eingesetzt um nicht unnötig oft Klassen zu vererben und die Übersicht entspannt zu halten.

5.6 Grundbausteinmethoden:

Klasse: TokenService:


SaveTokenAsync(ISageldToken token)

Wird angewendet um das erhaltene Token in abgesicherten 64-Byte-Code zu konvertieren und im isolierten Speicher abzulegen. Diese Methode ist immer dann notwendig, wenn eine Anmeldung ausgeführt wurde und der Token bis zum Ablauf seiner Gültigkeit genutzt werden soll, ohne sich zu jeder Benutzung des Programms neu anmelden zu müssen. Daher ist es möglich mit einer lokal angelegten Datenbank, nach der ersten Anmeldung innerhalb der Ablaufzeit von 8 Stunden, auch ohne Internet zu arbeiten, solange man sich im internen Netzwerk aufhält.

DeleteToken()

Die Methode zum Löschen des Tokens sucht anhand vom Scope, ob ein Token im isolierten Speicher hinterlegt ist und entfernt diesen, sollte einer gefunden werden.

GetTokenAsync()

Sucht nach einem bereits abgelegten und bestehenden Token im isolierten Speicher und entschlüsselt diesen aus dem abgesichertem Base-64-Code Format, um diesen anschließend als einen IdToken an die aufrufende Klasse zu übergeben. Des Weiteren wird die Methode ToSecureString() für die Inhalte angesprochen.

Klasse: SageldToken:

GetOidcClientOptions()

Diese Methode greift auf die Displayeinstellungen des Endgerätes auf dem das Programm ausgeführt wird. Damit wird erfragt, über welche Bildschirmauflösung das Endgerät verfügt, ob es in der Ausführung Fehlerbilder gab und fängt diese zur Übergabe ab, sollten Fehlerbilder entstanden sein. Jegliche Information zu den Anmeldedaten werden abgefragt und in die SageldRequestResult Klasse abgelegt.

Ebenfalls beinhaltet die Klasse Methoden zum Anmelden oder Abmelden, die auf die Basic-Methoden der TokenService Klasse zugreifen.

GetEntitlementAsync(ISageldToken sageldToken) und GetDatasetsAsync(ISageldToken sageldToken, int entitlementid). Diese Methoden empfangen anhand des Tokens nach der Anmeldung die Berechtigungen für Partnerlizenzen, sowie freigegebene Datenbanken, um diese später auf einem Oberflächenelement der Anwender:innen zur Auswahl und Verfügung zu stellen.

MauiProgram-Klasse:

Sorgt für die Erstellung des eigentlichen Programms und für eine globale Verknüpfung jeglicher im Projekt erhaltener und für das Projekt notwendiger Klassen.

AppShell.xaml.cs-Klasse:

Ist für das Routing der Klassen zur Auflistung der angeforderten Inhalte über Namens- und Typverknüpfungen.

6 Qualitätsmanagement

6.1 Manuelle Tests

Durch Unit-Tests wurden die Methoden zum Erhalten, Speichern und Löschen der Token Daten geprüft. Weitere Methoden, wie der Login und der Logout, basieren auf den erst genannten Methoden und wurden somit ausgelassen oder zum Teil getestet. Die Unit-Tests wurden via Screenshot im Anhang hinterlegt.

[\[Anhang IV.1 TokenService\]](#)

Ebenfalls mit Unit-Tests wurde die Auswahl der Lizenz und Datenbanken getestet, da dieser Schritt in der Implementierung einen großen Anteil hatte, wurde sichergegangen, dass alles wie gewünscht funktioniert. Im Anhang finden Sie Screenshots dazu.

[\[Anhang IV.2 Lizenz und Datenbank Auswahl\]](#)

Da des Weiteren alle Bereiche über eine grafische Oberfläche verfügen, wurden die Tests manuell durchgeführt und im Falle einer Störung mit try-/catch-Methoden und Haltepunkten gefunden und beseitigt.

6.2 Fehlerbehebung

Der Autor schritt nach Prinzip des Wasserfall-Modells voran und erstellte Unit-Test's nach jedem Implementierungsschritt. Dabei hielt er sich nicht an die Standard-Regeln des Wasserfall-Modells für das gesamte Projekt, sondern wand es iterativ für jeden einzelnen Schritt an. Anhand dieser Vorgehensweise sind keine objektiven Fehler des Programms bekannt. Der Mobile Client konnte jedoch nicht auf verschiedenen Plattformen getestet werden und wird somit durchaus noch gewartet oder erweitert werden müssen.

Sollten auf Kundenseite in der Live Umgebung Fehler auftreten, werden diese nach auftreten schnell behoben, angepasst, bzw. gepatcht werden können, da ein funktionierender Ansatz bereits besteht und es sich lediglich um die Plattform handeln kann für die kleine Erweiterungen von Notwendigkeit sein können.

7 Fazit

7.1 Abnahme

Es erfolgte eine Präsentation an den Auftraggeber mit Vorschau der Funktionalität. Des Weiteren wurden zusätzliche Ziele zum Design und Möglichkeiten, wie das Anzeigen einer PDF durch das OCR oder gängigen DevExpress-Optionen gefordert. Diese Punkte gehören zu weiteren Maßnahmen am Mobilen Client und nicht zur Projektarbeit und werden daher nach Abschluss dieser besprochen und angegangen.

7.2 Soll-Ist-Vergleich

In der Funktion entspricht das Programm der Vorstellung des Auftraggebers, doch die Inhalte sind noch die falschen. Auf Grund eines internen Fehlerbildes, entschied der Autor für die Fertigstellung der Projektarbeit eine alternative API-Schnittstelle der [REDACTED] zu implementieren um die Funktionalitäten zu testen. Um die vorgegebenen 80 Stunden der IHK zur Projektarbeit nicht zu überziehen, wird erst zur Fertigstellung der Projektarbeit nach einer Lösung gesucht.

Fehlerbild des internen Fehlers:

Im Ist Zustand müssen Kund:innen für die Funktionalität des Programms die derzeit aktuelle Microsoft.WindowsAppSDK 1.3/1.5 verwenden, welche in den hiesigen erwerbbaaren Betriebssystemen gegeben ist. Sollten Kund:innen bspw. Eine aktuelle Preview-Version vom Windows Server 2022 verwenden, werden Fehler auftauchen, da eine erforderliche SDK nicht gegeben ist.

Zusammenfassend handelt es sich um eine temporäre Abweichung des Soll Zustandes, welches aber wegen des Benutzerhandbuches oder durch Servicegespräche eine untergeordnete Rolle spielt. Wird die SDK beim Windows 2022 Server nachgereicht, wird diese automatisch angesprochen.

In der Konklusion ist es eine Frage der Zeit bis der Soll Zustand ohne aktives Zutun erreicht wird.

7.3 Zukunftsorientierung

Der Mobile [REDACTED] Client ist sehr gut ausbaubar und auf Kund:innenwunsch anpassbar. Sowohl weitere Datenbanken anzubinden, als auch das Design zu ändern, erfordert nur die letzteren Schritte der Implementierungsphase.

Zugleich wird das Team der [REDACTED] immer einen Entwickler zur Wartung des Clients benötigen, da dieser mit dem Framework DevExpress arbeitet und auf externe [REDACTED] API-Schnittstellen zugreifen muss. Diesbezüglich muss immer darauf geachtet werden, welche Updates die Plattformen herausbringen und ob Patches des Clients zur Anpassung für weitere Funktion notwendig sind.

In grober Erwartung an die Standardisierung des Frameworks des DevEx und der API-Schnittstellen der [REDACTED], vermutet der Autor keinen großen Aufwand zur weiteren Wartung des Clients und schließt somit großen Zeit- und Kostenaufwand dafür aus.

Anlagen:

A. Anhangsverzeichnis

I. Pflichtenheft

Pflichtenheft:

Folgende Punkte sollen umgesetzt werden:

- ▶ Anmeldung und Authentifikation über den Mobilen Client an [REDACTED]
 - Weiterleitung aus dem Mobilen Client auf die [REDACTED] com Webseite zur OAuth 2.0 Authentifikation und Erhalt des Tokens nach der Anmeldung.
 - Speichern des Tokens im isolierten Speicher auf dem genutzten Gerät, sowie Sicherung der erhaltenen Anmeldedaten.
- ▶ Auswahl der Lizenz und Datenbanken
 - Auswahl der zu nutzenden Lizenz anhand der Authentifikation.
 - Auswahl der Datenbank/en anhand der gewählten Lizenz.
- ▶ Zugriff auf den Inhalt vom Posteingangsmanager (PEM)
 - Ansprache und Verbindung an die vorgegebene API-Schnittstelle mit dem dahinter liegendem Posteingangsmanager-Modul der [REDACTED].
 - Auswahl der anzuzeigenden Daten aus dem Inhalt der mit dem Posteingangsmanager-Modul verknüpften Datenbanken.
- ▶ Ausgabe der Inhalte vom Posteingangsmanager
 - Eine Visuelle Ausgabe der Inhalte vom Posteingangsmanager in dem Mobilen Client mit Status der Bearbeitung einzelner internen Schritte und künftiger Anzeige der PDF-Daten über das OCR. (Nach Abschließung der Projektarbeit)
- ▶ Bearbeitung bestimmter einzelner Inhaltspunkte des Posteingangsmanagers
 - Bestätigung der fertiggestellten Aufgabepunkte im Mobilen Client mit der CRUD-Update-Methode, dieser Information im Modul des Posteingangsmanagers.
- ▶ Nutzen vom Framework „DevExpress“ zur besseren Visualisierung des Mobilen Clients, (nach Abschluss der Projektarbeit)
 - Log-In-, Lizenz- und Datenbankauswahlfenster, sowie Anzeige des Postmanagers und die Ausgabe der Daten sollen vereinheitlicht und verkaufsorientiert designet werden

II. Verwendete NuGet-Pakete und Bibliotheken

II.1 Klassenbibliothek:

Integrative.CrossProtect	=>	Version 0.1.3
↳ Stellt die kryptografische Logik- und Schlüsselspeicherlösungen bereit.		
System.Text.Json	=>	Version 6.0.5
↳ Funktionen zur Verarbeitung und Serialisierung von JSON Objekten		
Microsoft.Extensions.Http	=>	Version 6.0.0
↳ Klasse zur Konfiguration der IHttpConnectionFactory-Schnittstelle		
IdentityModel.OidcClient	=>	Version 5.0.2
↳ Connect-Client-Bibliothek für native Anwendungen		
System.Net.Http.Json	=>	Version 6.0.0
↳ Erweiterungsmethoden zum Lesen und auswerten des HttpContent aus JSON		

II.2 .NET MAUI Anwendung:

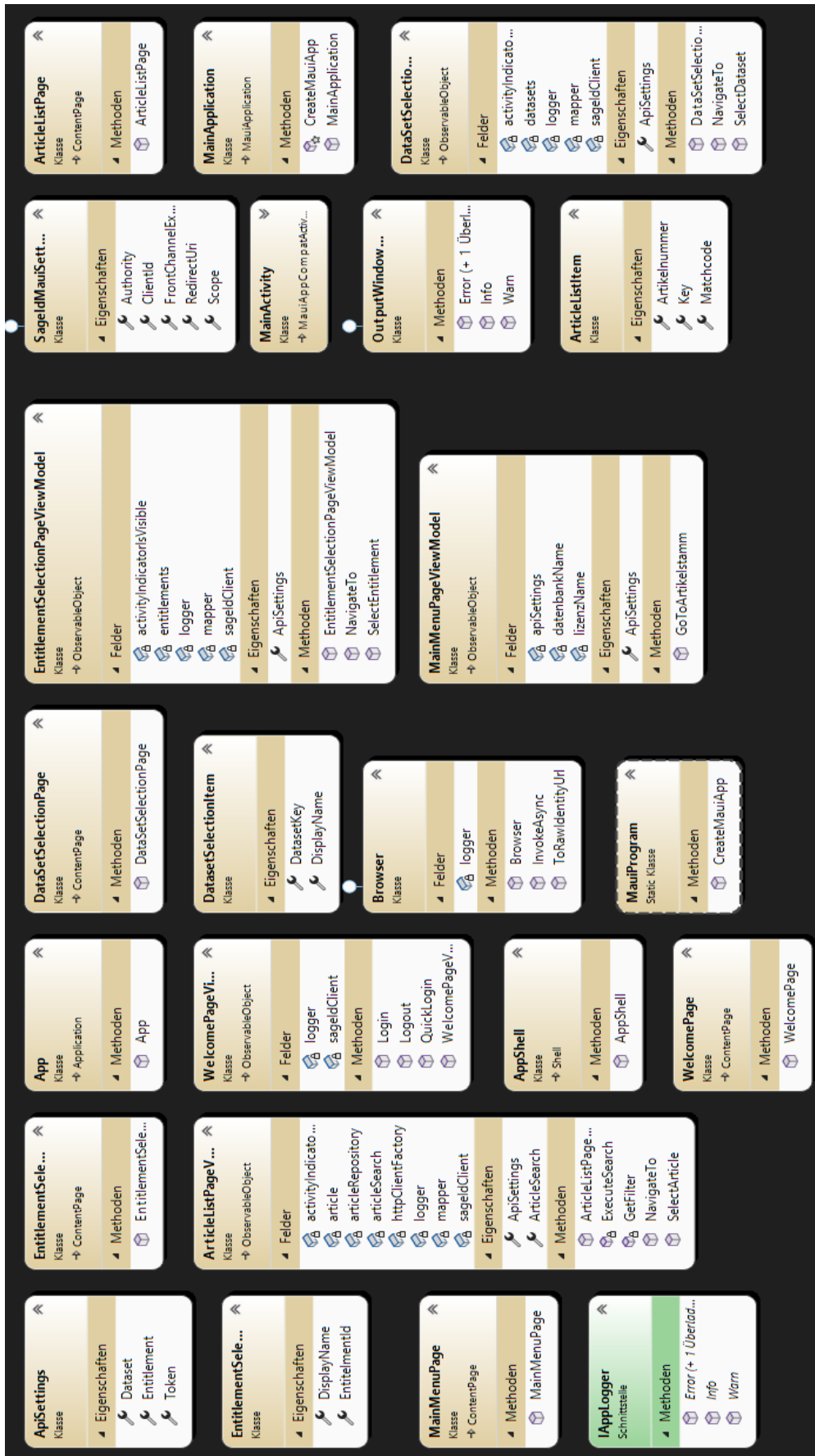
AutoMapper	=>	Version 11.0.1
↳ Ein konventionsbasierter Objekt-Objekt Mapper.		

AutoMapper ist auf Modellprojektionsszenarien ausgerichtet, um komplexe Objektmodelle auf DTOs und andere einfache Objekte zu reduzieren, deren Design besser für Serialisierung, Kommunikation, Messaging oder einfach als Antikorruptionsschicht zwischen der Domänen- und Anwendungsschicht geeignet ist.

CommunityToolkit.Maui	=>	Version 1.2.0
CommunityToolkit.Mvvm	=>	Version 8.0.0
IdentityModel.OidcClient	=>	Version 5.0.2
↳ Connect-Client-Bibliothek für native Anwendungen		
Integrative.CrossProtect	=>	Version 0.1.3
↳ Stellt die kryptografische Logik- und Schlüsselspeicherlösungen bereit.		
Microsoft.Extensions.Http	=>	Version 6.0.0
↳ Klasse zur Konfiguration der IHttpConnectionFactory-Schnittstelle		
System.Security.Cryptography.ProtectedData	=>	Version 6.0.0
↳ Stellt Methoden zum Ver- und Entschlüsseln von Daten bereit. Nicht vererbbar.		

III. Klassendiagramm

Klassendiagramm MAUI-Projekt



Klassendiagramm Klassenbibliothek:



IV. Screenshots zum Quellcode der Manuellen Tests

IV.1 => TokenService

Token speichern, erhalten und löschen.

```
[TestMethod()]
0 Verweise
public async Task SaveTokenAsyncTest()
{
    var tokenService = new TokenService();
    var token = new SageIdToken()
    {
        AccessToken = "AccessToken".ToSecureString(),
        RefreshToken = "RefreshToken".ToSecureString(),
        Expiry = DateTime.Now
    };
    var result = await tokenService.SaveTokenAsync(token);
    Assert.IsTrue(result);
}

[TestMethod()]
0 Verweise
public async Task DeleteTokenTest()
{
    var tokenService = new TokenService();
    tokenService.DeleteToken();
    var token = await tokenService.GetTokenAsync();
    Assert.IsNull(token);
}

[TestMethod()]
0 Verweise
public async Task GetTokenAsyncTest()
{
    var tokenService = new TokenService();
    var token = await tokenService.GetTokenAsync();
    Assert.IsNotNull(token);
}
```

Ein Test der alle drei Methoden beinhaltet.

```
[TestMethod()]
0 Verweise
public async Task SaveAndGetTokenAsync()
{
    var tokenService = new TokenService();
    tokenService.DeleteToken();
    var token = new SageIdToken()
    {
        AccessToken = "AccessToken".ToSecureString(),
        RefreshToken = "RefreshToken".ToSecureString(),
        Expiry = DateTime.Now
    };
    var result = await tokenService.SaveTokenAsync(token);
    Assert.IsTrue(result);
}
```

IV.2 => Lizenz und Datenbank Auswahl

```
[TestMethod()]
0 Verweise
public async Task GetEntitlementAsyncTest()
{
    var client = new SageIdClient(null, null, null, new TestHttpClientFactory());
    var token = new SageIdToken()
    {
        AccessToken = "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Iks56TXdPVVJHTVZNU5
        .ToSecureString()
    };
    var result = await client.GetEntitlementAsync(token);
    Assert.IsNotNull(result);
}

[TestMethod()]
0 Verweise
public async Task GetDataSetsAsyncTest()
{
    var client = new SageIdClient(null, null, null, new TestHttpClientFactory());
    var token = new SageIdToken()
    {
        AccessToken = "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Iks56TXdPVVJHTVZNU5
        .ToSecureString()
    };
    var result = await client.GetDatasetsAsync(token, null, null);

    Assert.IsNotNull(result);
}
```

V. Verwendete Ressourcen

V.1 Personal

- [REDACTED]
- Ein Angestellter der Hardwareabteilung zur Überwachung der Server-Umgebung

[REDACTED]

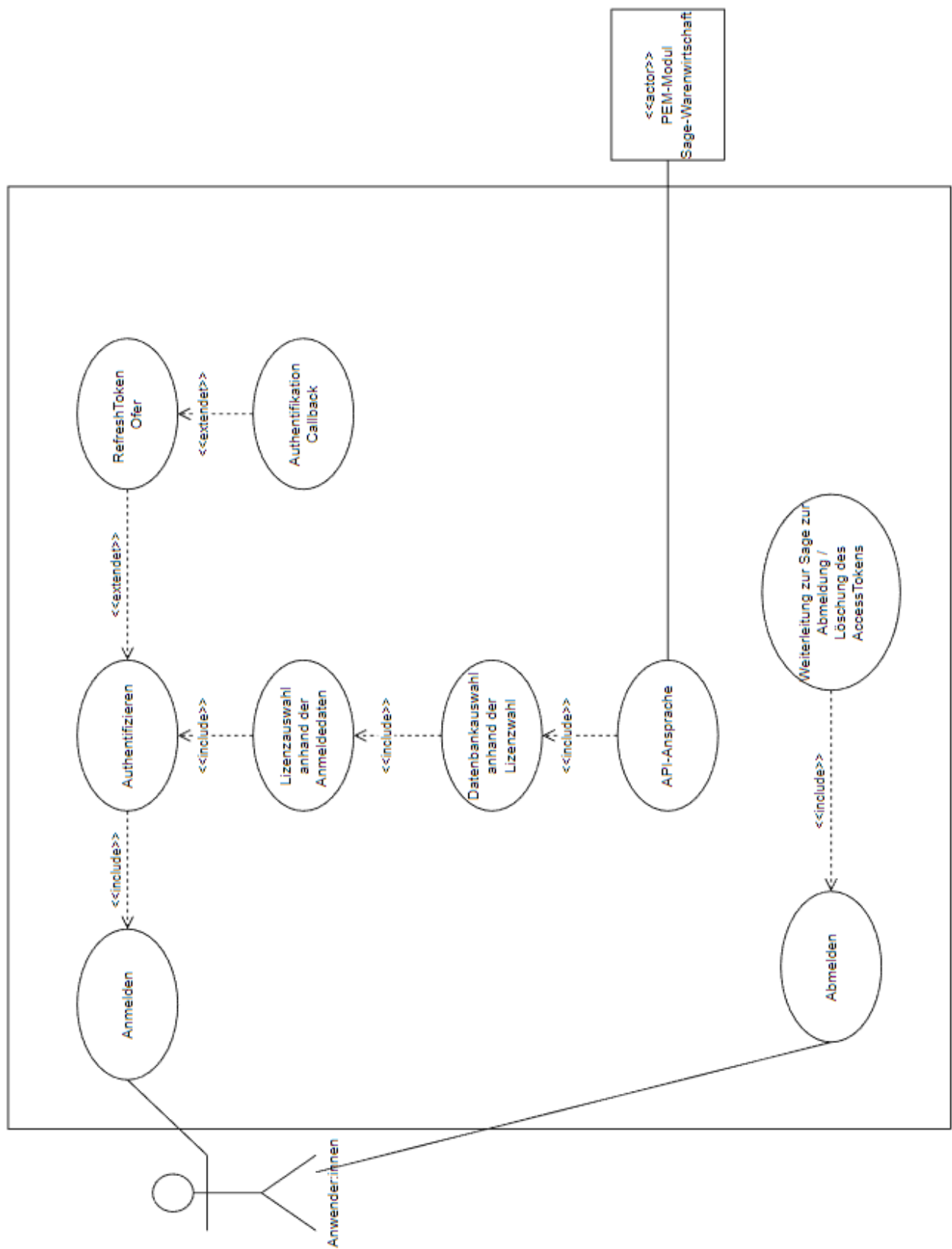
V.2 Software

- Visual Studio inklusive NuGet – Pakete und Zusatzbibliotheken,
- Violet UML-Editor,
- Postman zur Überprüfung der API-Abfragen,
- Internetbrowser für Dokumentationen,
- ein SQL-Server Management Studio mit einer Datenbank (Demo: Datenbank),
- Demo Datenbank mit der API-Schnittstelle vom „Post-Eingangs-Manager“,
- dass Sage 100 Softwarepaket,
- Sage Administrator Software,
- Sage Server Manager Software,
- Sage Warenwirtschaft zum Prüfen der Datenbankinhalte über das System,
- eine Sage 100 Partnerlizenz der AAIC Soft Systems GmbH
- und die für eine API-Authentifizierung notwendige Client ID der AAIC Soft Systems GmbH.

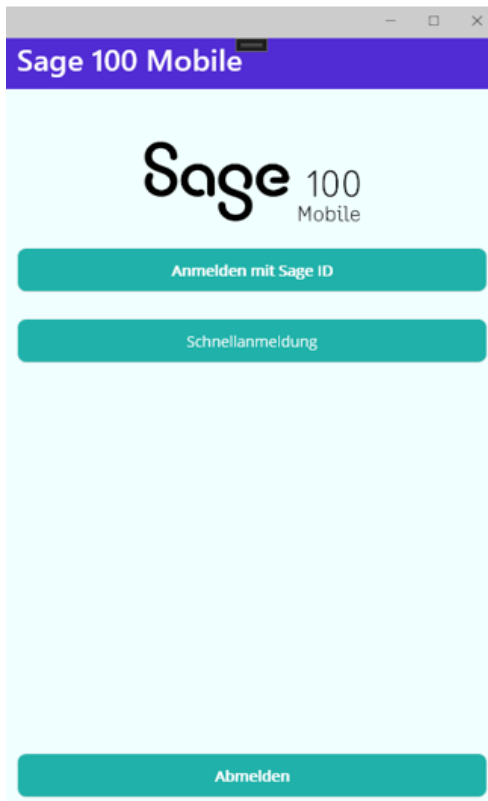
V.3 Hardware

- Arbeits-PC mit Zugriff auf einen Firmeninternen Test-Server
- Ein zweiter Monitor
- Internetanschluss

VI. Use-Case Diagramm



VII. Benutzerhandbuch



Anmeldedialog:

Mobiler Client der AAIC Soft Systems GmbH zum Ansprechen der Sage 100 API-Schnittstellen.

Am Titelschirm sind die Optionen zum An- und Abmelden zur Verfügung gestellt.

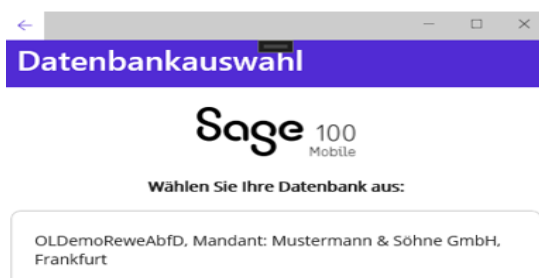
Durch die Auswahl einer dieser Optionen werden die Anwender:innen auf die Sage 100 Seite über ein Browserfenster geschickt und können sich so Authentifizieren oder Abmelden.



Lizenz- und Datenbankauswahl:

Nach einer erfolgreichen Authentifizierung auf der Sage 100 Web Seite, erhält der Mobile Client die notwendigen Informationen über die Anwender:innen und stellt somit die Auswahl der anhand von Anmeldedaten zur Verfügung stehender Lizenzen bereit.

Entsprechend der Auswahl der Lizenz wird auch die Wahl zur Datenbank zur weiteren Nutzung angeboten.



Hauptmenü

Sage 100 Mobile

- Artikelstamm
- Kundenstamm
- Lieferanten
- Posteingangsmanager

Hauptmenü:

Sobald die Auswahlen getroffen wurden eröffnet sich den Nutzer:innen das Hauptmenü. Hier wird darüber entschieden mit welchen Stammdaten gearbeitet werden möchte. In unserem Beispiel fällt die Auswahl auf die Artikelstammdaten.

Nach der Auswahl der Stammdaten spricht der Mobile Client die API-Schnittstelle der Sage 100 nach den Informationen zum Artikelstamm an und bildet die erhaltenen Informationen zur Nutzung ab.

- Im Hauptmenü immer mit abgebildet, die genutzte Lizenz und die gewählte Datenbank. Somit ist immer ein Überblick des Arbeitsumfeldes gegeben.

Mandant: OLDemoReweAbfD, Mandant: Mustermann & Söhne GmbH, Frankfurt
Lizenz: AAI C Soft Systems GmbH (1037903)

Inhaltsanzeige der Artikelstamm-Tabelle:

Artikel: Tischplatte

Grundlagen

Artikelnummer: 00001031

Bezeichnung 1: Tischplatte

Bezeichnung 2: Kiefernholz, unbeschicht

Basicmengeneinheit: qm

Artikelgruppe: 010

Hauptartikelgruppe: 010

Vaterartikelgruppe: EMPTY

Details

Hersteller: Holzbau AG

EAN-Nummer: 6576754363753

Verkauf

Verkauf: ☒

Preiseinheit-VK: 1

VK-Mengeneinheit: qm

Speichern

Artikelstamm

Artikel finden

Tischplatte	00100040	Sani-HDR-CX 200 Full HD Camera (Auslaufartikel)
T-Shirt (Variante)	00100041	
T-Shirt (Variante)	00100041	

Nun können Stammdaten gesucht und bearbeitet werden. Durch die Filterfunktion ist es möglich durch eine Sucheingabe schneller die gewünschten Artikel zu finden und mit der Auswahl eröffnet sich die Detailanzeige.

Hierbei werden die Informationen aus den Datenbanktabellen zum Artikel gezeigt mit der Möglichkeit diese zu Verändern und die Veränderung zu speichern.

VIII. Screenshots zum Quellcode

Klassenbibliothek – ohne Interfaces

SecureStringExtensions:

```
public static class SecureStringExtensions
{
    // Konvertiert eine normale Zeichenkette in eine "Nur-Lesen"-Zeichenkette (
    16 Verweise
    public static SecureString ToSecureString(this string source)           //
    {
        if (string.IsNullOrEmpty(source))                                   //
        {                                                                    //
            return null;                                                    //
        }                                                                    //
        var secureString = new SecureString();                             //
        foreach (var character in source)                                   //
        {                                                                    //
            secureString.AppendChar(character);                             //
        }                                                                    //
        secureString.MakeReadOnly();                                        //
        return secureString;                                                //
    }

    // Konvertiert aus einer "sicheren" Zeichentekke eine normale Zeichenkette
    9 Verweise
    public static string FromSecureString(this SecureString source)
    {
        if (source == null)                                                //
        {                                                                    //
            return null;                                                    //
        }                                                                    //
        var handle = IntPtr.Zero;                                           //
        try                                                                    //
        {                                                                    //
            handle = Marshal.SecureStringToGlobalAllocUnicode(source);      //
                                                                    //
            return Marshal.PtrToStringUni(handle);                          //
        }                                                                    //
        finally                                                                //
        {                                                                    //
            Marshal.ZeroFreeGlobalAllocUnicode(handle);                     //
        }                                                                    //
    }
}
```

SageIdClient:

```
public class SageIdClient : ISageIdClient
{
    private readonly ISageIdSettings sageIdSettings;
    private readonly IBrowser browser;
    private readonly ITokenService tokenService;
    private readonly IHttpClientFactory httpClientFactory;

    4 Verweise
    public SageIdClient(ISageIdSettings sageIdSettings,
        IBrowser browser, ITokenService tokenService,
        IHttpClientFactory httpClientFactory)
    {
        this.sageIdSettings = sageIdSettings;
        this.browser = browser;
        this.tokenService = tokenService;
        this.httpClientFactory = httpClientFactory;
    }

    // Es werden Methoden wie Login, RefreshLogin, Logout, E

    2 Verweise
    private OidcClientOptions GetOidcClientOptions()
    {
        return new OidcClientOptions
        {
            Authority = sageIdSettings.Authority,
            ClientId = sageIdSettings.ClientId,
            Scope = sageIdSettings.Scope,
            RedirectUri = sageIdSettings.RedirectUri,
            Browser = browser
        };
    }

    public async Task<SageIdRequestResult> LoginAsync()
    {
        // Erstellen einer Token Variablen
        var token = await tokenService.GetTokenAsync();
        // Abfrage ob ein Token vorhanden ist (not null von return)
        if (token != null)
        {
            // Ist der Token noch gültig? Sobald das Ablaufdatum größer ist als.UtcNow, kö
            if (token.Expiry > DateTime.UtcNow)
            {
                return new SageIdRequestResult()
                {
                    IsError = false, // ist Fehler vorhanden?
                    Error = "", // Fehler
                    Token = token // Token deklaration
                };
            }
            //Enthält unser Token einen RefreshToken? (RefreshToken not null)
            else if (token.RefreshToken != null)
            {
                SageIdRequestResult refreshResult = await RefreshAccessTokenAsync(token);
                if (!refreshResult?.IsError ?? false)
                    return refreshResult;
            }
        }

        var options = GetOidcClientOptions();
        var oidcClient = new OidcClient(options);
        var loginResult = await oidcClient.LoginAsync(new LoginRequest()
        {
            FrontChannelExtraParameters = sageIdSettings.FrontChannelExtraParameters
        });

        var sageIdResult = new SageIdRequestResult()
        {
            IsError = loginResult.IsError,
            Error = loginResult.Error,
            Token = new SageIdToken()
            {
                //Inhalt des SageIdToken
                AccessToken = loginResult.AccessToken.ToSecureString(),
                RefreshToken = loginResult.RefreshToken.ToSecureString(),
            }
        }
    }
}
```

```

        return SageIdResult; // gebe
    }

```

3 Verweise

```

public async Task<bool> LogoutAsync()
{
    var startUrl = new RequestUrl(sageIdSettings.Authority + // Erzeu
        "v2/logout").Create(new Parameters(new Dictionary
        {string, string>
        {
            {"client_id", sageIdSettings.ClientId},
            {"returnTo", sageIdSettings.RedirectUri}
        });

    var result = await browser.InvokeAsync(new BrowserOptions // Warte
        (startUrl, sageIdSettings.RedirectUri));
    if (result.ResultType == BrowserResultType.Success) // Wenn
        tokenService.DeleteToken(); // lösch

    return result.ResultType == BrowserResultType.Success; // Gebe
}

```

3 Verweise

```

public async Task<IEnumerable<Entitlement>> GetEntitlementAsync(
    ISageIdToken sageIdToken) // Nach der erstellung der Klassen Entitem

{
    var client = httpClientFactory.CreateClient("entitlementApi");
    client.DefaultRequestHeaders.Accept.Add(
        new MediaTypeWithQualityHeaderValue("application/json"));
    client.DefaultRequestHeaders.Authorization =

        new AuthenticationHeaderValue(
            OidcConstants.AuthenticationSchemes.AuthorizationHeaderBearer,
            sageIdToken.AccessToken.FromSecureString());

    var response = await client.GetAsync(
        "sdata/entitlementWS/entitlementContract10/-/entitlements");
    var response = await client.GetAsync(
        "sdata/entitlementWS/entitlementContract10/-/entitlements");

    if (!response.IsSuccessStatusCode)
    {
        /*Nur bis zurKonfiguration von HttpException benötigt um
        Fehler in Schnellansicht auszulesen und JSON zu erstellen */
        /*var error = await response.Content.ReadAsStringAsync();
        return null;*/
        throw new HttpException(response.StatusCode, await response.Content
            .ReadAsStringAsync());
    }
    var entitlements = await response.Content.ReadFromJsonAsync<Entitlements>();
    return entitlements.resources.
        Where(e => e.servicematchcode == "Connectivity_Sage_OL_DE");
}

```

3 Verweise

```

public async Task<DatasetResponse> GetDatasetsAsync(
    ISageIdToken sageIdToken, int entitlementid)

{
    var client = httpClientFactory.CreateClient("connectivityApi");
    client.DefaultRequestHeaders.Add("X-Sage-ConnectivityVersion", "1.3");

    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue(
        "application/json"));

    client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(
        OidcConstants.AuthenticationSchemes.AuthorizationHeaderBearer,
        sageIdToken.AccessToken.FromSecureString());

    var response = await client.GetAsync($"ws/{entitlementid}/sdata/ol/apiKunden.Sage.API");
    //var response = await client.GetAsync($"{entitlementid}/sdata/ol/apiKunden.Sage.API");

    if (!response.IsSuccessStatusCode)
    {

```

```

if (!response.IsSuccessStatusCode) // We
{
    //Nur bis zurKonfiguration von HttpException benötigt um Fehler in Schnellansicht auszule
    //var error = await response.Content.ReadAsStringAsync();
    //return null; // Di
    throw new HttpException(response.StatusCode, await response.Content.ReadAsStringAsync());
}
return await response.Content.ReadFromJsonAsync<DatasetResponse>(); // Er
}

```

SageIdToken:

```
namespace AAIC.API.CLIent.Sage100.Mobile.Lib.Token
```

```

{
    11 Verweise
    public class SageIdToken : ISageIdToken
    {
        16 Verweise
        public SecureString AccessToken { get; set; }
        12 Verweise
        public SecureString RefreshToken { get; set; }
        11 Verweise
        public DateTime Expiry { get; set; }
    }
}

```

TokenService:

```

8 Verweise
public class TokenService : ITokenService
{
    private readonly string tokenFile = "██████████";
    private readonly IsolatedStorageScope scope;

    5 Verweise
    public TokenService()
    {
        scope = IsolatedStorageScope.User |
                IsolatedStorageScope.Domain |
                IsolatedStorageScope.Assembly;
    }

    // Der TokenService bietet drei Methoden an Token speichern, Token löschen und Token

    5 Verweise
    public async Task<bool> SaveTokenAsync(ISageIdToken token)
    {
        using (var isoStore = IsolatedStorageFile.GetStore(scope, null, null))
        {
            // Objekt für die Serialisierung erstellen
            var insecureToken = new InsecureSageIdToken()
            {
                AccessToken = token.AccessToken.FromSecureString(),
                RefreshToken = token.RefreshToken.FromSecureString(),
                Expiry = token.Expiry.ToUniversalTime()
            };

            // Objekt serialisieren
            var json = JsonSerializer.Serialize(insecureToken);

            //JSON in verschlüsselte Bytes konvertieren
            var protectedJson = CrossProtect.Protect(
                Encoding.UTF8.GetBytes(json), null, DataProtectionScope.CurrentUser);
        }
    }
}

```

```

        // Verschlüsselte Daten in Base64 kodieren
        var protectedBase64Json = Convert.ToBase64String(protectedJson);

        //Base-64-Kodierten String abspeichern
        using (var isoStream = new IsolatedStorageFileStream(
            tokenFile, FileMode.Create, isoStore))
        {
            using (var writer = new StreamWriter(isoStream))
            {
                await writer.WriteAsync(protectedBase64Json);
            }
        }
        return true;
    }
}

5 Verweise
public void DeleteToken()
{
    using (var isoStore = IsolatedStorageFile.GetStore(scope, null, null))
    {
        isoStore.DeleteFile(tokenFile);
    }
}

5 Verweise
public async Task<ISageIdToken> GetTokenAsync()
{
    using (var isoStore = IsolatedStorageFile.GetStore(scope, null, null))
    {
        if (isoStore.FileExists(tokenFile))
        {
            using (var isoStream = new IsolatedStorageFileStream(
                tokenFile, FileMode.Open, isoStore))
            {
                using (StreamReader reader = new StreamReader(isoStream))
                {
                    //Daten auslesen
                    var protectedBase64Json = await reader.ReadToEndAsync();

                    //Bytes vom 64-Base-kodierter Zeichenkette auslesen
                    var protectedJson = Convert.FromBase64String(protectedBase64Json);

                    //Daten entschlüsseln
                    var json = CrossProtect.Unprotect(protectedJson, null,
                        DataProtectionScope.CurrentUser);

                    // In Objekt deserialisieren
                    var insecureToken = JsonSerializer.Deserialize<InsecureSageIdToken>(json);

                    //Sicheres Objekt zurückgeben
                    return new SageIdToken()
                    {
                        AccessToken = insecureToken.AccessToken.ToSecureString(),
                        RefreshToken = insecureToken.RefreshToken.ToSecureString(),
                        Expiry = insecureToken.Expiry
                    };
                }
            }
        }
    }
}

return null;

```


InsecureSageIdToken:

```
namespace AAIC.API.Client.Sage100.Mobile.Lib.Model
{
    4 Verweise
    public class InsecureSageIdToken
    {
        4 Verweise
        public string AccessToken { get; set; }
        4 Verweise
        public string RefreshToken { get; set; }
        4 Verweise
        public DateTime Expiry { get; set; }
    }
}
```

SageIdRequestResult:

```
namespace AAIC.API.Client.Sage100.Mobile.Lib.Model
{
    public class SageIdRequestResult
    {
        9 Verweise
        public bool IsError { get; set; }
        8 Verweise
        public string Error { get; set; }
        9 Verweise
        public ISageIdToken Token { get; set; }
    }
}
```

HttpExceptions:

```
namespace AAIC.API.Client.Sage100.Mobile.Lib.Exceptions
{
    19 Verweise
    public class HttpException : Exception
    {
        6 Verweise
        public HttpStatusCode StatusCode { get; private set; }
        8 Verweise
        public DiagnosList DiagnosList { get; private set; } // PRIVATE SET LESEN
        9 Verweise
        public HttpException (HttpStatusCode statusCode, string content) : base (content)
        {
            StatusCode = statusCode;
            if (string.IsNullOrWhiteSpace(content) || statusCode == HttpStatusCode.Unauthorized)
            {
                //Sonderfälle wie beispielsweise Statuscode 401
                var message = statusCode.ToString();
                if (!string.IsNullOrWhiteSpace(content)) // Wenn "content" nicht // Erstellt Variable "m
                    message = content;

                DiagnosList = new DiagnosList(); // Erstellt das Objekt
                DiagnosList.diagnoses = new Diagnoses[1]; // Inhalt der Diagnosen
                DiagnosList.diagnoses[0] = new Diagnoses() // Index "0" erhält ein
                {
                    message = "Fehler: " + message // Das Feld "Nachricht"
                };
            }
            else
                DiagnosList = JsonSerializer.Deserialize<DiagnosList>(content); // Im anderen F
        }
    }
}
```

Model:

DatasetItem:

```
public class DatasetItem
{
    [JsonPropertyName("$url")]
    0 Verweise
    public string url { get; set; }

    [JsonPropertyName("$key")]
    1 Verweis
    public string key { get; set; }

    [JsonPropertyName("$descriptor")]
    1 Verweis
    public string descriptor { get; set; }

    [JsonPropertyName("$updated")]
    0 Verweise
    public DateTime updated { get; set; }
}
```

DatasetResponse

```
public class DatasetResponse
{
    [JsonPropertyName("$url")]
    public string url { get; set; }

    [JsonPropertyName("$descriptor")]
    public string descriptor { get; set; }

    [JsonPropertyName("$updated")]
    0 Verweise
    public DateTime updated { get; set; }

    [JsonPropertyName("$resources")]
    2 Verweise
    public DatasetItem[] resources { get; set; }
}
```

Diagnoses:

```
public class Diagnoses
{
    // Für jede Property muss das jeweilige JSON-Attribut "JsonPropertyName"
    [JsonPropertyName("$severity")]
    0 Verweise
    public string Dseverity { set { severity = value; } }

    [JsonPropertyName("$sdataCode")]
    0 Verweise
    public string DsdataCode { set { sdataCode = value; } }

    [JsonPropertyName("$applicationCode")]
    0 Verweise
    public string DapplicationCode { set { applicationCode = value; } }

    [JsonPropertyName("$message")]
    0 Verweise
    public string Dmessage { set { message = value; } }

    [JsonPropertyName("$stackTrace")]
    0 Verweise
    public string DstackTrace { set { stackTrace = value; } }

    [JsonPropertyName("$payloadPath")]
    0 Verweise
    public string DpayloadPath { set { payloadPath = value; } }

    // Die Sage 100 liefert Fehlerobjekte ohne das
    // da JsonProperty nur einzelne Werte zulässt.
    [JsonPropertyName("severity")]
    1 Verweis
    public string severity { get; set; }

    [JsonPropertyName("sdataCode")]
    1 Verweis
    public string sdataCode { get; set; }

    [JsonPropertyName("applicationCode")]
    1 Verweis
    public string applicationCode { get; set; }

    [JsonPropertyName("message")]
    public string message { get; set; }

    [JsonPropertyName("stackTrace")]
    public string stackTrace { get; set; }

    [JsonPropertyName("payloadPath")]
    public string payloadPath { get; set; }
}
```

DiagnosList:

```
3 Verweise
public class DiagnosList
{
    // Für jede Property muss das jeweilige JSON
    [JsonPropertyName("$diagnoses")]
    6 Verweise
    public Diagnoses[] diagnoses { get; set; }
}
```

Entitlement:

```
4 Verweise
public class Entitlement // Umbenennen des Resou-
{
    1 Verweis
    public int entitlementid { get; set; }

    1 Verweis
    public string servicematchcode { get; set; }
    1 Verweis
    public string displayname { get; set; }
    0 Verweise
    public string accountid { get; set; }
    0 Verweise
    public string accountsourc { get; set; }
    0 Verweise
    public bool isdemo { get; set; }
}
```

Entitlements:

```
public class Entitlements // Umbenennen des Root
{
    public int totalentitlements { get; set; }
    [JsonPropertyName("$resources")]
    public Entitlement[] resources { get; set; }
}
```

MAUI-Projekt: ohne Interfaces

OutputWindowLogger:

```
public class OutputWindowLogger : IAppLogger
// Diese Klasse stellt Informationen zur Fehlern,Warnungen od
{
    public void Error(string message)
        => Debug.WriteLine("ERROR: " + message);

    public void Error(Exception exception)
        => Debug.WriteLine("ERROR: " + exception.ToString());

    public void Info(string message)
        => Debug.WriteLine("Info: " + message);

    public void Warn(string message)
        => Debug.WriteLine("Warn: " + message);
}
```

ApiSettings:

```
public class ApiSettings
{
    public ISageIdToken Token { get; internal set; }

    public EntitlementSelectedItem Entitlement { get; internal set; }

    public DatasetSelectedItem Dataset { get; internal set; }
}
```

ArticleListItem:

```
public class ArticleListItem
{
    public string Artikelnummer { get; init; }
    public string Matchcode { get; init; }
    public string Key { get; set; }
}
```

DatasetSelectionItem:

```
public class DatasetSelectionItem
{
    public string DisplayName { get; init; }
    public string DatasetKey { get; init; }
}
```

EntitlementSelectionItem:

```
public class EntitlementSelectionItem
{
    public string DisplayName { get; init; }
    public int EntitelmentId { get; init; }
}
```

Browser:

```
2 Verweise
internal class Browser : IdentityModel.OidcClient.Browser.IBrowser
{
    private readonly IAppLogger logger;
    0 Verweise
    public Browser(IAppLogger logger)
    {
        this.logger = logger;
    }

    0 Verweise
    public async Task<BrowserResult> InvokeAsync(BrowserOptions options, CancellationToken cancellationToken = default)
    {
        try
        {
            WebAuthenticatorResult authResult = null;

            authResult = await WinUIEx.WebAuthenticator.AuthenticateAsync(new Uri(options.StartUrl), // "file" zum Ausle:
                new Uri("myprot://myapp.com/callback"));

            authResult = await WebAuthenticator.AuthenticateAsync(new Uri(options.StartUrl), // Wo kommt die StartUrl
                new Uri("myprot://myapp.com/callback")); // Callback Uri ist ange
                // In SageMauiSettings i:

            /*
            var url = new Uri("https://id.sage.com/");
            var callbackUrl = new Uri("myprot://myapp.com/callback");

            authResult = await WebAuthenticator.AuthenticateAsync(new WebAuthenticatorOptions
            {
                Url = url,
                CallbackUrl = callbackUrl,
            });
            */

            var authorizeResponse = ToRawIdentityUrl(options.EndUrl, authResult);

            return new BrowserResult
            {
                Response = authorizeResponse
            };
        }
    }
}
```

SageIdMauiSettings:

```
public class SageIdMauiSettings : ISageIdSettings
{
    3 Verweise
    public string Authority => "[REDACTED]";
    //public string Authority => "[REDACTED]";

    3 Verweise
    public string ClientId => "[REDACTED]";

    3 Verweise
    public Parameters FrontChannelExtraParameters => new Parameters(new List<KeyValuePair<string, string>> { new KeyValuePair<string, string>("audience", "[REDACTED]") });

    4 Verweise
    public string RedirectUri => "myprot://myapp.com/callback";

    2 Verweise
    public string Scope => "openid token access_token offline:access email profile";
}
```

ArticleListPageViewModel:

```
[QueryProperty(nameof(ApiSettings), "ApiSettings")]
```

6 Verweise

```
public partial class ArticleListPageViewModel : ObservableObject  
{
```

```
    [QueryProperties]
```

```
    [MVVM]
```

```
    [Fields]
```

```
    [Dependency Injection]
```

```
    // Konstruktor
```

0 Verweise

```
    public ArticleListPageViewModel(ISageIdClient sageIdClient,  
        IAppLogger logger, IMapper mapper, IHttpApiClientFactory httpClientFactory)
```

```
    {  
        this.sageIdClient = sageIdClient;  
        this.logger = logger;  
        this.mapper = mapper;  
        this.httpClientFactory = httpClientFactory;  
        article = new();  
    }
```

2 Verweise

```
    private string GetFilter()
```

```
    {  
        if (string.IsNullOrEmpty(ArticleSearch))  
            return "";  
  
        var filter = "(Artikelnummer eq '$filter') or (Matchcode like '%$filter%')"  
            .Replace("$filter", ArticleSearch);  
        logger.Info("ArtikelstammPageViewModel/ExecuteSearch-Filter => " + filter);  
        return filter;  
    }
```

```
    [RelayCommand]
```

2 Verweise

```
    public async void NavigateTo()
```

```
    {  
        logger.Info("ArticleListPageViewModel/NavigateTo-Entry");  
        ActivityIndicatorIsVisible = true;  
        article.Clear();  
  
        if (this.articleRepository == null)  
            this.articleRepository = new ArticleRepository(sageIdClient,  
                ApiSettings.Entitlement.EntitlementId, ApiSettings.Dataset.DatasetKey, httpClientFactory);  
  
        try  
        {  
            var result = await articleRepository.GetListAsync(GetFilter());  
            if (result != null && result.resources != null)  
            {  
                foreach (var item in result.resources)  
                {  
                    article.Add(mapper.Map<ArticleListItem>(item));  
                }  
            }  
        }  
    }
```

```

    }
    catch (HttpException e)
    {
        await Toast.Make(e.DiagnosList.diagnoses.First().message).Show();
    }
    ActivityIndicatorIsVisible = false;
    logger.Info("ArticleListPageViewModel/NavigateTo-Exit");
}

1 Verweis
private async Task ExecuteSearch()
{
    logger.Info("ArticleListPageViewModel/ExecuteSearch-Entry");
    ActivityIndicatorIsVisible = true;
    article.Clear();

    try
    {
        var result = await articleRepository.GetListAsync(GetFilter());
        if (result != null && result.resources != null)
        {
            article.Clear();

            foreach (var item in result.resources)
            {
                logger.Info("Add item: " + item.Matchcode);
                article.Add(mapper.Map<ArticleListItem>(item));
            }
        }
    }
    catch (HttpException e)
    {
        await Toast.Make(e.DiagnosList.diagnoses.First().message).Show();
    }

    ActivityIndicatorIsVisible = false;
    logger.Info("ArticleListPageViewModel/ExecuteSearch-Exit");
}

[RelayCommand]
2 Verweise
public async Task SelectArticle(ArticleListItem item)
{
    await Shell.Current.GoToAsync("ArticleDetailPage",
        new Dictionary<string, object>()
        {
            { "ApiSettings", ApiSettings },
            { "ArticleListItem", item }
        });
}
}

```

DataSetSelectionPageViewModel:

```
[QueryProperty(nameof(ApiSettings), "ApiSettings")]
```

6 Verweise

```
public partial class DataSetSelectionPageViewModel : ObservableObject
{
```

```
    [QueryProperties]
```

```
    #region MVVM
```

```
    [ObservableProperty]
```

```
    ObservableCollection<DatasetSelectedItem> datasets;
```

```
    [ObservableProperty]
```

```
    bool activityIndicatorIsVisible;
```

```
    #endregion
```

```
    #region Dependency Injection
```

```
    private readonly ISageIdClient sageIdClient;
```

```
    private readonly IAppLogger logger;
```

```
    private readonly IMapper mapper;
```

```
    #endregion
```

```
    //Konstruktor
```

0 Verweise

```
    public DataSetSelectionPageViewModel (ISageIdClient sageIdClient,
        IAppLogger logger, IMapper mapper)
```

```
    {
        this.sageIdClient = sageIdClient;
        this.logger = logger;
        this.mapper = mapper;
        datasets = new();
    }
```

```
    #region Code
```

```
    [RelayCommand]
```

2 Verweise

```
    public async void NavigateTo()
```

```
    {
        logger.Info("DataSetSelectionPageViewModel/NavigatedTo-Entry");
        ActivityIndicatorIsVisible = true;
        datasets.Clear();

        try
        {
            var result = await sageIdClient.GetDatasetsAsync(ApiSettings.Token,
                ApiSettings.Entitlement.EntitlementId);
            if (result != null && result.resources != null)
            {
                foreach (var item in result.resources)
                {
                    datasets.Add(mapper.Map<DatasetSelectedItem>(item));
                }
            }
        }
        catch (HttpException e)
        {
            await Toast.Make(e.DiagnosList.diagnoses.First().message).Show();
        }
        ActivityIndicatorIsVisible = false;
        logger.Info("DataSetSelectionPageViewModel/NavigatedTo-Exit");
    }
```

```
    [RelayCommand]
```

2 Verweise

```
    public async void SelectDataset(DatasetSelectedItem item)
```

```
    {
        try
        {
            ApiSettings.Dataset = item;
            await Shell.Current.GoToAsync("MainMenuPage", new Dictionary<string, object>()
                { { "ApiSettings", ApiSettings } });
        }
        catch (Exception e)
        {
            logger.Error(e);
        }
    }
```


EntitlementSelectionPageViewModel:

```
[QueryProperty(nameof(ApiSettings), "ApiSettings" )]/* <= oder einen festen String wie : "A
6 Verweise
public partial class EntitlementSelectionPageViewModel : ObservableObject
{

    #region QueryProperties
    4 Verweise
    public ApiSettings ApiSettings { get; set; }
    #endregion

    #region MVVM
    [ObservableProperty]
    ObservableCollection<EntitlementSelectedItem> entitlements; |

    [ObservableProperty]
    bool activityIndicatorIsVisible;
    #endregion

    #region Dependency Injection
    private readonly ISageIdClient sageIdClient;
    private readonly IAppLogger logger;
    private readonly IMapper mapper;
    #endregion
    //Konstruktor
    0 Verweise
    public EntitlementSelectionPageViewModel(ISageIdClient sageIdClient, IAppLogger logger,
        IMapper mapper) //EntitlementSelectionPageViewModel viewModel
    {

        this.sageIdClient = sageIdClient;
        this.logger = logger;
        this.mapper = mapper;
        entitlements = new();
    }

    #region Code
    [RelayCommand]
    2 Verweise
    public async void NavigateTo()
    {
        logger.Info("EntitlementSelectionPageViewModel/NavigatedTo-Entry");
        ActivityIndicatorIsVisible = true;
        entitlements.Clear();
        try
        {
            var result = await sageIdClient.GetEntitlementAsync(ApiSettings.Token);
            if(result != null)
            {
                foreach(var item in result)
                {
                    entitlements.Add(mapper.Map<EntitlementSelectedItem>(item));
                }
            }
        }
        catch(HttpException e)
        {
            await Toast.Make(e.DiagnosList.diagnoses.First().message).Show();
        }
    }

    ActivityIndicatorIsVisible = false;
    logger.Info("EntitlementSelectionPageViewModel/NavigatedTo-Exit");
}

[RelayCommand]
2 Verweise
public async Task SelectEntitlement(EntitlementSelectedItem item)
{
    ApiSettings.Entitlement = item;
    await Shell.Current.GoToAsync("DataSetSelectionPage", new Dictionary<string, object>()
    { { "ApiSettings", ApiSettings } });
}
```

MainMenuPageViewModel:

```
namespace AAIC.API.Client.Sage100.Mobile.App.ViewModel
{
    [QueryProperty(nameof(ApiSettings), "ApiSettings")]
    4 Verweise
    public partial class MainMenuPageViewModel : ObservableObject
    {
        #region QueryProperties
        private ApiSettings apiSettings;
        4 Verweise
        public ApiSettings ApiSettings
        {
            get { return apiSettings; }
            set { apiSettings = value;
                LizenzName = ApiSettings.Entitlement.DisplayName;
                DatenbankName = ApiSettings.Dataset.DisplayName; }
        }
        #endregion

        #region MVVM
        [ObservableProperty]
        string datenbankName;
        [ObservableProperty]
        string lizenzName;
        #endregion

        #region Code
        [RelayCommand]
        2 Verweise
        public async Task GoToArtikelstamm()
        {
            await Shell.Current.GoToAsync("ArticleListPage", new Dictionary<string, object>()
            { { "ApiSettings", ApiSettings } });
        }

        #endregion
    }
}
```

WelcomePageViewModel:

```
//vmTemplateHeader Android                                using CommunityToolkit.Mvvm.ComponentModel;
6 Verweise
public partial class WelcomePageViewModel : ObservableObject
{
    //vmTemplateBody
    #region Dependency Injection
    private readonly ISageIdClient sageIdClient;
    private readonly IAppLogger logger;
    #endregion

    0 Verweise
    public WelcomePageViewModel(ISageIdClient sageIdClient, IAppLogger logger)
    {
        this.sageIdClient = sageIdClient;
        this.logger = logger;
    }
    #region Code
    [RelayCommand]
    2 Verweise
    public async Task Login()
    {
        var result = await sageIdClient.LoginAsync();
        if (result.IsError)
        {
            logger.Error(result.Error);
            //using CommunityToolkit.Maui.Alerts;
            await Toast.Make("Fehler bei der Anmeldung:" + Environment.NewLine + result.Error).Show();
        }
        else
        {
            var apiSettings = new ApiSettings
            {
                Token = result.Token
            };
            await Shell.Current.GoToAsync("EntitlementSelectionPage",
                new Dictionary<string, object>() { { "ApiSettings", apiSettings } });
        }
    }

    [RelayCommand]
    2 Verweise
    public async Task Logout()
    {
        await sageIdClient.LogoutAsync();
    }

    [RelayCommand]
    2 Verweise
    public async Task QuickLogin()
    {
        var result = await sageIdClient.LoginAsync();
        if (result.IsError)
        {
            logger.Error(result.Error);
            await Toast.Make("Fehler bei der Anmeldung:" + Environment.NewLine + result.Error).Show();
        }
    }
}
```

```

    }
    else
    {
        var api = new ApiSettings()
        {
            Token = result.Token,
            Entitlement = new EntitlementSelectedItem()
            {
                DisplayName = "AAIC Soft Systems (1037903)",
                EntitlementId = 1037903
            },
            Dataset = new DatasetSelectedItem()
            {
                DatasetKey = "OLDemoReweAbFD;123",
                DisplayName = "Demo, Mandant: Testmann & Söhne GmbH, Frankfurt"
            }
        };

        await Shell.Current.GoToAsync("MainMenuPage",
            new Dictionary<string, object>() { { "ApiSettings", api } });
    }
}
#endregion

```

MAUIProgram:

```
public static class MauiProgram
{
    4 Verweise
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseMauiApp<App>()
            .UseMauiCommunityToolkit()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
            });

        // AutoMapper
        var autoMapperConfig = new MapperConfiguration(c =>
        {
            c.CreateMap<Entitlement, Model.EntitlementSelectedItem>()
                .ForMember(dest => dest.EntitlementId, opt => opt.MapFrom(src => src.entitlementid))
                .ForMember(dest => dest.DisplayName, opt => opt.MapFrom(src => src.displayName));

            c.CreateMap<DatasetItem, Model.DatasetSelectedItem>()
                .ForMember(dest => dest.DatasetKey, opt => opt.MapFrom(src => src.key))
                .ForMember(dest => dest.DisplayName, opt => opt.MapFrom(src => src.descriptor));
        });

        builder.Services.AddSingleton(typeof(IMapper), autoMapperConfig.CreateMapper());

        // Dialogs
        builder.Services.AddSingleton<WelcomePage>();
        builder.Services.AddSingleton<WelcomePageViewModel>();
        builder.Services.AddTransient<EntitlementSelectionPage>();
        builder.Services.AddTransient<EntitlementSelectionPageViewModel>();
        builder.Services.AddTransient<DataSetSelectionPage>();
        builder.Services.AddTransient<DataSetSelectionPageViewModel>();
        builder.Services.AddTransient<MainMenuPage>();
        builder.Services.AddTransient<MainMenuPageViewModel>();
        builder.Services.AddTransient<ArticleListPage>();
        builder.Services.AddTransient<ArticleListPageViewModel>();

        // Logger
        builder.Services.AddSingleton<IAppLogger, OutputWindowLogger>();

        // SageId
        builder.Services.AddSingleton<ITokenService, TokenService>();
        builder.Services.AddSingleton<ISageIdToken, SageIdToken>();
        builder.Services.AddSingleton<ISageIdClient, SageIdClient>();
        builder.Services.AddSingleton<ISageIdSettings, OpenId.SageIdMauiSettings>();
        builder.Services.AddSingleton<IdentityModel.OidcClient.Browser.IBrowser, OpenId.Browser>();

        // SageId - HTTP-Clients
        builder.Services.AddHttpClient("entitlementApi",
            client => { client.BaseAddress = new Uri("https://entitlement.sage.de/"); });
        //client => { client.BaseAddress = new Uri("https://entitlement.sage.de/"); });
        builder.Services.AddHttpClient("connectivityApi",
            client => { client.BaseAddress = new Uri("https://connectivity.sage.de/"); });
        //client => { client.BaseAddress = new Uri("https://connectivity.sage.de/"); });

        // SageId - HTTP-Clients
        builder.Services.AddHttpClient("entitlementApi",
            client => { client.BaseAddress = new Uri("https://entitlement.sage.de/"); });
        //client => { client.BaseAddress = new Uri("https://entitlement.sage.de/"); });
        builder.Services.AddHttpClient("connectivityApi",
            client => { client.BaseAddress = new Uri("https://connectivity.sage.de/"); });
        //client => { client.BaseAddress = new Uri("https://connectivity.sage.de/"); });

        return builder.Build();
    }
}
```

ArticleListPage.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="AAIC.API.Client.Sage100.Mobile.App.View.ArticleListPage"
  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
  xmlns:vm="clr-namespace:AAIC.API.Client.Sage100.Mobile.App.ViewModel"
  Title="Artikelstamm">

  <ContentPage.Behaviors>
    <toolkit:EventToCommandBehavior EventName="NavigatedTo" Command="{Binding NavigateToCommand}" />
  </ContentPage.Behaviors>

  <Grid RowDefinitions="Auto,*" Padding="8" RowSpacing="10">
    <Border Grid.Row="0" StrokeShape="RoundRectangle 7">
      <Entry Margin="5,0" Placeholder="Artikel finden" VerticalTextAlignment="Center" Text="{Binding ArticleSearch}" />
    </Border>

    <ScrollView Grid.Row="1">
      <Grid RowDefinitions="Auto,Auto">

        <ActivityIndicator IsRunning="true" IsVisible="{Binding ActivityIndicatorIsVisible}" Grid.Row="0" />

        <CollectionView ItemsSource="{Binding Article}" Grid.Row="1">
          <CollectionView.ItemTemplate>
            <DataTemplate>
              <Frame Padding="8" Margin="0,2,0,0">
                <Frame.GestureRecognizers>
                  <TapGestureRecognizer Command="{Binding Path=SelectArticleCommand, Source={RelativeSource
                    AncestorType={x:Type vm:ArticleListPageViewModel}}}"
                    CommandParameter="{Binding .}" />
                </Frame.GestureRecognizers>

                <Grid RowDefinitions="25,25">
                  <Label Grid.Row="0" Text="{Binding Artikelnummer}" MaxLines="2" />
                  <Label Grid.Row="1" Text="{Binding Matchcode}" MaxLines="2" />
                </Grid>
              </Frame>
            </DataTemplate>
          </CollectionView.ItemTemplate>
        </CollectionView>
      </Grid>
    </ScrollView>
  </Grid>
</ContentPage>
```

DataSetSelectionPage.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"

    Title="Datenbankauswahl"
    x:Class="AAIC.API.Client.Sage100.Mobile.App.View.DataSetSelectionPage"
    xmlns:vm="clr-namespace:AAIC.API.Client.Sage100.Mobile.App.ViewModel" >
    <ContentPage.Behaviors>
        <toolkit:EventToCommandBehavior EventName="NavigatedTo" Command="{Binding NavigateToCommand}" />
    </ContentPage.Behaviors>
    <VerticalStackLayout>
        <Image Source="sage100mobile_black.png"
            HeightRequest="50" Margin="0,20,0,20" />
        <Label Text="Wählen Sie Ihre Datenbank aus:" FontAttributes="Bold"
            VerticalOptions="Center" HorizontalOptions="Center" Margin="0,0,0,10" />
        <ActivityIndicator IsRunning="True"
            IsVisible="{Binding ActivityIndicatorIsVisible}" />
        <CollectionView ItemsSource="{Binding Datasets}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Grid Padding="5,5">
                        <Frame>
                            <Frame.GestureRecognizers>
                                <TapGestureRecognizer Command="{Binding Path=SelectDatasetCommand,
                                    Source={RelativeSource AncestorType={x:Type
                                        vm:DataSetSelectionPageViewModel}}}"
                                    CommandParameter="{Binding .}" />
                            </Frame.GestureRecognizers>
                            <Label Text="{Binding DisplayName}" MaxLines="2" />
                        </Frame>
                    </Grid>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </VerticalStackLayout>
</ContentPage>
```

EntitlementSelectionPage.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AAIC.API.Client.Sage100.Mobile.App.View.EntitlementSelectionPage"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"

    Title="Lizenzauswahl"
    xmlns:vm="clr-namespace:AAIC.API.Client.Sage100.Mobile.App.ViewModel">
    <ContentPage.Behaviors>
        <toolkit:EventToCommandBehavior EventName="NavigatedTo" Command="{Binding NavigateToCommand}" />
    </ContentPage.Behaviors>

    <VerticalStackLayout>
        <Image Source="sage100mobile_black.png" HeightRequest="50" Margin="0,20,0,20" />
        <Label Text="Wählen Sie Ihre Lizenz aus:" FontAttributes="Bold" VerticalOptions="Center"
            HorizontalOptions="Center" Margin="0,0,0,10" />

        <ActivityIndicator IsRunning="True" IsVisible="{Binding ActivityIndicatorIsVisible}" />

        <CollectionView ItemsSource="{Binding Entitlements}" VerticalOptions="Center">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Grid Padding="2,5">
                        <Frame>
                            <Frame.GestureRecognizers>
                                <TapGestureRecognizer Command="{Binding Path=SelectEntitlementCommand,
                                    Source={RelativeSource AncestorType={x:Type vm:EntitlementSelectionPageViewModel}}}"
                                    CommandParameter="{Binding .}" />
                            </Frame.GestureRecognizers>
                            <Label Text="{Binding DisplayName}" MaxLines="2" />
                        </Frame>
                    </Grid>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </VerticalStackLayout>
</ContentPage>
```


MainMenu.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AAIC.API.Client.Sage100.Mobile.App.View.MainMenuPage"
    Title="Hauptmenü">

    <Grid RowDefinitions="Auto,*,Auto" Padding="5" RowSpacing="10">
        <Image Source="sage100mobile_black.png" HeightRequest="50" Margin="0,20,0,20" />

        <ScrollView Grid.Row="1">
            <VerticalStackLayout Spacing="5">
                <Button Text="Artikelstamm" Command="{Binding GoToArtikelstammCommand}" ImageSource="booktag30.png" Margin="30,0,30,5" />

                <Button Text="Kundenstamm" Command="{Binding GoToArtikelstammCommand}" ImageSource="person30.png" Margin="30,0,30,5"/>
                <Button Text="Lieferanten" Command="{Binding GoToArtikelstammCommand}" ImageSource="person30.png" Margin="30,0,30,5"/>
                <Button Text="Posteingangsmanager" Command="{Binding GoToPosteingangsmanagerCommand}" ImageSource="person30.png" Margin="30,0,30,5" />
            </VerticalStackLayout>
        </ScrollView>

        <VerticalStackLayout Grid.Row="2" HorizontalOptions="Center">
            <Grid RowDefinitions="Auto, Auto" ColumnDefinitions="Auto,Auto" ColumnSpacing="5">
                <Label Grid.Row="0" Grid.Column="0" Text="Mandant: " FontSize="12" HorizontalOptions="Start" />
                <Label Grid.Row="1" Grid.Column="0" Text="Lizenz: " FontSize="12" HorizontalOptions="Start" />
                <Label Grid.Row="0" Grid.Column="1" Text="{Binding DatenbankName}" FontSize="12" HorizontalOptions="Start" />
                <Label Grid.Row="1" Grid.Column="1" Text="{Binding LizenzName}" FontSize="12" HorizontalOptions="Start" />
            </Grid>
        </VerticalStackLayout>
    </Grid>
</ContentPage>
```

WelcomePage.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="AAIC.API.Client.Sage100.Mobile.App.View.WelcomePage"

    xmlns:viewmodel="clr-namespace:AAIC.API.Client.Sage100.Mobile.App.ViewModel"
    x:DataType="viewmodel:WelcomePageViewModel"
    Title="Sage 100 Mobile"
    BackgroundColor="Azure" >

    <Grid RowDefinitions="15,75,Auto,*,Auto" RowSpacing="25" Margin="10" Background="azure" >
        <Image Source="sage100mobile_black.png" Grid.Row="1" />

        <Button Text="Anmelden mit Sage ID" Command="{Binding LoginCommand}" Grid.Row="2" Background="LightSeaGreen" FontAttributes="Bold" />
        <Button Text="Schnellanmeldung" Command="{Binding QuickLoginCommand}" Grid.Row="3" Background="LightSeaGreen" />
        <Button Text="Abmelden" Command="{Binding LogoutCommand}" Grid.Row="5" Background="LightSeaGreen" FontAttributes="Bold" />
    </Grid>
</ContentPage>
```

App.xaml.cs:

```
namespace AAIC.API.Client.Sage100.Mobile.App
{
    5 Verweise
    public partial class App : Application
    {
        0 Verweise
        public App()
        {
            InitializeComponent();

            AppDomain.CurrentDomain.FirstChanceException += (sender, e) =>
                Debug.WriteLine("FirstChanceException: " + e.Exception.ToString());

            AppDomain.CurrentDomain.UnhandledException += (sender, e) =>
                Debug.WriteLine("UnhandledException: " + e.ExceptionObject.ToString());

            MainPage = new AppShell();
        }
    }
}
```


App.xaml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:AAIC.API.Client.Sage100.Mobile.App"
    x:Class="AAIC.API.Client.Sage100.Mobile.App.App">

    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Resources/Styles/Colors.xaml" />
                <ResourceDictionary Source="Resources/Styles/Styles.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

AppShell.caml.cs:

```
namespace AAIC.API.Client.Sage100.Mobile.App
{
    5 Verweise
    public partial class AppShell : Shell
    {
        1 Verweis
        public AppShell()
        {
            InitializeComponent();
            //Eine Route besteht immer aus einem Pfad(String) und dem Typen der Klasse, die aufgerufen wird
            Routing.RegisterRoute(nameof(EntitlementSelectionPage), typeof(EntitlementSelectionPage));

            Routing.RegisterRoute(nameof(DataSetSelectionPage), typeof(DataSetSelectionPage));

            Routing.RegisterRoute(nameof(MainMenuPage), typeof(MainMenuPage));

            Routing.RegisterRoute(nameof(ArticleListPage), typeof(ArticleListPage));
        }
    }
}
```

AppShell.xaml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell
    x:Class="AAIC.API.Client.Sage100.Mobile.App.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:AAIC.API.Client.Sage100.Mobile.App.View"
    Shell.FlyoutBehavior="Disabled">

    <ShellContent
        Title="Home"
        ContentTemplate="{DataTemplate local:WelcomePage}"
        Route="MainPage" />

</Shell>
```

B Quellenangaben

Webauthentifikator: Fehlerbildbehebung:

Microsoft: <https://learn.microsoft.com/de-de/dotnet/maui/platform-integration/communication/authentication?tabs=windows>

Authentifikationslösung:

GitHub: <https://github.com/dotMorten/WinUIEx/blob/main/src/WinUIEx/WebAuthenticator.cs>

Verwenden von Webauthenticator:

<https://learn.microsoft.com/de-de/xamarin/essentials/web-authenticator?tabs=android#using-webauthenticator>

Sage 100 API-Schnittstellen:

Swagger: <https://openapi.sage100.de/apiArtikel.Sage.API.html>