

---

# DIPLOMARBEIT

Gesamtprojekt



## ERP-Connect

**Erstellung einer Android-App**

Finn Dorninger      5BHIT

Betreuer: Gaisberger Alois, Prof. Dipl.-Ing.

**Projektpartner**

Softwareanbieter SYScO EDV. Technologiepark 3, 4311 Schwertberg

Auftraggeber: Peter Wurm, Geschäftsführer

Schuljahr 2019/20

Abgabevermerk:

Datum: 30.03.2020

## **Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Traun, am 27.03.2020

Verfasser:

---

Finn Dorninger

**DIPLOMARBEIT**

## DOKUMENTATION

Namen der Verfasser*innen	Finn Dorninger
Jahrgang Schuljahr	5BHT 2019/20
Thema der Diplomarbeit	ERP-Connect
Kooperationspartner	SYSco EDV, Peter Wurm

Aufgabenstellung	Für das Unternehmen „SYSco EDV“ soll eine Android App entwickelt werden. Die Anwendung soll, aus den von Unternehmen eingesetzten „Win-Line“ ERP-Systemen, Kunden- und zugehörige Ansprechpartnerdaten beziehen und übersichtlich visualisieren. Außerdem muss die Sicherheit der zu verarbeitenden Daten, vor allem aus datenschutzrechtlichen Gründen, gewährleistet werden. Um die Informationsmöglichkeit, unabhängig von der Internetverbindung, zu ermöglichen, soll eine Offline-Fähigkeit implementiert werden. Mit innovativen Funktionen der Anwendung soll die Kommunikation erleichtert und damit auch die Kundenbindung zwischen Mitarbeiter und Kunde verbessert werden.
------------------	--

Realisierung	Für die Programmierung einer nativen Android App wurde die Entwicklungsumgebung Android Studio und das Android-SDK verwendet.  Als Programmiersprache wurde entschieden Kotlin zu verwenden, da dessen Eigenschaften die Android Programmierung erleichtern.  Für die Daten-Speicherung und die Layout-Definition wurde die Sprache XML verwendet.
--------------	--

Ergebnisse	Das Ergebnis ist eine Android App, welche die Kunden oder Ansprechpartner übersichtlich in einer Listenansicht darstellt. Durch die Auswahl eines Listeneintrages, erscheint eine detaillierte Ansicht des ausgewählten Kunden oder Ansprechpartners.
------------	---

Typische Grafik, Foto etc. (mit Erläuterung)

Screenshot der Listenansicht:

ERP_Connect	
:	
🔍	
Aktiv Sport GmbH & Co KG	330050
Alexandra Fetz	230F004
Allsport GmbH	330001
Ammansberger	230A002
Andrew Mc Gyver	230MC2
Annas Sportwelt	230A001
Anton Müller & Co	24001
Auermann Sport	230A003
<span>📞</span> <span>💬</span> <span>📍</span> <span>📤</span>	
Konten	Kontakte

Screenshot der Detailansicht inklusive Funktionen:

ERP_Connect	
Annas Sportwelt (230A001)	
IHRE ANSPRECHPARTNER	
📞	💬
📍	📤
<u>ADRESSE</u>	
Staat	A
PLZ	4950
Ort	Altheim
Straße	Linzer Str. 12
<u>KONTAKT</u>	
Mail	anna@sportwelt.at
Homepage	
<u>TELEFON</u>	
Land-Vorwahl	+43
Ort-Vorwahl	1
Nummer	97030
<u>NOTIZ</u>	
Annas Sportwelt ist besonders auf den Mountainbike Sektor spezialisiert.	
Dr. Seidelbast ist selbst ein begeisteter	

Teilnahme an Wettbewerben, Auszeichnungen

–

Möglichkeiten der Einsichtnahme in die Arbeit	Der Sourcecode der Anwendung steht im Schularchiv zur Verfügung. Auch ist es möglich, den Auftragsgeber sowie den Autor dieser Arbeit zu kontaktieren.
---	--

Approbation (Datum/Unterschrift)	Prüfer*in  Prof. Alois Gaisberger	Direktor*in / Abteilungsvorstand*in  Dir. Doris Riha
-------------------------------------	---	--

**DIPLOMA THESIS**

## DOCUMENTATION

Author(s)	Finn Dorninger
Form	5BHIT
Academic year	2019/20
Topic	ERP-Connect
Co-operation partners	SYSco EDV, Peter Wurm

Assignment of tasks	An Android app is to be developed for the company "SYSco EDV". The application will retrieve "customer"- and related "contact person"-data from "WinLine" ERP systems that are used by companies and will visualize them in a clear and concise manner. In addition, the security of the data to be processed must be guaranteed, especially for data protection reasons. In order to provide the possibility of information access independent of the Internet connection, an offline availability is to be implemented. Innovative functions of the application should facilitate communication and thereby strengthens customer loyalty between the employee and customers.
---------------------	--

Realization	The development environment Android Studio and the Android SDK were used for programming a native Android app.  Kotlin was chosen as the programming language, because its features facilitate Android programming.  For the storage of data and the layout definition, the language XML was used.
-------------	--

Results	The result is an Android app that displays the customers or contact persons in a list view. By selecting an entry in the list, a detailed view of the selected customer or contact person appears.
---------	--

Illustrative graph, photo (incl. explanation)	Screenshot of the list view: <table border="1"> <tbody> <tr><td>Aktiv Sport GmbH &amp; Co KG</td><td>330050</td></tr> <tr><td>Alexandra Fetz</td><td>230F004</td></tr> <tr><td>Allsport GmbH</td><td>330001</td></tr> <tr><td>Ammansberger</td><td>230A002</td></tr> <tr><td>Andrew Mc Gyver</td><td>230MC2</td></tr> <tr><td>Annas Sportwelt</td><td>230A001</td></tr> <tr><td>Anton Müller &amp; Co</td><td>24001</td></tr> <tr><td>Auermann Sport</td><td>230A003</td></tr> </tbody> </table> Screenshot of the detailed view including functions: <p>Annas Sportwelt (230A001)</p> <p>IHRE ANSPRECHPARTNER</p> <p>ADRESSE</p> <table> <tbody> <tr><td>Staat</td><td>A</td></tr> <tr><td>PLZ</td><td>4950</td></tr> <tr><td>Ort</td><td>Altheim</td></tr> <tr><td>Straße</td><td>Linzer Str. 12</td></tr> </tbody> </table> <p>KONTAKT</p> <table> <tbody> <tr><td>Mail</td><td>anna@sportwelt.at</td></tr> <tr><td>Homepage</td><td></td></tr> </tbody> </table> <p>TELEFON</p> <table> <tbody> <tr><td>Land-Vorwahl</td><td>+43</td></tr> <tr><td>Ort-Vorwahl</td><td>1</td></tr> <tr><td>Nummer</td><td>97030</td></tr> </tbody> </table> <p>NOTIZ</p> <p>Annas Sportwelt ist besonders auf den Mountainbike Sektor spezialisiert. Dr. Seidelbast ist selbst ein begeisterter</p>	Aktiv Sport GmbH & Co KG	330050	Alexandra Fetz	230F004	Allsport GmbH	330001	Ammansberger	230A002	Andrew Mc Gyver	230MC2	Annas Sportwelt	230A001	Anton Müller & Co	24001	Auermann Sport	230A003	Staat	A	PLZ	4950	Ort	Altheim	Straße	Linzer Str. 12	Mail	anna@sportwelt.at	Homepage		Land-Vorwahl	+43	Ort-Vorwahl	1	Nummer	97030
Aktiv Sport GmbH & Co KG	330050																																		
Alexandra Fetz	230F004																																		
Allsport GmbH	330001																																		
Ammansberger	230A002																																		
Andrew Mc Gyver	230MC2																																		
Annas Sportwelt	230A001																																		
Anton Müller & Co	24001																																		
Auermann Sport	230A003																																		
Staat	A																																		
PLZ	4950																																		
Ort	Altheim																																		
Straße	Linzer Str. 12																																		
Mail	anna@sportwelt.at																																		
Homepage																																			
Land-Vorwahl	+43																																		
Ort-Vorwahl	1																																		
Nummer	97030																																		
Participation in competitions, awards	–																																		

Accessibility of Thesis	The source code of the application is available in the school archive. It is also possible to contact the constituent and the author of this work.	
-------------------------	--	--

Approval (Date/Signature)	Examiner  Prof. Alois Gaisberger	Head of College / Department  Dir. Doris Riha
------------------------------	--	---

# Inhaltsverzeichnis

<b>1 Einleitung .....</b>	<b>4</b>
<b>2 Zielsetzung und Aufgabenstellung.....</b>	<b>5</b>
2.1 Gesamtprojekt .....	5
2.1.1 Thema und Aufgabenstellung.....	5
2.1.2 Ausgangslage und Zielsetzung.....	5
<b>3 Grundlagen und Methoden .....</b>	<b>6</b>
3.1 Konzept für die Programmierung.....	6
3.1.1 Programmierung mit Java .....	6
3.1.2 Programmierung mit Kotlin .....	6
3.1.3 Entscheidung .....	6
3.2 Konzept für die Speicherung der Daten .....	7
3.2.1 Speicherung als XML-Dokument.....	7
3.2.2 Speicherung in einer relationalen Datenbank .....	8
3.2.3 Speicherung in einer objektorientierten Datenbank .....	8
3.2.4 Speichern von Daten in SharedPreferences.....	8
3.2.5 Entscheidung .....	8
3.3 Gewährleistung der Sicherheit .....	9
3.3.1 Sichere Webservice-Kommunikation .....	9
3.3.2 Verschlüsselung der lokalen Daten.....	9
3.3.3 Analyse der Sicherheit der Daten unter Android.....	10
<b>4 ERP-Connect .....</b>	<b>11</b>
4.1 Startfenster und Kundenlistenansicht .....	11
4.2 Kundendetailansicht .....	13
4.3 Darstellung der Ansprechpartner.....	14
4.4 Einstellungsmenü für die Verbindungseigenschaften.....	15
<b>5 Technische Dokumentation.....</b>	<b>16</b>
5.1 Erstellung einer Android App.....	16
5.1.1 Architekturmuster.....	16
5.1.2 Activities .....	17
5.1.3 Intents .....	18
5.1.4 Layout .....	19
5.1.5 Speichern und Laden von Einstellungen .....	21
5.1.6 Testen der Anwendung .....	22
5.2 Webservice-Zugriff.....	24
5.2.1 SimpleXML-Converter .....	24
5.2.2 Erstellung von Entitäten .....	24
5.2.3 Zugriff mit Retrofit.....	27
5.3 Persistenz der Daten .....	29
5.3.1 Speichern von Objekten in einer XML-Datei.....	29
5.3.2 Laden von Elementen aus einer XML-Datei.....	31
5.4 Gewährleistung der Sicherheit .....	32
5.4.1 Verschlüsselung der lokalen Daten.....	32
5.4.2 Sicherer Webservice-Zugriff .....	34
<b>6 Zusammenfassung .....</b>	<b>35</b>
6.1 Resultat.....	35

6.1.1 Errungenschaften.....	35
6.1.2 Herausforderungen .....	35
6.2 Ausblick.....	36
<b>7 Quellen- / Literaturverzeichnis .....</b>	<b>37</b>
<b>8 Abbildungsverzeichnis .....</b>	<b>38</b>
<b>9 Begleitprotokolle.....</b>	<b>39</b>
<b>10.....</b>	<b>47</b>
<b>11 Projektdokumente .....</b>	<b>47</b>

Kapitel	Verfasser*in
1-10	Dorninger

## **Danksagung**

An dieser Stelle möchte ich mich bei allen Personen bedanken, welche mich bei der Erstellung dieser Arbeit unterstützt haben.

Vorerst möchte ich mich bei meinem Betreuungslehrer Alois Gaisberger bedanken. Seine hartnäckigen Verbesserungsvorschläge, sowohl bei der schriftlichen Diplomarbeit als auch bei den Präsentationen, förderte die Qualität dieser Arbeit. Auch seine Berufserfahrung, kombiniert mit einem offenen Ohr, halfen mir immer, wenn ich den roten Faden verloren habe.

Auch vielen Dank an den Auftragsgeber, Peter Wurm. Jener gab mir in Form eines Praktikums die Möglichkeit, mich auf diese Arbeit vorzubereiten. Auch half er mir dabei, einen Überblick über das Projekt zu bekommen.

Zu guter Letzt sei auch meiner Familie gedankt, welche mich vor allem durch Aufmunterung und dem Korrekturlesen enorm unterstützte.

# 1 Einleitung

Aufgrund meines Interesses an der Android-Entwicklung, wurde mit der Unterstützung meines Betreuers ein passender Auftragsgeber mit einem spannenden Projekt gefunden. Der Auftragsgeber SYScor EDV ist ein Softwareanbieter aus der Region Schwerberg in Oberösterreich. Das Projekt ERP-Connect, eine Android-Anwendung, bietet Mitarbeitern eines Unternehmens eine Übersicht über deren Kunden sowie den zugehörigen Ansprechpartnern. Diese Daten, welche in einem ERP-System gespeichert sind, werden über einen Webservice für die Anwendung bereitgestellt. Die Anwendung ermöglicht den Mitarbeitern eine umfangreiche Informationsmöglichkeit und erleichtert die Kommunikation mit praktischen Funktionen. Durch die lokale Speicherung der Daten erzielt die Anwendung den Nutzen, auch an Orten ohne Datenverbindung verwendbar zu sein. Eine Verschlüsselung der lokalen Daten und eine sichere Webservice-Anwendung-Kommunikation dienen zur Gewährleistung der Sicherheit von sensiblen Kundendaten.

Für die Entwicklung wurde die Programmiersprache Kotlin und das Android-SDK im Selbststudium beigebracht. Kotlin bietet die Möglichkeit sicher und effizient zu programmieren.

Beim Verfassen dieser Diplomarbeit wurde auf das Gendern gänzlich verzichtet, um die Lesbarkeit zu erhöhen.

## **2 Zielsetzung und Aufgabenstellung**

### **2.1 Gesamtprojekt**

#### **2.1.1 Thema und Aufgabenstellung**

Der Auftragsgeber, das Unternehmen SYScor EDV, ist ein WinLine Vertriebs- und Entwicklungspartner. Die Software WinLine ist eine ERP-Lösung für Unternehmen, welche die Ressourcenplanung eines Unternehmens unterstützt. Zu diesem Zweck werden zu verschiedenen Bereichen eines Unternehmens Datenbanken angelegt und mit der Hilfe des ERP-Systems miteinander verknüpft. Der Anwender eines WinLine ERP-Systems erlangt damit zum Beispiel eine Übersicht über die erfassten Kunden- und den zugehörigen Ansprechpartnerdaten. Durch diese Informationsmöglichkeit kann die Kundenbindung gefördert werden. Hierfür soll eine mobile Android-Anwendung mit Zugriff auf das WinLine ERP-System erstellt werden, welchen Mitarbeitern eines Unternehmens Kunden- und Ansprechpartnerdaten übersichtlich visualisiert. Durch diese mobile Anwendung können Mitarbeiter, jederzeit auf diese Informationen zugreifen. Auch kann mit den Funktionen der Anwendung, welche auf diese Daten zugreifen, ein innovatives Hilfsmittel entstehen. Zum Beispiel kann eine Funktion erstellt werden, welche eine in dem ERP-System hinterlegte Adresse mit Google Maps öffnet, damit eine einfache Navigation des Mitarbeiters zu dem Kunden möglich ist.

#### **2.1.2 Ausgangslage und Zielsetzung**

Der Auftragsgeber verwendet bereits das Vorgängerprojekt „Mobile WinLine“, mit diesem ist jedoch keine Offline-Benutzung möglich und die Anwendung stellt alle Daten eines ERP-Systems dar. Für den Zweck, ausschließlich Kunden und Ansprechpartner anzuzeigen ist diese Anwendung zu unübersichtlich. Daher ist es ein Ziel dieses Projekts eine Offline-Verfügbarkeit sowie eine übersichtliche Darstellung von Kunden und Ansprechpartnern zu bieten. Ein weiteres Ziel ist die Gewährleistung der Sicherheit, da die Anwendung persönliche Daten verarbeitet.

Für die Realisierung der nativen Android Anwendung ist es notwendig Daten aus dem ERP-System abzurufen und in der Anwendung möglichst intuitiv zu visualisieren. WinLine bietet dafür die Möglichkeit ausgewählte Daten aus dem ERP-System mit „WinLine Webservices“ bereitzustellen. Der Auftragsgeber stellte für dieses Projekt einen exemplarischen Webservice mit Kunden- und Ansprechpartnern bereit. ERP-Connect soll sich zu diesem Webservice verbinden können und die hinterlegten Kunden- und Ansprechpartner darstellen. Die Verbindungs-Eigenschaften sollen individuell konfiguriert werden können.

# **3 Grundlagen und Methoden**

## **3.1 Konzept für die Programmierung**

Eine Android-Anwendung basierend auf dem offiziellen Android-Framework kann mit den Programmiersprachen Java oder Kotlin programmiert werden.

### **3.1.1 Programmierung mit Java**

Der Autor Eugen Richter zeigt auf, dass das Android-Framework für die Programmiersprache Java und nicht für Kotlin entworfen wurde. (vgl. Richter 2019, S. 10) Java ist eine weitverbreitete Programmiersprache, welche im Bildungsweg der HTL Traun unterrichtet wird. Von den Kenntnissen der Programmiersprache kann die Entwicklung der Anwendung profitieren, da keine Einarbeitungszeit in die Programmiersprache notwendig ist. Aufgrund der weiten Verbreitung von Java würde sich die Beschaffung von passender Literatur für die Entwicklung von Android-Anwendungen erleichtern.

### **3.1.2 Programmierung mit Kotlin**

Die Syntax von Kotlin ist im Vergleich zu Java deutlich schlanker und kürzer. Ein Beispiel dafür sind sogenannte „Data Classes“, bei denen für die Klassenvariablen automatisch Setter- und Gettermethoden erzeugt werden. Dadurch ergibt sich der Vorteil, dass eine Anwendung mit Kotlin effizient programmiert und Boilerplatecode vermieden werden kann. Die Syntax ist Java sehr ähnlich, dadurch bleibt die Lernkurve der Programmiersprache niedrig.

Eine weitere Eigenschaft von Kotlin ist die Null-Sicherheit. Kotlin verfügt über „nullbare“ und „nichtnullbare“ Datentypen. Bei der Verwendung von nullbaren Datentypen muss bei einem Zugriff geprüft werden, ob diese ungleich null sind, ansonsten würde es zu einen Compilerfehler führen. Durch diese Nullsicherheit kann eine NullPointerException vermieden werden, wodurch Programmfehler vermindert und die Sicherheit erhöht wird.

Da bei dem Kompilieren von Kotlin ein Java kompatibler Bytecode für die JVM erzeugt wird, ist Kotlin zu Java kompatibel. Dadurch kann Kotlin-Code Java-Code aufrufen und auch andersherum. Somit können bei der Verwendung von der Programmiersprache Kotlin auch Java-Libraries verwendet werden.

Kotlin besitzt keine „Checked Exceptions“, dadurch kann übersehen werden eine notwendige Ausnahme zu prüfen. Dennoch ist von Vorteil, dass dadurch überflüssiger Code vermieden werden kann.

Der Autor Staudemeyer zeigt einen weiteren Vorteil der Programmiersprache auf. Durch die Verwendung von „Kotlin Android Extensions“ können Elemente des Layouts in eine Klasse importiert werden und direkt angesprochen werden. Ansonsten müssten diese aufwendig mit Funktionen anhand der Ressourcen-ID aufgerufen werden. (vgl. Staudemeyer 2019, S. 73)

### **3.1.3 Entscheidung**

Im Vergleich zu Java unterstützt Kotlin durch die kürzere Syntax effizienteres programmieren. Auch die gegebene Null-Sicherheit spricht für diese Programmiersprache. Noch dazu gelingt der Umstieg von Java auf Kotlin sehr einfach. Zudem kann bei Bedarf mit Java, aufgrund der Kompatibilität, weiterprogrammiert werden. Ein weiterer Grund für die Entscheidung, Kotlin für dieses Projekt zu verwenden, ist die einfache Einbindung von Layout-Elementen im Programmcode durch die „Kotlin Android Extensions“. Dadurch können Fehler bei der falschen Anwendung der Ressourcen-Funktionen vermieden werden.

## 3.2 Konzept für die Speicherung der Daten

Um die Offline-Verfügbarkeit zu gewährleisten, müssen die Daten auf dem mobilen Gerät gespeichert werden. Auf die Art wie diese Daten gespeichert werden, haben Anforderungen an die Sicherheit und Performance einen großen Einfluss.

### 3.2.1 Speicherung als XML-Dokument

XML (Extensible Markup Language) ist eine Auszeichnungssprache, ähnlich zu HTML. Auszeichnungssprachen beschreiben den Aufbau von Daten. XML dient der Darstellung strukturierter Daten im Format einer Textdatei. XML wurde zur Speicherung und dem Transport von Daten entworfen und besitzt selbst keine Funktion. Eine XML-Datei besteht aus Elementen, welche jeweils mit einem Tag geöffnet und geschlossen werden, Attributen und Wertzuweisungen. Die Bezeichnung der Tags können selbst definiert werden, wodurch der Inhalt erweiterbar ist. Die wenigen Richtlinien legen fest, dass Tags geöffnet und wieder geschlossen werden müssen und ein XML-Dokument nur ein Wurzel-Element (auch Rootelement genannt) besitzen darf. Diese Richtlinien nennt man Wohlgeformtheit (Well Formed), und müssen immer erfüllt sein. Die einfache Struktur ermöglicht eine gute Lesbarkeit, Austauschbarkeit und Erweiterbarkeit der Daten, sowohl durch die Person als auch durch den Computer.

```
<MESOWebService TemplateType="1" Template="KontenWebservice">
  <KontenWebservice>
    <Kontonummer>230000</Kontonummer>
    <Kennzeichen>2</Kennzeichen>
    <Kontoname>Diverse Debitoren</Kontoname>
    <Staat>A</Staat>
    <Land>Österreich</Land>
  </KontenWebservice>
  <KontenWebservice>
    <Kontonummer>2302050011</Kontonummer>
    <Kontoname>Heinrich Hill GmbH</Kontoname>
    <Staat>A</Staat>
    <Postleitzahl>1110</Postleitzahl>
    <Ort>Wien</Ort>
    <Land>Österreich</Land>
    <E-Mail-Adresse>office@hill-gmbh.net</E-Mail-Adresse>
  </KontenWebservice>
</MESOWebService>
```

Abbildung 1: Ausschnitt aus dem bereitgestellten XML-Dokument des Webservices

Ein Vorteil von XML ist, dass XML eine einfache Verarbeitung ermöglicht. Des Weiteren eignet sich XML durch die standardisierte und strukturierte Weise für den Datenaustausch zwischen unterschiedlichen Programmkomponenten und verschiedenen Systemen. Da die Anwendung Daten aus einem REST-Webservice in dem Format XML empfängt, welche wiederum gespeichert und geladen werden, kann von dieser Art des Datenaustausches profitiert werden.

Der Autor Michael Inden zeigt den Nachteil auf, dass bei Verwendung von XML mit großen Datensätzen eine Redundanz, durch nicht notwendige Element Start- und End-Tags entsteht, welche in Relation zu den Nutzdaten sehr groß ist. (vgl. Inden 2016, S. 1 - 4)

Die Sicherheit des gesamten XML-Dokumentes kann durch die Verschlüsselung der Daten erfolgen. Zur Speicherung der XML-Datei kann das Android-Filesystem verwendet werden.

### **3.2.2 Speicherung in einer relationalen Datenbank**

SQLite, ein „leichtgewichtiges, aber leistungsfähiges relationales Datenbanksystem“, ist in Android eingebettet. (Staudemeyer 2018, S. 223). Diese relationalen Datenbanken können zur Speicherung von Daten verwendet werden. Der Vorteil einer Datenbank ist, dass durch Daten-Beziehungen und Normalisierung der Datenbank Redundanzen vermieden werden können. Auch sind die hinterlegten Daten effizient durchsuchbar und änderbar.

Ein Nachteil einer relationalen Datenbank ist, die komplexe Struktur von Kotlin-Objekten. Diese müssen in die Form einer Tabelle gebracht werden, was zu Fehlern und einem aufwendigen Aufbau führen kann. Richter führt an, dass auf die SQLite Datenbank der Anwendung nur diese selbst zugreifen kann und deshalb auf eine Rechteverwaltung verzichtet wurde. (vgl. Richter 2019, S. 129) Weiters ist eine Verschlüsselung der Datenbank nur mit der kostenpflichtigen Erweiterung „SQLite Encryption Extension“ möglich.

### **3.2.3 Speicherung in einer objektorientierten Datenbank**

Um die Speicherung von Kotlin-Objekten im Vergleich zu einer relationalen Datenbank zu vereinfachen kann eine objektorientierte Datenbank verwendet werden. Die Datenbank „Realms“ wäre hierfür ein Beispiel. In einer objektorientierten Datenbank werden Kotlin-Objekte direkt in einer Datenbank gespeichert. Der Vorteil hierbei ist, dass Objekte aus einer objektorientierten Datenbank wieder als Objekte geladen und weiterverwendet werden können. Ein Nachteil ist, dass Probleme in der Schnittstelle zwischen Anwendung und Datenbank, durch fehlende Standards auftreten können.

### **3.2.4 Speichern von Daten in SharedPreferences**

Eine Android-App kann den Speicherbereich „SharedPreferences“ für das Persistieren von Daten verwenden. Dieser Speicherbereich dient zur Speicherung von Einstellungen, kann jedoch auch andere Werte eines primitiven Datentyps speichern. Die Daten werden mit einem Schlüssel verknüpft gespeichert und geladen. Die Speicherung erfolgt durch das Betriebssystem in einem XML-File des Anwendungs-Daten-Ordners. Das Speichern und Laden der Daten kann überall in der Anwendung stattfinden, solange der Anwendungs-Context zur Verfügung steht.

Ein Vorteil ist hierbei, dass die Daten bis zur Deinstallation der Anwendung persistent gespeichert werden. Ein weiterer Vorteil liegt darin, dass ein Event-Listener auf die Änderung dieser Daten, ohne großen Aufwand, gesetzt werden kann. Eine Einschränkung hingegen ist, dass nur primitive Datentypen, verknüpft mit einem Key, gespeichert werden können. Folglich eignet sich diese Methode nicht für komplexe Daten.

### **3.2.5 Entscheidung**

„SharedPreferences“ eignen sich im Vergleich zu den anderen Methoden für nicht komplexe Daten. Mit „SharedPreferences“ wäre es dementsprechend schwierig die komplexen Daten logisch zu speichern. Da die Daten bereits in dem Format XML aus dem Webservice geladen werden, ist es nicht schwer diese in dem ursprünglichen Format „XML“ zu speichern. Auch die Gewährleistung der Sicherheit ist mit der Verwendung von XML einfacher, da dabei das gesamte File verschlüsselt und entschlüsselt werden kann. SQLite würde dafür keine kostenlose Möglichkeit anbieten. Eine Verschlüsselung wäre hingegen mit „Realms“ möglich, jedoch entstanden bei der Verwendung von „Realms“ Schnittstellen Probleme mit der Programmiersprache Kotlin.

Im Vergleich zu der Speicherung als XML können mit relationalen Datenbanken Redundanzen vermieden werden. Dennoch kann mit XML eine geringe Dateigröße erreicht werden. Dabei muss darauf geachtet werden, nicht benötigte oder leere Elemente nicht zu speichern.

Aufgrund dieser Vorteile wurde entschieden, XML zur Speicherung der Daten zu verwenden. Da Verbindungeigenschaften auch gespeichert werden müssen und deren Änderungen leicht erfassbar sein sollen, wurde der Speicherbereich „SharedPreferences“ verwendet.

### **3.3 Gewährleistung der Sicherheit**

Da die Anwendung personenbezogene Daten verarbeitet, muss der Schutz des Datenverkehrs und der zu verarbeitenden Daten gewährleistet sein. Dafür soll die Kommunikation zwischen Anwendung und Webservice abgesichert werden. Durch diese Absicherung wird verhindert, dass durch Überwachung des Netzwerkverkehrs eine nichtberechtigte Person Daten entnehmen kann. Um die Anforderung der Sicherheit zu erfüllen, soll auch unzulässiges Auslesen der lokal gespeicherten Kunden- und Ansprechpartnerdaten verhindert werden. Zum Schutz des Benutzers muss ein unzulässiges Auslesen des für die Verbindung benötigten Passwortes verhindert werden.

#### **3.3.1 Sichere Webservice-Kommunikation**

Kunden- und Ansprechpartnerdaten werden aus dem bereitgestellten REST-Webservice zu der Anwendung transportiert. Diese Kommunikation kann über HTTPs abgesichert und verschlüsselt werden. Nach Absprache mit dem Auftragsgeber wurde die mögliche Kommunikation des Webservices von „http“ auf „https“ (HTTP mit TLS) umgestellt. Damit wird die Kommunikation vollständig mit TLS verschlüsselt.

Die Authentifizierung gelingt bei „WinLine Webservices“ über dessen URL. Dabei werden Passwort und Benutzername als Query-Parameter in der URL übergeben. Mit „https“ werden zwar die Query-Parameter bei einer Verbindung verschlüsselt, unter Umständen kann es jedoch passieren, dass die vollständige URL inklusive der Query-Parameter lokal unverschlüsselt gespeichert werden (z.B. im Browserverlauf eines Webbrowsers, der den Webservice aufruft). Ein Angreifer kann daraus Zugangsdaten entnehmen. Bei ERP-Connect wurde darauf geachtet, dass die vollständige URL niemals lokal gespeichert wird. Der für die Authentifizierung notwendige Query-Parameter „Passwort“ wird lokal verschlüsselt gespeichert und die URL erst bei Verwendung unverschlüsselt zusammengefügt.

#### **3.3.2 Verschlüsselung der lokalen Daten**

Die Anwendung speichert Kunden- und Ansprechpartnerdaten und Webservice-Verbindungseigenschaften. Diese Daten müssen zur Gewährleistung der Sicherheit verschlüsselt werden.

#### **Verschlüsselungsmethode**

Es gibt zwei Verschlüsselungsmethoden. Symmetrische und asymmetrische Verschlüsselung. Bei der symmetrischen Verschlüsselung wird eine Nachricht (Klartext) mit einem Schlüssel verschlüsselt. Dieser Schlüssel wird auch für die Entschlüsselung verwendet. Die asymmetrische Verschlüsselung verwendet hingegen zwei voneinander abhängige Schlüssel. Einer für die Verschlüsselung des Textes, ein anderer zur Entschlüsselung der Daten. Diese Verfahren eignen sich dafür Daten oder Absender zu authentifizieren. Ein Problem der symmetrischen Speicherung ist die Schlüsselspeicherung, da die Daten mit einem abhandengekommenen Schlüssel entschlüsselt werden könnten. Zur Lösung des Problems erwähnt der Autor die Möglichkeit den Schlüssel in einem KeyStore, anstatt in einer Datei oder SharedPreferences zu speichern. (vgl. Pragati 2013, Position 2805) Ein Vorteil der symmetrischen Verschlüsselung ist dessen Performanz im Vergleich zu Algorithmen der asymmetrischen Verschlüsselung.

#### **Verschlüsselungsalgorithmus und der Schlüsseleigenschaften**

Es wurde entschieden den AES (Advanced Encryption Standard) Algorithmus zu verwenden. Denn Pragati erklärt, dass dieser im Vergleich zu anderen gängigen blocksymmetrischen Verschlüsselungsalgorithmen, zu denen auch DES und DES3 zählen, deutlich sicherer ist. (vgl. Pragati 2013, Position 2033) Bei einer blocksymmetrischen Verschlüsselung wird der Klartext in Blöcke fester Größe geteilt und mit

dem Schlüssel einmalig verschlüsselt. Da jeder Block einzeln mit dem gleichen Schlüssel verschlüsselt wird, können Muster des Klartextes in den verschlüsselten Text übertragen werden. Um dies zu vermeiden, wurde entschieden, als Schlüsseleigenschaft den Blockmodus CBC zu verwenden. CBC verbindet Klartextblöcke über XOR mit dessen Vorgänger. Für die Verbindung des ersten Blocks wird ein sogenannter Initialisierungsvektor verwendet. Dieser wird auch für die Entschlüsselung benötigt. Ein weiteres Problem der blocksymmetrischen Verschlüsselung ist die Tatsache, dass die gesamte Blockgröße auf ein Vielfaches der Schlüsselgröße erweitert sein muss. Da die zu verschlüsselnden Daten unterschiedlich groß sein können, wurde entschieden, Padding zu verwenden. Dies füllt den letzten Block mit Zufalls-werten auf.

### **3.3.3 Analyse der Sicherheit der Daten unter Android**

SharedPreferences sind nur für die Applikation zugänglich. Auch lokal gespeicherte Daten, wie zum Beispiel die Kunden- und Kontaktdaten, können nur durch die Applikation abgerufen werden. Zur Gewährleistung dieser Sicherheit führt Android Applikationen in einer Sandbox aus. Die lokal gespeicherten Daten werden auch in dieser Sandbox gespeichert, auf welche nur die Anwendung Zugriff hat.

Durch einen Superuser (Root)-Zugriff könnten diese gespeicherten Daten trotzdem bearbeitet und abgerufen werden. Deswegen wurde entschieden die Daten dennoch zu verschlüsseln.

# 4 ERP-Connect

Die Anwendung ERP-Connect dient dazu, Mitarbeitern eines Unternehmens welches WinLine-ERP-Syste me verwendet, Daten über deren Kunden und den zugehörigen Ansprechpartner darzustellen. Die Da ten werden in den folgenden Ansichten dargestellt:

- Kunden-Listenansicht,
- Kunden-Detailansicht,
- Ansprechpartner (Kontakte)-Listenansicht,
- Ansprechpartner (Kontakte)-Detailansicht.

Diese Ansichten, inklusive dem Einstellungsmenü, werden in den nächsten Kapiteln näher erklärt.

## 4.1 Startfenster und Kundenlistenansicht

Bei dem Start der App öffnet sich die Kundenlistenansicht. Die Kundenlistenansicht stellt alle in dem ERP-System hinterlegten Kunden des Unternehmens in einer, nach dem Alphabet geordneten, Listenansicht dar. Jeder Listeneintrag besteht aus dem Namen des Unternehmens sowie der Kontonummer des Unternehmens. Diese Bestandteile können auch zur Filterung der Liste verwendet werden.

Beim Start stellt die App die Verbindung zu dem, in den Einstellungen hinterlegten, Webservice her. Die Daten werden nach dem Aufbau einer Verbindung heruntergeladen und persistent gespeichert. Während des Verbindungsbaues wird ein Ladesymbol angezeigt.



Abbildung 2: Kunden-Listenansicht

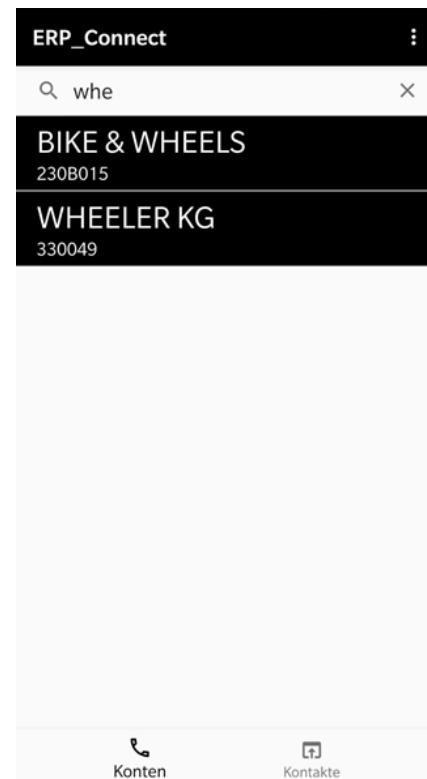


Abbildung 3: Gefilterte Listen-Ansicht

Bei dem Auftreten eines Fehlers während der Verbindung oder der Speicherung der Daten wird der Fehler in einer Snackbar dargestellt. Wenn die Verbindung fehlschlägt, jedoch persistierte Daten vorhanden sind und diese geladen werden können, wird dies ebenfalls in einer Snackbar dargestellt. Dadurch wird hingewiesen, dass die Daten möglicherweise nicht aktuell sind. Das Einstellungsmenü besitzt die Funktion „Retry“, für das Starten eines neuen Verbindungversuches und einen Button zum Öffnen der Verbindungseigenschaften.



Abbildung 4: Ausgeklapptes Einstellungsmenü

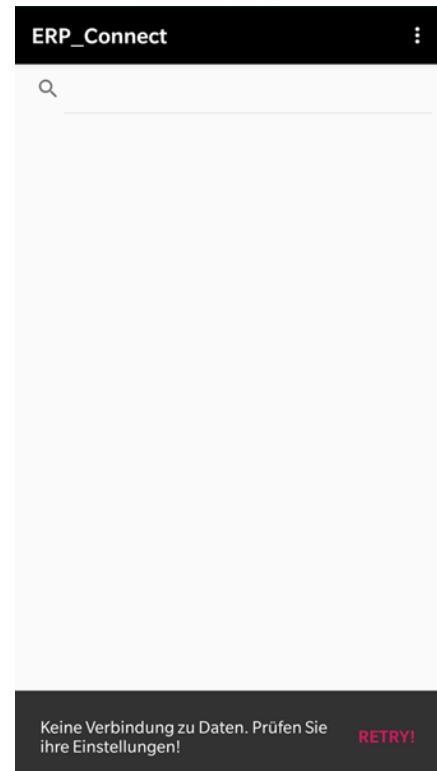


Abbildung 5: Fehlermeldung

## 4.2 Kundendetailansicht

Die Kundendetailansicht dient dazu, nähere Informationen über den Kunden darzustellen und verschiedene Funktionen zur Erleichterung der Kommunikation mit einem Kunden anzubieten. Diese Ansicht kann über die Auswahl eines Listeneintrages in der Kundenlistenansicht aufgerufen werden. Genauso wie bei der Listenansicht, wird zuerst versucht Daten aus dem Webservice zu laden, um eine Aktualität der Daten zu gewährleisten. Fehler werden ebenfalls in einer Snackbar dargestellt.



Abbildung 6: Kundendetailansicht mit Funktionen

Durch die Verwendung des Buttons „Ihre Ansprechpartner“ startet die Ansprechpartner-Listenansicht. Diese stellt dann die zu dem Unternehmen zugehörigen Ansprechpartner dar. Die restlichen Buttons dienen zur Erleichterung der Kommunikation mit dem Kunden. Dabei wird vorerst geprüft, ob die im ERP-System hinterlegten Daten vollständig sind, anschließend können folgende Funktionen durchgeführt werden:

1. Starten eines Anrufes an die in dem ERP-System hinterlegte Nummer.
2. Starten einer SMS-Kommunikation mit einer im ERP-System hinterlegten Mobilnummer.
3. Öffnen der hinterlegten Adresse mit Google Maps.
4. Öffnen der hinterlegten Internetadresse.

## 4.3 Darstellung der Ansprechpartner

ERP-Connect stellt eine Übersicht über alle Ansprechpartner (Kontakte) der Kunden dar. Dafür gibt es eine Ansprechpartner-Listenansicht und eine Ansprechpartner-Detailansicht. Diese ähneln dem Aufbau der zuvor genannten Ansichten. Ein Unterschied ist, dass bei der Detailansicht eine Funktion für den Start einer E-Mail-Konversation existiert. Auch die Listenansicht wurde mit dem Abteilungsnamen des Ansprechpartners, für einen besseren Überblick, ergänzt.

The screenshot shows a contact list titled "ERP\_Connect". It lists eight contacts:

- Bechem Carl (Einkauf | Einkauf)
- Bechem Maria (Einkauf | Einkauf)
- Beck Tom (Einkauf | Leitung EK)
- Behar Ulrike (Sekretariat | Sekretariat)
- Behr Oliver (Einkauf | Leitung EK)
- Ben Jenny (Buchhaltung | Buchhaltung)
- Benesz Jutta (Support | GF)
- Benl Eckhard (Lager | Lager)

At the bottom are navigation icons for "Konten" (Accounts) and "Kontakte" (Contacts).

Abbildung 7: Ansprechpartner Listenansicht

The screenshot shows a detailed contact view for "Achim Clarisa" from "ERP\_Connect". The contact details are as follows:

PERSON	
Vorname	Clarisa
Nachname	Achim
Geschlecht	Weiblich
Abteilung	Einkauf
Funktion	Einkauf
KONTAKT	
Mail	abtaenzer.cl@sv-asten.at
Homepage	<a href="http://www.sv-osten.at">www.sv-osten.at</a>
MOBILTELEFON	
Land-Vorwahl	
Mobil-Betreiber	
Nummer	
TELEFON	
Land-Vorwahl	+43
Ort-Vorwahl	
Nummer	-331

Abbildung 8: Ansprechpartner Detailansicht

## 4.4 Einstellungsmenü für die Verbindungseigenschaften

Das Einstellungsmenü ermöglicht verschiedene Anpassungen, zudem können hier die Verbindungseigenschaften zu dem WinLine Webservice festgelegt werden.

Durch die Einstellung „URL des Webservices“ kann die Basis-URL gewünschten Webservices festgelegt werden. Eine Basis-URL kann eine Domain mit oder ohne Subdomain oder eine IP-Adresse des Webservices sein. Die URL wird automatisch, für eine sichere Kommunikation, mit dem https-Schema ergänzt, oder das http-Schema mit einem https-Schema ersetzt. Auch ein benötigter Schrägstrich am Ende der URL wird, falls erforderlich, ergänzt. Das Einstellungsfenster dient auch zur Festlegung der Authentifizierung des Benutzers am Webservice. Dabei wurde darauf geachtet, bei der Eingabe des Passworts mit Sternen zu ersetzen, um eine Sicherheit bei der Eingabe des Passwortes zu gewährleisten.

Um eine bessere Usability zu bieten, stehen die Funktionen „Timeout: Connection“ und „Timeout: Read“ zur Verfügung. Damit kann die maximale Dauer für die Verbindung zu dem Webservice und die maximale Dauer einer Antwort des Webservices festgelegt werden.

Durch Auswahl von „Synchronisieren von Kontakten bei Start“ kann festgelegt werden, ob beim Start direkt alle Daten heruntergeladen und gespeichert werden sollen. Wenn diese Funktion nicht ausgewählt ist, werden die Kontaktdaten nur beim Start der Kontakte-Listenansicht heruntergeladen und persistiert.

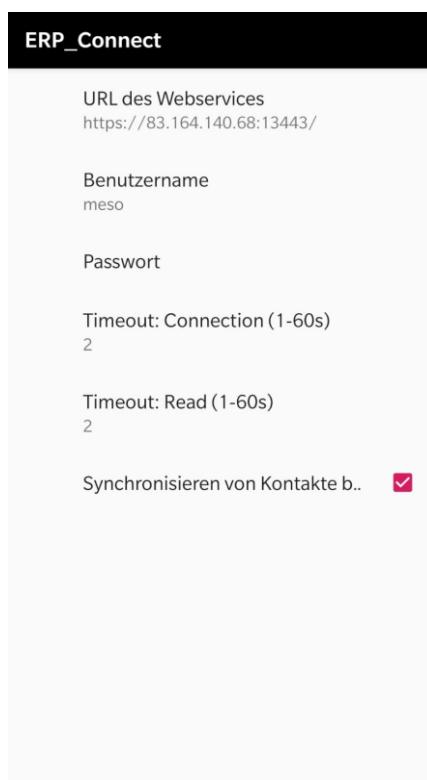


Abbildung 9: Einstellungsmenü für die Verbindungseigenschaften

# 5 Technische Dokumentation

## 5.1 Erstellung einer Android App

Die Android-Anwendung wurde mit der Entwicklungsumgebung Android Studio entwickelt. Die Entwicklungsumgebung bietet unter anderem einen Emulator für die Ausführung der Anwendung, einen Layout-Editor, eine einfache Einbindung externer Libraries mit dem Buildsystem „Gradle“ und Lint-Regeln welche Codeoptimierungen vorschlagen. Bei der Programmierung einer nativen Android-App wird das Android-SDK verwendet, welches die Java-SDK mit Android-spezifischen Funktionalitäten erweitert.

### 5.1.1 Architekturmuster

Ein Architekturmuster ist notwendig, um die Struktur des Programmcodes durch Trennung der Präsentierlogik von der Geschäftslogik zu verbessern. Es wurde entschieden, das Architekturmuster MVP zu verwenden. Dieses Architekturmuster besteht aus den Komponenten „View“ (Darstellung der Daten, Input-Verarbeitung), „Presenter“ (Kommunikation zwischen View und Model) und dem „Model“ (Geschäftslogik, Datenbeschaffung). Im Vergleich zu dem Architekturmuster MVC kommunizieren, „View“ und „Model“ nie miteinander, sondern nur über deren Zwischenstelle „Presenter“. Dadurch ist die „View“ vollständig unabhängig vom „Model“. Um den Aufbau näher zu erklären, folgt eine Übersicht über die verschiedenen Interfaces. Die Klassen implementieren diese und kommunizieren mit anderen Klassen nur über deren Interfaces.

```
interface KontoListContract {  
    interface View {  
        fun showProgress()  
        fun hideProgress()  
  
        fun onSucess(finishCode: String)  
        fun onError(failureCode: String)  
        fun displayKontoListInRecyclerView(kontoList: List<Konto>)  
    }  
    interface Presenter {  
        fun requestFromWS()  
        fun onDestroy()  
    }  
    interface Model {  
        interface OnFinishedListener {  
            fun onfinished(kontoArrayList: List<Konto>, finishCode: String)  
            fun onFailure(failureCode: String)  
        }  
        fun getKontoList(onFinishedListener: OnFinishedListener)  
    }  
}
```

Um die Daten aus dem Model in den Presenter zu transferieren wurde ein Listener (OnFinishedListener) verwendet.

### 5.1.2 Activities

Eine Android-Anwendung besteht aus verschiedenen Bildschirm-Ansichten, diese werden Android „Activities“ genannt. Dementsprechend wurde für jede zuvor erwähnte Ansicht eine eigene Activity erstellt. Activities haben auch Zugriff auf den „Applikations-Context“. Dieser Context ist für das Starten von anderen Activities, für den Zugriff auf das Filesystem und die Bereitstellung von Informationen bezüglich der Anwendung zuständig.

Activity-Klassen besitzen einen eigenen Lifecycle. Dies bedeutet, dass „zu bestimmten Punkten der Lebenszeit der Klasse [...] bestimmte Methoden aufgerufen“ (Richter 2019, S. 112) werden. Bei der Durchführung des Projektes war es wichtig, den Programmfluss der Anwendung zu beeinflussen. Dafür wurden in den verschiedenen Activities die Lifecycle-Methoden, „onCreate“, „onResume“ und „onStop“ überschrieben.

Die „onCreate“-Methode wird bei dem Start der Activity aufgerufen. Dementsprechend werden dort die benötigten Listener und die Recyclerview initialisiert. Ebenso wird das Layout der Anwendung festgelegt. Auch der Presenter wird beauftragt, Daten aus Datenquellen zu beschaffen.

Die „onResume“-Methode wird aufgerufen, wenn die Activity wieder in den Vordergrund geholt wird. Dort wurde eine Prüfung hinzugefügt, ob die darzustellenden Daten noch vorhanden sind. Dies ist notwendig, da Einstellungsänderungen eine Löschung der gespeicherten Daten auslösen können.

Die „onDestroy“-Methode wird aufgerufen, wenn die Activity von dem Betriebssystem zerstört wird. Wenn dies geschieht soll der Presenter von der Activity entkoppelt werden.

Es besteht die Möglichkeit Fragmente in einer Activity einzubinden. Ein Fragment ist eine wiederverwendbare Klasse, welche einen Teil einer Activity beschreibt. Fragmente besitzen einen eigenen, von dem Activity-Lifecycle abhängigen, Lifecycle. Das Layout eines Fragments wird ebenso wie bei einer Activity in einer XML-Datei definiert. Es ist auch möglich ein Fragment ohne Layout zu erstellen, um Funktionen, welche einen Lifecycle benötigen, in mehreren Activities zu implementieren. Der Vorteil von Fragmenten ist, dass diese leicht wiederverwendet werden können und die Programmierung der Navigation zwischen Fragmenten, durch den Austausch der Fragmente, erleichtert wird.

Beispielsweise wurde für die Navigation zwischen zwei Activities eine „BottomNavigationView“ verwendet. Diese ist jedoch für die Fragment-Navigation erschaffen worden. Bei der Verwendung von Fragmenten würde diese Leiste nur einmal instanziert werden, da diese nur einmal im Activity-Layout deklariert werden muss. Bei der Navigation werden dann nur die Fragmente im Layout getauscht.

Dies führt zu Problemen bei der Navigation zwischen zwei Activities. Dabei wird zwar die Navigation korrekt ausgeführt, aber auf der Ziel-Activity wird eine neue Bottom-Navigation-View instanziert. Dadurch wird die Navigation falsch angezeigt, was jedoch durch manuelle Programmierung behoben werden kann.

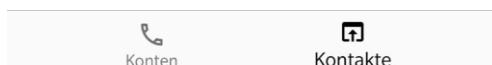


Abbildung 10: Erfolgreicher Wechsel



Abbildung 11: Falsche Anzeige

Es wurde dennoch darauf verzichtet eine Singleactivity-Applikation zu erstellen. Eine Singleactivity-Applikation besteht nur aus einer Activity und mehreren austauschbaren Fragmenten. Ein Nachteil von Fragmenten ist, dass deren Lebenszyklus „komplizierter als der einer Activity“ (Staudemeyer 2018, S. 179) ist und dadurch Fehler entstehen können. Außerdem wurde festgestellt, dass es vorkommen kann, dass ein Fragment keinen Context besitzt, welcher jedoch für Funktionen der Anwendung notwendig ist. Auch wird der Lifecycle eines Fragmentes nach dem Drehen des mobilen Gerätes neugestartet.

### 5.1.3 Intents

Intents werden von dem Android-Betriebssystem bereitgestellt. Diese beschreiben auszuführende Aufgaben und kontaktieren diesbezüglich die Zielkomponente. Die Aufgabe des Intents wird durch Programmierung definiert und können zum Starten einer externen Anwendung, dem Starten einer Activity, oder zur Übertragung von Daten verwendet werden.

#### Starten einer Activity mit Intents

Für die Navigation zwischen verschiedenen Activities ist es notwendig die Zielactivity zu starten. Dies geschieht mit einem konkreten Intent. „Wenn die Zielkomponente zur selben Anwendung gehört, gibt man sie durch den aktuellen Kontext und eine Referenz auf das Klassenobjekt an“. (Staudemeyer 2018, S. 115) Dadurch kann die Activity mit der Methode `startActivity` und dem Intent-Objekt gestartet werden.

```
val intent = Intent(this, AndereActivity::class.java)
startActivity(intent)
```

Durch das Starten der Ziel-Activity wird die ursprüngliche aktive Activity auf den Backstack verschoben. Activities auf dem Backstack können durch den „Zurück-Button“ eines Android-Gerätes aufgerufen werden.

#### Datenübertragung zwischen Activities mit Intents

Bei dem Start einer Detailansicht ist es für deren zuständige Activity notwendig zu wissen welches Konto oder welcher Kontakt geladen werden soll. Dafür gibt es bei Konten eine eindeutige Kontonummer, und bei Kontakten eine eindeutige Kontaktnummer. Diese Nummer muss nach der Wahl eines Listeneintrages in einer Listactivity zu der Detail-Activity übertragen werden. Intents bieten „die Möglichkeit, beliebige zusätzliche Informationen in Form von Schlüssel-/Wertpaaren zu speichern.“ (Staudemeyer 2018, S. 116) Diese Möglichkeit wird dafür verwendet, die Nummer auf die Ziel-Activity zu übertragen. Beifolgendem Code wird die Kontonummer als Extra an den Intent angehängt:

```
intent.putExtra("id", kontoList[position].kNumber)
```

Dieses Extra kann in der gestarteten Activity abgerufen werden.

```
val extra = intent.getStringExtra("id")
if (extra != null) {
    kontoNummer = extra
}
```

#### Starten von externen Applikationen mit Intents

Die Buttons auf den zuvor beschriebenen Detail-Ansichten führen verschiedene Aktionen mit den hinterlegten Kunden- oder Kontaktdata aus. Für diese Aktionen ist es notwendig externe Komponenten zu starten. Dafür eignen sich abstrakte Intents. Laut Staudemeyer benennen diese die auszuführende Aufgabe und die zugehörigen Informationen, welche für die Ausführung benötigt sind. Das Framework sucht dann die passende Komponente zur Durchführung des Intents. (vgl. Staudemeyer 2018, S. 113f)

```
val intent = Intent(Intent.ACTION_VIEW, address_url)
intent.setPackage("com.google.android.apps.maps")
if (intent.resolveActivity(packageManager) != null) {
    startActivity(intent)
}
```

Zum Beispiel zeigt dieser Code das Öffnen einer hinterlegten Adresse mit Google Maps. Dabei wird eine Google-Maps-API URL inklusive der zu öffnenden Adresse erstellt und als Parameter angehängt. Diese URL wird mit der Variable `adress_url` dem Intent übergeben. Diese URL ist ausschließlich mit Google-Maps kompatibel, daher wurde dieser Intent mit der Funktion `setPackage` auf die Anwendung „Maps“ beschränkt. Mit `resolveActivity` wird geprüft ob der Intent ausführbar ist, also ob die benötigten Komponenten für diesen Intent vorhanden sind.

### 5.1.4 Layout

Das Layout einer Activity wird in einer XML-Datei definiert. Diese kann entweder händisch oder durch den graphischen Layout-Editor von Android-Studio editiert werden. Es wurde festgestellt, dass durch die Verwendung des Layout-Editors die Gefahr besteht, dass nicht notwendiger XML-Code generiert wird, was zu einer schlechteren Übersicht des XML-Codes führt. Deswegen wurden Layouts händisch korrigiert.

### Layout der Detailansichten

Für die Darstellung wurden verschiedene Layouts verwendet. Diese definieren, wie sogenannte View-Elemente (zum Beispiel: „`TextView`“ – zur Darstellung von Texten, „`ButtonView`“ – zur Darstellung eines Buttons und die „`ProgressBar`“ – zur Darstellung eines Ladebalkens) angeordnet werden sollen. Da die darzustellenden Daten auch länger als eine Bildschirmhöhe sein können, wurde entschieden, als Root-Element das „`ScrollView`“-Layout zu verwenden. Dies ermöglicht Komponenten oder andere Layouts durch Scrollen zu erreichen. In der „`ScrollView`“ wurde ein `LinearLayout` definiert, welches die darin befindenden Elemente der Reihe nach horizontal abbildet. In der Detailansicht sollen Daten tabellarisch angezeigt werden. Dafür wurde ein sogenanntes `TableLayout` in das `LinearLayout` eingebunden. Damit können Zeilen und Spalten definiert werden, welche wiederum die darzustellenden View-Elemente definieren.

Durch verschiedene Attribute können Eigenschaften der Komponenten beeinflusst werden. Dies kann auch durch Programm-Code durchgeführt werden.

```
<TableRow android:id="@+id/rowCountry"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:weightSum="2">
    <TextView
        android:id="@+id/textStaat"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:text="@string/descriptionStaat"
        android:layout_width="0dp"
        android:layout_weight="1"/>
    <TextView
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:id="@+id/textInputStaat"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:textColor="@android:color/white"
        android:textStyle="bold"
        android:textIsSelectable="true" />
</TableRow>
```

## Layout der Listenansichten

Für die Darstellung von Daten einer Liste bietet Android die Möglichkeit an, ein ListView oder ein Recyclerview zu verwenden. Es wurde entschieden ein Recyclerview zu verwenden, da dieser bei großen Datenmengen effizienter als eine ListView ist. Ein Layout wird mit Programmcode mit einer Activity verknüpft.

Ein Recyclerview besteht aus mehreren Listen-Einträgen. Vorerst muss ein Layout für diese Einträge definiert werden.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tv_fullEntry"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/tv_konto_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/tv_konto_number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Es wird eine Klasse „ViewHolder“, abgeleitet aus der Klasse RecyclerView.ViewHolder, erstellt. Dort werden Variablen erstellt, welche direkten Zugriff auf die Elemente des Eintrags-Layouts besitzen.

Nun wird eine Adapter-Klasse erstellt. Diese wird aus der Klasse RecyclerView.Adapter<ViewHolder> abgeleitet. Für die Anpassung des Recyclerviews an die eigenen Anforderungen werden folgende Methoden überschrieben:

- onCreateViewHolder: Zur Festlegung des zuvor erstellen Eintrag-Layouts.
- onBindViewHolder(holder: ViewHolder, position: Int): Diese Methode erstellt die einzelnen Listeneinträge und bindet Werte an das Layout mit der Hilfe der zuvor erstellten ViewHolder-Klasse.

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    holder.tvKontoName?.text = kontoList[position].kName
    holder.tvKontoNumber?.text = kontoList[position].kNumber

    val intent = Intent(context, KontoDetailActivity::class.java)
    intent.putExtra("id", kontoList[position].kNumber)
    holder.tvHolder.setOnClickListener { v -> context.startActivity(intent) }
}
```

Da die darzustellende Liste bereits nach dem Alphabet geordnet ist und die Eintragsnummierung (Position) ebenfalls bei 0 startet, können Werte der Liste aus der gleichen Position verwendet werden.

Zusätzlich wurde wie oben zu sehen ein „onClickListener“ auf jeden Eintrag gesetzt, welcher bei der Auswahl eines Eintrages die zugehörige Detail-Ansicht startet.

Zur Erfüllung der Anforderung der Durchsuchbarkeit des Recyclerviews wurde ein Filter definiert, welcher die Einträge nach Suchkriterien filtert. Dieser Filter wird durch Benutzereingaben in die View-Komponente „SearchView“ aufgerufen.

## Verbesserung der Effizienz des Layouts durch Data Binding

Eine Möglichkeit die Effizienz des Layouts zu steigern ist die Verwendung von Data Binding. Damit können UI-Komponenten deklarativ an eine Datenquelle gebunden werden. Das bedeutet, dass Werte einer Datenquelle nicht wie üblich durch Programmcode im Layout gesetzt werden, sondern stattdessen deklarativ beschrieben wird welches Werte-Feld der Datenquelle verwendet werden soll. Es wurde entschieden Data Binding nicht zu verwenden, da die Datenquelle dafür observable (beobachtbar) sein muss. Um dies zu erreichen müssten beispielsweise Observable-Datentypen verwendet werden, was den Aufbau der Anwendung, im Vergleich zu dem erreichten Nutzen, deutlich erschweren würde.

### 5.1.5 Speichern und Laden von Einstellungen

Die Anwendung speichert Einstellungen. Um diese Einstellungsmöglichkeiten übersichtlich darzustellen, wird ein Fragment erstellt. Dieses erbt von der Klasse PreferenceFragmentCompat, welche sich für eine Listendarstellung einer Hierarchie von Einstellungsobjekten eignet. Diese Einstellungshierarchie muss, in Form einer XML-Datei, erstellt werden.

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <EditTextPreference
        android:title="url des Webservices"
        android:key="base_url"
        app:useSimpleSummaryProvider="true"/>
    <CheckBoxPreference
        android:key="auto_sync"
        android:title="Synchronisieren von Kontakte bei Start?"
        android:defaultValue="true" />
</PreferenceScreen>
```

Wichtig ist die Vergabe eines Schlüsselnamens der Einstellungsmöglichkeiten. Mit diesem Key wird der Wert in den SharedPreferences gespeichert und kann auch abgerufen werden. Durch useSimpleSummaryProvider wird die Benutzereingabe unter dem Einstellungsnamen im Einstellungsfenster angezeigt, was zu einer besseren Usability führt.

Diese XML-Datei wird in dem Settings-Fragment eingebunden. Die einzelnen Einstellungen können mit einem Preference-Change-Listener ausgestattet werden. Damit können Benutzereingaben geprüft werden. Auch werden gegebenenfalls, nach einer Änderung, Kunden- und Ansprechpartnerdaten gelöscht, damit unter einer neuen Verbindungseigenschaft keine alten Daten dargestellt werden.

Die Einstellungen können nun überall in der Anwendung mit dem PreferenceManager und dem Schlüssel abgerufen werden.

```
val sharedPref = PreferenceManager.getDefaultSharedPreferences(context)
val userName = sharedPref.getString("user_name", "")
```

### 5.1.6 Testen der Anwendung

Durch das Testen der Anwendung werden Fehler des Programmes entdeckt und ausgebessert. Eine Möglichkeit die Funktionalität der Anwendung zu testen ist die Anwendung auf einem Android Gerät manuell zu starten und verschiedene Funktionen auszuprobieren. Jedoch steht für das Testen an einem Gerät, nur eine begrenzte Anzahl an Testgeräten zur Verfügung. Deshalb ist es auch möglich die Anwendung auf einem virtuellen Gerät zu emulieren. Dies wurde verwendet, um das Verhalten des Layouts bei Geräten mit kleineren Displays zu prüfen.

Eine weitere Möglichkeit ist das Schreiben von Testfällen. Hierbei wird zwischen lokalen Unit Tests und instrumentierten Unit Tests unterschieden. Laut Staudemeyer, werden mit lokalen Tests Funktionen geprüft welche unabhängig vom Android-System funktionieren. Auch weist er auf, dass instrumentierte Tests zum Testen von Funktionen, welche Zugriff auf das System benötigen, dienen. (vgl. Staudemeyer 2018, S. 64)

Bei der Entwicklung wurde auch die Log-Api verwendet, diese dient dazu Log-Output in die Konsole zu schreiben. Damit kann zum Beispiel geprüft werden ob verschiedene Methoden erfolgreich ausgeführt wurden.

Für das Testen des Presenters war es notwendig sogenannte Mock-Objekte zu erstellen. Damit können Rückgabewerte von Funktionen des Mock-Objektes simuliert werden. In Kotlin sind alle Klassen final. Mockito unterstützt jedoch das mocken von „final-Klassen“ nicht. Deswegen wäre es notwendig alle zu mockende Klassen mit dem Schlüsselwort „open“ zu versehen. Um dies zu vermeiden, wurde die Konfiguration von Mockito mit der Zeile „mock-maker-inline“ erweitert. Durch das Verwenden der für Java entwickelten Mockito Library entstanden Null-Fehler, deswegen wurde eine an Kotlin angepasste Mockito-Library verwendet („Mockito-Kotlin“ von nhaarman).

Nun kann die View und das Model gemockt werden, und der Konstruktor des Presenters mit diesen Objekten aufgerufen werden.

```
var mView: KontoDetailContract.View = mock()
var mModelKonto: KontoDetailModel = mock()
var mPresenter: KontoDetailPresenter = KontoDetailPresenter(mView, mModelKonto)
```

Danach stehen verschiedene Mockito-Funktionen zur Verfügung.

```
@Test
fun loadDataSuccessFromWeb() {
    val konto = Konto("1", "Test")
    val finishedCode = FinishCode.finishedOnWeb

    doAnswer {
        val callback: KontoDetailContract.Model.OnFinishedListener = it.getArgument(0)
        callback.onfinished(konto, finishedCode)
    }.whenever(mModelKonto).getKontoDetail(any(), eq("1"))

    mPresenter.requestFromWS("1")
    verify(mView, never()).onError(any())
    verify(mView, never()).onSuccess(any())
    verify(mView).initListener(konto)
    verify(mView).hideProgress()
    verify(mView).setTextData(konto)
```

```
}
```

Mit @Test wird eine Funktion annotiert, um diese als Test zu markieren. Die Funktion doAnswer legt einen Rückgabewert einer Funktion, welche mit whenever() und dem Funktionsnamen festgelegt wird, fest. Damit können Funktionen des Models simuliert werden. Mit der Funktion verify wird überprüft ob bestimmte Funktionen des gemockten Objekten aufgerufen werden. Auch wurden mit JUnit-Tests der Output von Funktionen geprüft.

Activities und verschiedene Model-Funktionen greifen auf das Android-Betriebssystem zu. Deswegen ist es notwendig diese Tests instrumentiert auszuführen. Dafür wird die Klasse mit @RunWith(AndroidJUnit4ClassRunner::class) annotiert.

Für das Testen der Activities wurde entschieden die Espresso-Library zu verwenden. Damit ist es möglich festzustellen ob GUI-Elemente richtig angezeigt werden. Es ist notwendig ein testbares Objekt der Activity zu erstellen und diese zu starten.

```
var rule: ActivityTestRule<KontoListActivity> = ActivityTestRule(KontoListActivity::class.java)
val activity = rule.launchActivity(null)
```

Dann können für diese Activity Testfälle erstellt werden.

```
@Test
fun testDisplayRecyclerview() {
    val activity = rule.launchActivity(null)
    val konto = Konto("100")
    val kontolist = listOf<Konto>(konto)
    activity.runOnUiThread(object : Runnable {
        override fun run() {
            activity.displayKontoListInRecyclerView(kontolist)
        }
    })
    onView(withId(id.rv_konto_list)).check(matches(isDisplayed()))
    onView(withId(id.search_konto)).check(matches(isDisplayed()))
}
```

Wichtig ist hierbei, dass Funktionen die auf GUI-Elemente zugreifen, mit der Funktion runOnUiThread() auszuführen sind. Dadurch wird die Funktion auf dem UI-Thread, anstelle des Haupt-Threads der testenden Activity, ausgeführt. Ansonsten würde bei dem Zugriff auf GUI-Elemente außerhalb des UI-Threads Fehler entstehen.

Nun stehen die Funktionen der Library Espresso zur Verfügung. Aus dem Beispiel zu entnehmen, ist die Prüfung der Sichtbarkeit des Elementes mit der ID „rv\_konto\_list“ (die Recyclerview). Dafür werden die Funktionen „onView“, „check“, „matches“ und „isDisplayed“, der Espresso Library verwendet.

Auch war es notwendig einen Countdown für asynchrone Methoden einzufügen, damit der Test erst fortgesetzt wird, wenn eine asynchrone Methode einen Wert liefert. Dafür wurde die Java-Klasse CountdownLatch verwendet.

## 5.2 Webservice-Zugriff

Für die Anwendung ist es notwendig Daten aus einem Webservice zu beziehen. Hierbei kann es sich sowohl um Listen aller Konten, Listen aller Kontakte, als auch um einzelne Konten beziehungsweise Kontakte handeln. In den folgenden Kapiteln wird erklärt wie die Library Retrofit für die Kommunikation mit dem Webservice durchgeführt wird.

### 5.2.1 SimpleXML-Converter

Die aus dem Webservice gelieferten XML-Daten müssen verarbeitet werden. Dadurch entsteht die Aufgabe die Daten eines XML-Dokumentes zur weiteren Verarbeitung in Objekte umzuwandeln.

Der Autor beschreibt, dass das Auslesen der Informationen aus dem XML, die Umwandlung in die Java-Welt und später wieder die Rückwandlung in ein korrespondierendes XML-Dokument eine Aufgabe des Entwicklers ist. Dafür kann die API JAXB verwendet werden. Dabei wird zwischen XML-Dokument und Java-Objekten automatisch abgeglichen. Die Konvertierung von XML nach Java und zurück wird mittels Konventionen und Annotationen festgelegt. (vgl. Inden 2016, S. 49)

Es stellte sich heraus, dass Android JAXB nicht unterstützt. Als Alternative wird SimpleXML verwendet. Dieser Converter verwendet ähnlich zu JAXB, Annotationen und Konventionen zur Kennzeichnung der Objekte.

### 5.2.2 Erstellung von Entitäten

Entitäts-Klassen sind Klassen, welche dafür verwendet werden, Daten des Webservices in Kotlin-Objekte zu speichern. Diese Entität-Klassen werden auch benötigt, um Daten aus den lokal gespeicherten Dateien zu laden. Im Laufe des Projektes stellte sich heraus, dass es sich für die Erstellung von Entitäts-Klassen empfiehlt Kotlin-Datenklassen zu verwenden. Dadurch werden Setter- und Getter-Methoden automatisch generiert. In den Datenklassen wird der Aufbau des XML-Response aus dem Webservice mit Objektattributen, welche mit Annotationen beschrieben werden, nachgebaut. Es wurde darauf geachtet nur jene Objekt-Attribute zu definieren welche in der Anwendung auch benötigt werden, um möglichst wenig Daten zu speichern.



Abbildung 12: UML-Diagramme der Entitäts-Klassen

Jene Klassen, welche auf „List“ enden besitzen eine Liste aller Konten/Kontakte. Die Klassen „Kontakte“ und „Konto“ beschreiben die Eigenschaften eines Ansprechpartners/Kunden. Es ist zu beachten, dass in diesem Kapitel nur auf den Aufbau der Klassen KontoList und Konto näher eingegangen wird, da der Aufbau der Klassen „KontakteList“ und „Kontakt“ sich sehr ähnelt.

Für das Mapping durch den SimpleXML-Converter ist es notwendig die Entitäts-Klassen, abhängig zu dem XML-Aufbau, zu annotieren.

```
<MESOWebservice>
    <KontenKontenWebservice>
        <Kontoname>HTL Traun</Kontoname>
        <Kontonummer>39050</Kontonummer>
    </KontenKontenWebservice>
    <KontenKontenWebservice>
        <Kontoname>Another AG</Kontoname>
        <Kontonummer>3453</Kontonummer>
        <Strasse>Musterstraße</Strasse>
    </KontenKontenWebservice>
</MESOWebservice>
```

*Abbildung 13: Aufbau des XML-Dokumentes*

Das Wurzelement „MESOWebservice“ des XML-Dokumentes besteht aus einer Liste sich wiederholender XML-Elemente mit dem Namen „KontenWebservice“.

```
@Root(name = "MESOWebservice", strict = false)
data class KontoList(
    @field:ElementList(name = "KontenWebservice", inline = true)
    var kontenList: List<Konto>? = null
)
```

*Abbildung 14: Aufbau KontoList - Entitätsklasse*

Der Klassenname der „KontoList“-Klasse benötigt die Annotation @Root(name = "MESOWebservice"). Dadurch wird der Name des Wurzel-Elements des zu verarbeitenden XML-Dokumentes festgelegt. In der Klasse „KontoList“ wird eine Liste „kontenList“ definiert, welche auf Objekte der Klasse „Konto“ beschränkt ist. Diese Liste wird mit der Annotation @field:ElementList(name = "KontenWebservice") annotiert. Diese Annotation beschreibt, dass alle mit dem Namen „KontenWebservice“ versehenen Elemente in dieser Liste gespeichert werden.

```
@Root(name = "KontenWebservice", strict = false)
data class Konto(
    @field:Element(name = "Kontonummer")
    var kNumber: String? = null,
    @field:Element(name = "Kontoname", required = false)
    var kName: String? = null,
    @field:Element(name = "Strasse", required = false)
    var kStreet: String? = null,
)
```

*Abbildung 15: Ausschnitt der Konto Entitäts-Klasse*

Diese Entitäts-Klasse „Konto“ beschreibt den Aufbau des sich in dem XML-Dokument wiederholenden Elementes „KontenWebservice“ und dessen zugehörige Kinderelemente. Dafür wird diese Klasse mit der @Root-Annotation(name=„KontenWebservice“) annotiert. Die Variablen werden mit der Annotation @field:Element(name = "Kontoname") versehen. Dadurch wird die Variable mit dem Wert des genannten Elementes gefüllt.

### 5.2.3 Zugriff mit Retrofit

Retrofit ist eine Opensource Library. Retrofit ist ein REST-Client, ein REST-Client dient dafür Daten aus einem REST-Webservice abzufragen. Retrofit wandelt die HTTP-API in ein Java Interface um, womit ein Request an den Webserver durchgeführt werden kann.

Die Entscheidung Retrofit zu verwenden beruht darauf, dass diese Library den zuvor genannten Simplexml-Converter verwenden kann. Damit ist die automatische Umwandlung des XML-Responses in Kotlin Objekte möglich. Folgend können händische Eingriffe für das Parsen vermieden werden.

Für die Verwendung der Library ist es notwendig, diese in der build.gradle unter dependencies hinzuzufügen. Uwe Post erklärt, dass eine Dependency eine Library ist, welche für den Build-Prozess der Anwendung erforderlich ist und auf der Festplatte gespeichert wird. (vgl. Post 2019, S. 107)

```
dependencies {  
    implementation 'com.squareup.retrofit2:retrofit:2.3.0'  
    implementation 'com.squareup.retrofit2:converter-simplexml:2.0.0-beta3'  
}
```

Die Anwendung benötigt für den Internetzugriff, welcher bei einer Webservicesanfrage benötigt wird, eine Berechtigung. Diese Berechtigung wird in der AndroidManifest.xml mit folgender Zeile gesetzt:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Der Autor zeigt auf, dass Retrofit aus einem Interface, welches die REST-Abfragen und deren Rückgabewerte definiert, und Datenklassen, welche für die Speicherung von Rückgabewerten verwendet werden, eine Implementierung erzeugt. (vgl. Post 2019, S. 384)

```
interface WebserviceApi {  
    @GET("/ewlservice/export?Company=300M&Type=1&Vorlage=KontenWebservice&Key=FILTER-WSKonten")  
    fun getKontoList(@Query("Password") pw: String, @Query("User") user: String): Call<KontoList>  
  
    @GET("/ewlservice/export?Company=300M&Type=1&Vorlage=KontenWebservice")  
    fun getKonto(@Query("Password") pw: String, @Query("User") user: String, @Query("Key") kontoNummer: String): Call<KontoList>  
  
    @GET("/ewlservice/export?Company=300M&Type=7&Vorlage=KontakteWebservice&Key=FILTER-WSKontakte")  
    fun getKontakteList(@Query("Password") pw: String, @Query("User") user: String): Call<KontakteList>  
  
    @GET("/ewlservice/export?Company=300M&Type=7&Vorlage=KontakteWebservice&Key=FILTER-WSKontakte")  
    fun getKontakt(@Query("Password") pw: String, @Query("User") user: String, @Query("Key") kontoNummer: String): Call<KontakteList>  
  
    object Factory {  
        fun getApi(baseURL: String): WebserviceApi {  
            val retrofit = Retrofit.Builder()  
                .baseUrl(baseURL)
```

```

        .client(HttpClient.getOkHttpClient())
        .addConverterFactory(SimpleXmlConverterFactory.create())
        .build()
    return retrofit.create(WebserviceApi.class.java)
}
}
}

```

In dem Interface sind die Funktionen getKontakt, getkontakteList, getKonto und getKontoList definiert. Die Funktionen des Interfaces werden mit der Annotation @GET markiert, welche den relativen Teil der URL zu dem Webservice beschreibt.

Für die Authentifizierung bei dem Webservice sind Passwörter und Benutzernamen notwendig, diese werden an der URL angehängt. Deswegen ist es notwendig die URL bei dem Abruf der Funktion mit den eingegebenen Parametern zu ändern. Dies geschieht durch das Verwenden der Annotation @Query. Dadurch wird die URL durch den in der Klammer stehenden Wert, einem „=“ und dem eingegebenen Parameterwert ergänzt. Der Rückgabewert der in dem Interface definierten Funktionen sind die im vorigen Kapitel erstellten Entitäten: „kontakteList“ und „KontoList“.

In dem Interface befindet sich die Klasse „Factory“. Diese beinhaltet eine Funktion, welche eine Instanz von Retrofit initialisiert und das Interface implementiert. Die Funktion besitzt einen Parameter „baseUrl“ welche die Basis-URL für den Webservice-Zugriff festlegt. Des Weiteren wird mit der Funktion .client der HTTP-Client festgelegt, da dieser eine selbst geschriebene Funktion für die maximale Dauer eines HTTP-Requests beinhaltet.

Der zuvor erwähnte Simplexml Converter wird mit der Methode .addConverterFactory(SimpleXmlConverterFactory.create()) festgelegt.

Für den Aufruf wird zunächst eine Variable erstellt, welche die Retrofit-Instanz beinhaltet. Über diese kann durch die in dem zuvor erstellten Interface definierte Funktion getKontoListe() eine Variable vom Typ Call<KontoList> erstellt werden.

```

val kontoService = WebserviceApi.Factory.getApi(baseUrl)
val call = kontoService.getKontoList(userPW,userName)

```

Dieser Aufruf kann asynchron mit der Funktion enqueue oder synchron mit der Funktion execute ausgeführt werden. Bei einem synchronen Call läuft dieser auf dem Haupt-Thread und blockiert die Benutzeroberfläche. Um das Blockieren der Benutzeroberfläche zu verhindern wurde entschieden die asynchrone Variante enqueue zu verwenden. Bei dem Aufruf der Methode wird in dem Methodenkopf ein Objekt von dem Typ Callback<KontoList> erstellt, notwendig ist hierbei die beiden zu implementierenden Methoden onResponse und onFailure anzupassen.

```

call.enqueue(object : Callback<KontoList> {
    override fun onResponse(call: Call<KontoList>, response: Response<KontoList>) {
        var responseKontoList = response.body()?.kontenList
        responseKontoList = responseKontoList?.sortedWith(compareBy({ it.kName }))
        if (responseKontoList != null && response.isSuccessful) {
            onFinishListener.onfinished(responseKontoList, FinishCode.finishedOnWeb)
        } else {
            tryLoadingFromFile(onFinishListener)
        }
    }
    override fun onFailure(call: Call<KontoList>, t: Throwable) {
        tryLoadingFromFile(onFinishListener)
    }
})

```

OnResponse liefert bei einem erfolgreichen Webservice-Call einen Response, dieser beinhaltet die Methode body(), welche wiederum die in den Entität-Klasse definierte gefüllte Objekt-Liste kontenList beinhaltet. In dem oberen Ausschnitt wird diese Liste sortiert und dann über den Listener onFinishListener dem Presenter übermittelt. Der Listener wurde dafür erstellt, um dem Presenter mitteilen zu können, dass der asynchrone Call erfolgreich beendet wurde. Wenn der Webservice-Call fehlschlägt wird die Methode onFailure aufgerufen, was in diesem Fall dazu führt, dass eine Methode zum Laden der lokalen Dateien aufgerufen wird.

## 5.3 Persistenz der Daten

Für die Offline-Fähigkeit der Daten ist es notwendig die Kunden- oder Kontaktdaten im Android-Filesystem zu speichern. Da eine Aktualität der Daten sehr wichtig ist, werden diese nach jedem Abruf einer Kunden- oder Kontaktliste aus dem Webservice in einem XML-Dokument gespeichert. Diese persistenten Daten können dann bei fehlender Internetverbindung geladen werden. In den folgenden Kapiteln wird die genaue Vorgangsweise näher erklärt.

### 5.3.1 Speichern von Objekten in einer XML-Datei

Nach einem Webservice-Call steht eine Objekte-Liste aller Konten oder Kontakte bereit. Um diese zu speichern werden die Kotlin-Objekte in XML-Elemente umgewandelt, welche als XML-Datei im Android-Filesystem gespeichert werden. Es wurde entschieden die Objekte aus dem Webservice in XML umzuwandeln und zu speichern. Anstatt den kompletten XML-Response des Webservices zu speichern. Ein Grund dafür ist, dass durch die Speicherung des vollständigen Responses nicht benötigte Daten gespeichert werden würden, was eine deutlich größere Datengröße verursacht. Auch aus Datenschutzrechtlichen Gründen sollen nur benötigte Informationen gespeichert werden. Ferner können die Entitäts-Klassen, aus dem später erklärten Laden der Objekte aus dem XML-Dokument wiederverwendet werden.

Die Objekte aus der Retrofit-Antwort besitzen den gleichen Aufbau wie die Entitäten und nicht benötigte XML-Elemente aus dem Webservice-Response wurden bereits durch den Converter entfernt, nun müssen jene Daten in einem XML-Dokument gespeichert werden.

Der Autor zeigt auf, dass zum Schreiben von Dokumenten ein XMLStreamWriter geeignet ist. Dies ist ein Interface mit diversen Methoden zum Erstellen eines XML-Dokumentes. Vorteile des zur Java Stream API (StAX) gehörende XMLStreamWriter ist die effiziente Verarbeitung und Elemente direkt schreibt und

nicht zuerst als Objekte in einem XML-Baum erzeugt. (vgl. Inden 2016, S. 48) Ein bekräftigender Aspekt ist auch, dass dieses Interface im Android-Framework zur Verfügung steht.

Für die Verwendung des XMLStreamWriter wird zuerst eine XMLOutputFactory erstellt, welche einen XMLStreamWriter erzeugen kann.

```
val xmlStreamWriter = XMLOutputFactory.newInstance().createXMLStreamWriter(writer)
```

Dafür wird ein OutputStreamWriter benötigt. Dieser wird aus dem EncryptedFile mit folgender Zeile erstellt.

```
val writer = encryptedFile.openFileOutput().writer(charset =Charsets.UTF_8)
```

Danach wird die XML-Datei erstellt, der Code dafür schaut wie folgt aus.

```
xmlStreamWriter.writeStartDocument()  
xmlStreamWriter.writeStartElement("MESOWebService")  
for (konto in listToSave) {  
    xmlStreamWriter.writeStartElement("KontenWebservice")  
    if (konto.kNumber != null) {  
        xmlStreamWriter.writeStartElement("Kontonummer")  
        xmlStreamWriter.writeCharacters(konto.kNumber)  
        xmlStreamWriter.writeEndElement()  
    }  
    if (konto.kName != null) {  
        xmlStreamWriter.writeStartElement("Kontoname")  
        xmlStreamWriter.writeCharacters(konto.kName)  
        xmlStreamWriter.writeEndElement()  
    }  
    ....  
    if (konto.kNote != null) {  
        xmlStreamWriter.writeStartElement("Notiz")  
        xmlStreamWriter.writeCharacters(konto.kNote)  
        xmlStreamWriter.writeEndElement()  
    }  
    xmlStreamWriter.writeEndElement()  
}  
xmlStreamWriter.writeEndElement()  
xmlStreamWriter.writeEndDocument()
```

Mit writeStartDocument() und writeEndDocument() wird das Dokument erstellt und geschlossen. Für das Erstellen eines Elementes wird die Funktion writeStartElement(String) verwendet, welche darauffolgend mit writeCharacters() befüllt wird. Danach muss jedes XML-Element mit writeEndElement() geschlossen werden. In der Foreach-Schleife wird hierbei über alle Objekte der Input-Liste aus dem Retrofit-Response iteriert. Um eine Wiederholung von nicht benötigtem Start- und Endtag zu vermeiden, prüft eine If-Abfrage ob eine Instanzvariable ungleich null ist. Dadurch werden leere Elemente, welche nicht benötigten Start- und Endtags beinhalten, vermieden.

### 5.3.2 Laden von Elementen aus einer XML-Datei

Für das Laden des lokalen XML-Files ist es notwendig, die Elemente des XML-Dokumentes in Objekte umzuwandeln. Dafür wird der bereits erwähnte Converter simplexml-converter verwendet. Für diese Umwandlung wird die Methode read der Klasse Persister verwendet. Diese führt die Umwandlung unter Angabe der Ziel-Entität-Klasse und den FileInputStream des XML-Dokumentes durch und kann danach im weiteren Programm Verlauf verwendet werden.

```
val kontoList = Persister().read(KontoList::class.java, fileInputStream)
```

## 5.4 Gewährleistung der Sicherheit

Um die Anforderung der Datensicherheit zu erfüllen, werden diese verschlüsselt, und die Kommunikation mit dem Webservice abgesichert.

### 5.4.1 Verschlüsselung der lokalen Daten

#### Sicheres Speichern und Laden des Webservice-Passwortes

Das Passwort für die Webservice-Verbindung wird standardmäßig als Klartext in den SharedPreferences gespeichert und ist damit eine offene Sicherheitslücke. Eine Möglichkeit das Passwort sicher zu speichern wäre, das Passwort in einen Hash zu verwandeln und diesen bei der Webservice-Verbindung zu übergeben. Der Autor Pragati Ogal erläutert den daraus resultierenden Vorteil, dass es für einen Angreifer nicht möglich wäre das ursprüngliche Passwort aus einem Hash herzuleiten. (vgl. Pragati 2013, Position 1871) Das Hashen ist jedoch nicht möglich, da der bereitgestellte Webservice keine Hash-Werte verarbeiten kann. Um dennoch die Daten sicher zu speichern wird das Passwort verschlüsselt gespeichert.

Es wurde entschieden den AndroidKeyStore als KeyStore-Provider zu verwenden. Mit diesem können Schlüssel mit einem Alias gespeichert und geladen werden. Mit folgender Zeile wird geprüft, ob sich ein passender Schlüssel bereits im KeyStore befindet.

```
val keyStore = KeyStore.getInstance("AndroidKeyStore")
    keyStore.load(null)
    if (!keyStore.containsAlias(keyAlias)) {
        ...
    }
```

Wenn noch kein passender Schlüssel vorhanden ist, wird ein zu dem gewählten Verschlüsselungsalgorithmus (AES) passender Schlüssel, wie folgend zu sehen, erstellt. Bei der Schlüsselerstellung wird auch der oben gewählte Blockmodus (CBC) und das Padding festgelegt.

```
val keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES, "AndroidKeyStore")
val keyGenParameterSpec = KeyGenParameterSpec.Builder(keyAlias, KeyProperties.PURPOSE_ENCRYPT or KeyProperties.PURPOSE_DECRYPT)
    .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
    .build()
keyGenerator.init(keyGenParameterSpec)
```

Ein gespeicherter Schlüssel kann nach der Erstellung bei Gebrauch, wie folgend zu sehen, aus dem KeyStore geladen werden.

```
keyStore.getKey(keyAlias, null)
```

Mit dem Schlüssel und einem Klartext, umgewandelt in einen ByteArray, kann nun die Verschlüsselung durchgeführt werden.

```
fun encrypt(plainText: ByteArray, key: Key): Pair<ByteArray, ByteArray>? {
    val cipher = Cipher.getInstance("AES/CBC/PKCS7Padding")
    cipher.init(Cipher.ENCRYPT_MODE, key)
```

```

    val cipherText = cipher.doFinal(plainText)
    return Pair(cipherText, cipher.iv)
}

```

Bei dem Aufbau der Funktion wurde entschieden, den Datentyp „Pair“ für den Rückgabewert zu verwenden. Da wie zuvor erwähnt CBC als Blockmodus gewählt wurde, wird ein Initialisierungsvektor für die erste Block-XOR-Verknüpfung verwendet. Dieser Initialisierungsvektor und der verschlüsselte Text wird durch diesen „Pair“ zurückgegeben und gespeichert. Beide sind für die spätere Entschlüsselung notwendig.

```

fun decrypt(cipherText: ByteArray, key: Key, iv: IvParameterSpec): ByteArray {
    val cipher = Cipher.getInstance(cipherAlgorithm)
    try {
        cipher.init(Cipher.DECRYPT_MODE, key, iv)
    } catch (e: InvalidKeyException) {
        return cipherText
    } catch (e: InvalidAlgorithmParameterException) {
        return cipherText
    }
    return cipher.doFinal(cipherText)
}

```

## Verschlüsselung der XML-Dateien

Die Verschlüsselung der XML-Dateien wurde mit der Android Security Library durchgeführt. Die Entscheidung diese zu verwenden beruht darauf, dass diese eine einfache, aber sichere Implementierung von Sicherheitspraktiken zum Schreiben und Lesen eines Files bietet. Mit folgendem Code wird ein bereits existierendes, verschlüsseltes File geöffnet, ansonsten kann damit ein neues verschlüsseltes File erstellt werden.

```

val masterKeyAlias = MasterKeys.getOrCreate(MasterKeys.AES256_GCM_SPEC)
val encFile = File(context.filesDir, KONTO_LIST_FILE_NAME)
    val encryptedFile = EncryptedFile.Builder(
        encFile,
        context,
        masterKeyAlias,
        EncryptedFile.FileEncryptionScheme.AES256_GCM_HKDF_4KB
    ).build()

```

Dieses File kann nun durch folgende Zeilen mit verschlüsselten Daten beschrieben oder dessen Daten entschlüsselt gelesen werden.

```

encryptedFile.openFileInput() //Lesen des verschlüsselten Files
encryptedFile.openFileOutput() //Schreiben eines verschlüsselten Files

```

## 5.4.2 Sicherer Webservice-Zugriff

Die Kommunikation zwischen Anwendung und Webserver soll nur mit „https“ geschehen. Dies dient zur Verschlüsselung der zu transferierenden Daten und der URL-Parameter. Dafür werden Webserver-URL-Eingaben des Benutzers nach dem „https-Schema“ geprüft. Die Anwendung baut, wenn keine URL mit „https-Schema“ vorliegt, eine URL aus den Eingaben mit diesem Schema.

Für die Prüfung des sicheren Datenaustausches wurde die Software Wireshark verwendet. Die Anwendung wurde auf einem Emulator ausgeführt und dessen Netzwerk-Verkehr aufgezeichnet.

Der Datenaustausch zwischen Client und Server findet wie folgt statt.

Zuerst wird das „Client-Hello“ und „Server Hello“ übertragen. Diese dienen zum Starten der Session zwischen Client und Server.

10.0.0.20	83.164.140.68	TLSv1.2	313	Client Hello
83.164.140.68	10.0.0.20	TLSv1.2	1462	Server Hello

Als nächstes authentisiert sich der Webserver gegenüber dem Client mit einer Übertragung seines Zertifikates.

83.164.140.68	10.0.0.20	TLSv1.2	1055	Certificate, Server Hello Done
---------------	-----------	---------	------	--------------------------------

Der Server sendet keinen „CertificateRequest“, eine Anforderung zur Authentifizierung des Clients. Somit sendet der Client kein Zertifikat. Der Client sendet weiters dem Server seinen Schlüssel und teilt dem Server mit „Change Cipher Spec“ mit, dass die zukünftigen Daten verschlüsselt sind.

10.0.0.20	83.164.140.68	TLSv1.2	372	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
-----------	---------------	---------	-----	--

Der Server versendet darauffolgend ebenfalls ein „Change Cipher Spec“ und die „Encrypted Handshake Message“, womit bestätigt wird, dass der Schlüsselaustausch erfolgte und die Authentifizierung abgeschlossen ist.

83.164.140.68	10.0.0.20	TLSv1.2	280	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
---------------	-----------	---------	-----	---

Über „Application Data“ werden die Daten nun verschlüsselt an den Client übertragen.

10.0.0.20	83.164.140.68	TLSv1.2	306	Application Data
-----------	---------------	---------	-----	------------------

Dieses Packet wurde untersucht. Hierbei wurde festgestellt, dass die übertragenen Daten „Encrypted Application Data“ tatsächlich verschlüsselt übertragen wurden.

Transport Layer Security  
▼ TLSv1.2 Record Layer: Application Data Protocol: Application Data  
Content Type: Application Data (23)  
Version: TLS 1.2 (0x0303)  
Length: 247  
Encrypted Application Data: 00000000000000001936d7f9b902a0425327d1f27753b8dd6...

Abbildung 16: Verschlüsselte Daten

# 6 Zusammenfassung

## 6.1 Resultat

Trotz diverser Grundkenntnisse, welche im Unterricht der HTL Traun erworben wurden, war es eine große Herausforderung, eine komplexe Anwendung zu programmieren. Durch die Kombination einer neuen Programmiersprache und der neuen Android-Umgebung war die Entwicklung mit vielen Herausforderungen verbunden. Diese konnten durch Recherchen, Ausprobieren und Verwenden der Android-Dokumentation behoben werden. Es wurde darauf geachtet, die Funktionen der Detail-Ansichten möglichst praktisch zu halten, um das Ziel des Arbeitgebers, die Anwendung als innovatives Hilfsmittel zu verwenden, erfüllen zu können.

### 6.1.1 Errungenschaften

Die Anwendung bietet einem Mitarbeiter eines Unternehmens, welches ein WinLine ERP-System verwendet, folgende Funktionen:

1. Zugriff auf Kunden- und Ansprechpartner, jederzeit und ortsunabhängig, durch die Offline-Zwischenabspeicherung.
2. Verbesserung der Kommunikation mit dem Kunden oder dem Ansprechpartner durch Ausführung der Funktionen (Anrufen, SMS, Adresse mit Google Maps öffnen, URL öffnen, E-Mail schreiben) mit den in dem ERP-System hinterlegten Daten.
3. Gewährleistung der Sicherheit durch Verschlüsselung der lokalen Daten und der Webservice-Kommunikation.
4. Einstellungsmenü zum Festlegen der Verbindungseigenschaften zu dem Webservice, und Anpassbarkeit des Timeouts.

### 6.1.2 Herausforderungen

Es wäre an dieser Stelle zu umfangreich, sämtliche aufgetretene Probleme und Schwierigkeiten zu beschreiben. Im Folgenden werden daher nur die wesentlichsten Herausforderungen beschrieben.

Die größten Herausforderungen entstanden aufgrund fehlender Kenntnisse über das Android-SDK. Dadurch waren Implementationen, wie zum Beispiel die Listenansicht inklusive der Suche oder das richtige Anwenden der Lifecycle-Methoden, sehr zeitaufwendig. Dies verbesserte sich jedoch mit zunehmender Erfahrung im Verlauf des Projektes.

Auch war es eine Herausforderung erstmalig eine komplexe Anwendung zu programmieren. Eine erhebliche Erleichterung, lag diesbezüglich in der Verwendung des zuvor genannten Architekturmusters.

Bei der Erstellung des Layouts kam des Öfteren vor, dass Daten nicht vollständig angezeigt wurden, dies musste aufwendig mit Layout-Attributen behoben werden.

Die Tatsache, dass die Sprache Kotlin erst im Laufe des Projekts, im Selbststudium, angeeignet wurde erschwerte die Programmierung.

Beispielsweise musste bei der Programmierung der Suche, ein Array geleert werden. Dies ist jedoch nur mit einem Mutable-Array, ein Array, welches auch leer sein darf, möglich. Aus diesem Grunde musste die Anwendung umgeschrieben werden. Auch war es gewöhnungsbedürftig die Null-Sicherheit korrekt zu implementieren, was jedoch nach der Eingewöhnungszeit kein Problem mehr war.

Eine weitere, mit der Programmiersprache Kotlin zusammenhängende, Schwierigkeit lag in der Verwendung der Library Retrofit, welche für den Webservicezugriff zuständig ist. Die Herausforderung lag in der Kompatibilität mit Kotlin. Dadurch musste die Vorgehensweise, von der in der Literatur beziehungsweise

in der offiziellen Retrofit-Dokumentation beschriebenen Vorgehensweise, abweichen. Dies konnte aufgrund von Online-Recherchen bewerkstelligt werden.

Zudem entstand durch das Verwenden von asynchronen Funktionen, welche nicht auf den Haupt-Threads durchgeführt werden, eine Notwendigkeit die Resultate zu überwachen. Es würde für diesen Zweck externe Libraries geben, was jedoch die Komplexität der Anwendung erhöhen würde. Schlussendlich konnte dieses Problem mit einem einfachen Listener gelöst werden.

Eine weitere Schwierigkeit lag darin, die Implementierung von Funktionen als abgeschlossen zu sehen, weil es immer noch Kleinigkeiten gab, welche noch verbessert werden könnten. Diese hätten jedoch keine große Auswirkung auf den erfolgreichen Projektabschluss.

## 6.2 Ausblick

In einem zukünftigen Projekt könnte die Synchronisation der Daten verbessert werden. Bisher werden bei jedem „Detail-Aufruf“ und „Ansprechpartner-Listeansicht-Aufruf“ Daten aus dem Webservice geladen. Diese könnten bereits lokal persistent gespeichert und aktuell sein, was den Download nicht notwendig machen würde. Durch ein besseres Synchronisationssystem könnte ein geringerer Datenverbrauch entstehen.

## 7 Quellen- / Literaturverzeichnis

Inden M.; *Der Java-Profi: Persistenzlösungen und REST-Services*, 1. Auflage – Heidelberg: dpunkt.verlag, 2016

ISBN: 978-3-86490-374-8

Richter E.; *Android-Apps programmieren - Praxiseinstieg mit Android Studio*, 2. Auflage – Bonn: mitp, 2019

ISBN: 978-3-95845-754-6

Post U.; *Android-Apps entwickeln für Einsteiger*, 8. Auflage – Bonn: Rheinwerk Computing, 2019

ISBN: 978-3-8362-6928-5

Pragati O.; *Android Application Security Essentials*, 1. Auflage – Birmingham: Packt, 2013

ISBN: 978-1-84951-560-3

Staudemeyer J.; *Android mit Kotlin - kurz&gut*, 1. Auflage – Sebastopol: O'Reilly, 2018

ISBN: 978-3-96009-038-0

Tilkov S.; Eigenbrodt M.; Schreier S.; Wolf O.: *REST und http – Entwicklung und Integration nach dem Architekturstil des Web*, 3. Auflage – Heidelberg: Dpunkt.verlag, 2015

ISBN: 978-3-86490-120-1

Theist T.; *Einstieg in Kotlin – Apps entwickeln mit Android Studio*, 1.Auflage – Bonn: Rheinwerk Computing, 2019

ISBN: 978-3-8362-6872-1

# **8 Abbildungsverzeichnis**

## **Abbildungsverzeichnis**

Abbildung 1: Ausschnitt aus dem bereitgestellten XML-Dokument des Webservices .....	7
Abbildung 2: Kunden-Listenansicht .....	11
Abbildung 3: Gefilterte Listen-Ansicht .....	11
Abbildung 4: Ausgeklapptes Einstellungsmenü .....	12
Abbildung 5: Fehlermeldung.....	12
Abbildung 6: Kundendetailansicht mit Funktionen.....	13
Abbildung 7: Ansprechpartner Listenansicht .....	14
Abbildung 8: Ansprechpartner Detailansicht.....	14
Abbildung 9: Einstellungsmenü für die Verbindungseigenschaften .....	15
Abbildung 10: Erfolgreicher Wechsel.....	17
Abbildung 11: Falsche Anzeige .....	17
Abbildung 12: UML-Diagramme der Entitäts-Klassen .....	24
Abbildung 13: Aufbau des XML-Dokumentes .....	25
Abbildung 14: Aufbau KontoList - Entitätsklasse .....	25
Abbildung 15: Ausschnitt der Konto Entitäts-Klasse.....	26
Abbildung 16: Verschlüsselte Daten .....	34

# 9 Begleitprotokolle

## Besprechungsprotokoll

### Diplomarbeit - Besprechungsprotokoll

HTL Traun - Höhere Lehranstalt für Informationstechnologie - Fachschule für Informationstechnik



<b>Thema</b> (übergeordnetes Projekt): ERP-Connect <b>Teammitglieder</b>	Finn Dorninger	<b>Jahrgang:</b> 5BHIT <b>Schuljahr:</b> 2019/2020 <b>Betreuer/in:</b> DI Alois Gaisberger <b>Kooperationspartner/in:</b> DI Peter Wurm, Fa. SYSc
---	----------------	--

Datum der Besprechung	Teilnehmer/innen der Besprechung	Themen	Termin zur Erledigung	Vereinbarungen	optional: Signatur Kandidat/in
08/05/2019	Alois Gaisberger Finn Dorninger	Projektstart-Besprechung, Kooperationspartner, Anforderungen	30/05/2019	Terminvereinbarung mit Kooperationspartner	
05/06/2019	Peter Wurm Alois Gaisberger Finn Dorninger	Ziele des Projektes, Möglichkeit eines Praktikums, Kontaktmöglichkeiten,	10/06/2019	Erstellung Projektantrag	
26/06/2019	Alois Gaisberger Finn Dorninger	Vorgehensweise, Speichern der Daten auf Git, Präsentationen, Korrekturlesen.	Anfang Schulbeginn	Pflichtenheft, Projektplan, Zeitplanung, grobe Planung des Inhaltsverzeichnisses.	
29/08/2019	Peter Wurm Finn Dorninger	Review Pflichtenheft, Kontaktmöglichkeiten			
05/09/2019	Thomas Rafetseder Finn Dorninger	Projektstand-Updates per Telefon/Treffen bei dem Unternehmen.	Alle 3 Wochen Alle 6 Wochen	Prüfen und organisieren Kontaktmöglichkeiten	
10/09/2019	Alois Gaisberger Finn Dorninger	Besprechen Projektzwischenstand	20/09/2019	Aufteilung Projekt in kleinere Teile	
11/10/2019	Alois Gaisberger Finn Dorninger	Review des Projektstandes, Präsentationen, Vorgehensweise bei dem Verfassen der Diplomarbeit.	27/09/2019	Projektordner freigeben, GIT-Zugriff, Kapitelübersicht, Präsentation.	
26/11/2019	Alois Gaisberger Finn Dorninger	Projektstand, Feedback - Prototyp, Vorgehensweise	05/12/2019	Hinzufügen der automatischen Synchronisation	
05/12/2019	Alois Gaisberger Finn Dorninger	Projektstand und Präsentation, Vorgehensweise bei der Erstellung der	11/12/2019	Fertigstellung Präsentation	
06/12/2019	Peter Wurm Finn Dorninger	Vorstellung des Prototypen, Feedback	Nach Fertigstellung der Arbeit	Handbuch zur Installation der Android-App	
19/12/2019	Alois Gaisberger Finn Dorninger	Vorgehensweise schriftliche Diplomarbeit, Feedback Präsentation			
09/01/2020	Alois Gaisberger Finn Dorninger	Feedback Präsentation, Änderung des Zeitplanes	/	Änderung des Zeitplanes	
16/01/2020	Alois Gaisberger Finn Dorninger	Feedback Präsentation, Schriftlicher Teil der Diplomarbeit	19/01/2020	Abgabe erste Version	
12/02/2020	Alois Gaisberger Finn Dorninger	Projektzwischenstand, Schriftlicher Teil der Diplomarbeit	27/02/2020	Abgabe vollständig ausformulierte Diplomarbeit	

# Begleitprotokoll

Name der Schüler/in und Klasse: Finn Dorninger, 5BHIT  
 Projektthema: ERP-Connect  
 Individuelle Themenstellung: Programmierung, Projektmanagement, Testen

Datum	Tätigkeit, Hilfsmittel, Hilfestellungen	Dauer
	SUMME	377
	Projektphase: Vorstudie	0
2019.06.05	<b>Besprechung mit Auftraggeber:</b> Hr. Wurm, Hr. Gaisberger, Finn Dorninger <b>Klärung:</b> Projektinhalt, Kommunikation mit dem Unternehmen	2
2019.06.12	<b>Besprechung mit Betreuer - DA Antrag:</b> Hr. Gaisberger, Finn Dorninger Projektinhalt klären, DA-Antrag vorbereiten.	1
2019.06.19	DA-Antrag erstellen.	3
	Projektphasen: Definition + Planung + Analyse	0
2019.06.26	<b>Besprechung mit Betreuer - Projektstart:</b> Prof. Gaisberger, Finn Dorninger. <b>Klärung:</b> Projektdokumentationen, Projektablauf, Beurteilung.	1
2019.07.01	Einarbeiten in die Entwicklungsumgebung: (mit Google Codelabs)	3
2019.07.02	Ebenso	3
2019.07.03	Ebenso	3
2019.07.04	Ebenso	3
2019.07.05	Einlesen in die Programmiersprache: Kotlin (Basics, Null-Safety, Android mit Kotlin)	3
2019.07.06	Ebenso	3
2019.07.07	Ebenso	3
2019.08.16	<b>Erstellung:</b> Pflichtenheft	3
2019.08.19	<b>Erstellung:</b> Projekthandbuch(Projektumfeldanalyse, Objektstrukturplan)	2
2019.08.20	<b>Erstellung:</b> Projekthandbuch(Projektstrukturplan, Datenschutzaspekte)	5
2019.08.21	Installation WinLine, Export Webservice	2
2019.08.22	<b>Einarbeitung:</b> Android Studio: Layout, Lifecycle, Navigation zwischen Activitys, Shared Preferences	4
2019.08.23	<b>Einarbeitung:</b> Android Studio: Click-Listener, Ressourcen, Data Binding	3

2019.08.26	<b>Einarbeitung:</b> Erstellung Einstellungsmenü mit Fragmenten (Schwierigkeiten mit NavGraph)	7
2019.08.27	<b>Einarbeitung:</b> Speicherung Einstellungsdaten (Versuch Anwendung Shared Preferences, Schwierigkeiten mit Context), hat dann aber noch funktioniert.	7
2019.08.28	<b>Einarbeitung:</b> Erstellung Einstellungsmenü: Speicherung Einstellungsdaten in Realm-Datenbank (Schwierigkeiten mit Context)	7
2019.08.29	<b>Besprechung mit Auftragsgeber:</b> Peter Wurm, Finn Dorninger. Review Pflichtenheft.	1
2019.08.30	<b>Minimalsystem:</b> Zugriff auf Webservice API über AsyncTask (Sehr unübersichtlich, zu viel Aufwand)	6
2019.08.30	<b>Erstellung:</b> Pflichtenheft fertiggestellt.	1
2019.09.02	<b>Minimalsystem:</b> Zugriff Webservice API mit Retrofit (Probleme mit POJOs → Kotlin Annotations) Vergessen von @ElementList.	7
2019.09.03	<b>Minimalsystem:</b> Zugriff Webservice API mit Retrofit, Darstellung Daten (Listview zu langsam, Recyclerview nicht geschafft)	7
2019.09.04	<b>Minimalsystem:</b> Speichern der Daten in Realm-Datenbank	7
2019.09.05	<b>Minimalsystem:</b> Darstellung der Daten (Schwierigkeiten mit der Darstellung von Datenbank-Daten) -> Grund: Kein Architekturmuster!	7
2019.09.06	<b>Besprechung mit Auftragsgeber:</b> Hr. Rafetseder, Finn Dorninger Klärung: Reviews mit Auftragsgeber, Kontakt	1
2019.09.06	<b>Erstellung:</b> Meilensteinliste, Projektrisikoanalyse, Qualitätsmanagement	4
2019.09.10	<b>Gespräch mit Betreuer:</b> Einteilung in kleinere Bereiche. Architekturmuster	1
2019.09.11	<b>Minimalsystem:</b> Architekturmuster festlegen/Recherche: Model View Presenter	4
2019.09.12	<b>Minimalsystem:</b> Projekt an Architekturmuster anpassen	4
2019.09.19	<b>Minimalsystem:</b> Speichern von Objekten in XML-Datei (Probleme mit Context → MVP anpassen)	4
2019.09.20	<b>Minimalsystem:</b> Erstellung von Objekten aus einer XML-Datei	3
2019.09.21	Erstellung Projektplan	6
2019.09.22	Fertigstellung Projektplan	4
2019.09.23	<b>Kontakt mit Auftragsgeber:</b> Kontakte in Webservice	1
2019.09.23	Projektstart abgeschlossen	1
	Meilenstein: Darstellung und Offline-Speicherung der Daten	0

2019.09.27	<b>Webservice-Zugriff:</b> Keine Probleme (Minimalsystem benutzt).	1
2019.09.28	<b>Konten lokal speichern:</b> Probleme: Context benötigt → MVP anpassen. .orEmpty() benutzen um NullpointerException zu vermeiden	4
2019.09.29	<b>Konten lokal speichern:</b> Exception-Handling	3
2019.09.30	<b>Konten lokal speichern:</b> Exception-Handling	3
2019.10.04	<b>Darstellung der Konto-Liste (mit Recyclerview):</b> (Probleme bei der Anwendung → Kotlin benötigt anderen Konstruktor)	5
2019.10.11	<b>Gespräch mit Betreuer:</b> Projektstand, Benötigte Unterlagen	1
2019.10.12	<b>Verknüpfung mit Konto-Details:</b> (Schwierigkeiten bei der Übertragung von Konten zwischen Activitys nur mit Serialisierung möglich → könnte fehlerhaft sein, deswegen neues Konto-Details Model erstellt)	5
2019.10.13	<b>Kontakte-Liste Suche:</b> Probleme bei der Darstellung: Daten werden nach entfernen der Suche nicht dargestellt. Crash wenn während dem Laden der Daten gesucht wurde)	4
2019.10.14	<b>Kontakte-Liste-Suche:</b> Fehler behoben.	2
2019.10.16	<b>Zugriff Webservice:</b> Check nach Internetverbindung hinzugefügt.	1
2019.10.17	<b>Kontakte-Details-Funktionen:</b> Starten von Intents. Probleme mit Google Maps, weil die Adress-Daten im ERP unterschiedlich aufgebaut sind. Bei dem Anruf-Intent musste man alle Stellen die null sind entfernen, sonst gibt es eine falsche Nummer	4
2019.10.18	<b>Kontakt mit Auftragsgeber:</b> Update-Projektstand	1
2019.10.19	<b>Erstellung GANTT</b> mit GanttProject	2
2019.10.25	<b>GUI-Fehler anzeigen</b> mit Snackbars.	5
2019.10.26	<b>Konto-Details Funktionen:</b> Beheben des Fehlers bei Google-Maps Intent Konto-Details: Hinzufügen von Zugriff über Webservice -> Schwierigkeiten mit @Query. Anzeigen von Fehlern mit der Snackbar.	6
2019.11.03	<b>Einstellungsmenü:</b> Es gab Fehler mit der Snackbar, welche kein richtiges Rootview zugewiesen hatte. Außerdem musste das Passwort-Feld in den Einstellungen angepasst werden. Bestehender Fehler: Bei falscher URL-Eingabe stürzt die APP ab.	5
2019.11.04	<b>Einstellungsmenü:</b> Behebung der Fehler durch Check nach valider URL. Hinzugefügt: Verstecken des Recyclerviews wenn keine Daten vorhanden.	3
2019.11.04	<b>Präsentation:</b> Grobe Strukturierung der einzelnen Folie	1
2019.11.06	<b>Darstellung der Konto-Details:</b> Versuch mit Grid-Layout eine Tabelle zu erstellen.	2
2019.11.07	<b>Darstellung der Konto-Details:</b> Grid-Layout, Zu lange Texte werden nicht richtig angezeigt.	2

2019.11.08	<b>Darstellung der Konto-Details:</b> Mit Table-Layout Problem gelöst.	2
2019.11.09	<b>Konten lokal speichern:</b> Hinzufügen von Asynchronität/Neuer Listener. <b>Fehler anzeigen:</b> Snackbar dismiss bei Änderung hinzugefügt. Presenter bei onDestroy null setzen	2
2019.11.10	<b>Bottom-Navigation-View:</b> Keine Probleme, nur Schwierigkeiten mit Layout Löschen der Recyclerview-Daten hinzugefügt (Nach Einstellungsänderungen) Refactoring.	4
Meilenstein	Darstellung und Offline-Speicherung der Kontodaten	0
	Prototyp für das Unternehmen	0
2019.11.10	Bottom-Navigation-View	4
2019.11.10	<b>Webservice Zugriff:</b> Beginn	1
2019.11.11	Verbesserung: Clear des Recyclerview, Hinzufügen eines Timers welcher nach langem Abruf von Webservice-Daten, die lokalen Daten ladet.	2
2019.11.12	<b>Kontakt mit Arbeitsgeber:</b> Update, fragen nach Termin für Vorstellung des Prototyps	1
2019.11.13	Verbesserung: Handler anstatt Timer (Asynchron)	1
2019.11.15	<b>Webservice-Zugriff:</b> Fertigstellung	1
2019.11.16	Darstellung Kontakte-Liste, Bottom-Navigation Menu anpassen	3
2019.11.17	Überarbeitung des Lifecycles (zum Beispiel: Konto-Liste wurde immer neu erstellt, bei dem Wechsel von Kontakten zu Konten)	2
2019.11.18	<b>Darstellung Kontakte-Details und Funktionen</b>	2
2019.11.22	<b>Präsentation:</b> Erstellung	1
2019.11.23	<b>Präsentation:</b> Erstellung	2
2019.11.26	<b>Gespräch mit Betreuer:</b> Projektstand, Feedback, Vorgehensweise Diplomarbeit	1
2019.11.29	<b>Einstellungsmenü:</b> Funktion ob Kontakte bei Start synchronisiert werden sollen. (Logik: Kontakte Presenter überarbeitet)	2
2019.11.29	<b>Präsentation:</b> Überarbeitung der besprochenen Punkte	1
2019.11.30	<b>Einstellungsmenü:</b> Hinzufügen von Timeout-Option. Timeout nun nicht mehr mit Handler sondern direkt mit Timeout-Funktion des OkHttp-Clients. URL-Option (Check ob URL gültig ist). Timeout gespeichert in SharedPreferences.	3
2019.12.03	<b>Präsentation:</b> Überarbeitung	1

2019.12.04	Überarbeitung: Sortierung im Recyclerview + richtige Namenanzeige, Ansprechpartner-Button-Funktion. Einstellungsmenü verbessert: Recyclerview-Clear bei Änderung von Username/URL/Passwort sonst nicht. Funktionen-Kontakte-Details	4
2019.12.05	<b>Gespräch mit Betreuer:</b> Projektstand und Präsentation, Vorgehensweise bei der Erstellung der Enddokumentation	1
2019.12.06	APK erstellen	1
2019.12.06	<b>Gespräch mit Auftraggeber:</b> Vorstellung des Prototyps, Feedback	1
2019.12.07	<b>Fertigstellung Präsentation</b>	1
	Meilenstein: Fertigstellung Prototyp	0
2019.12.11	<b>Zwischenpräsentation Diplomarbeit</b>	1
2019.12.13	<b>WS-Zugriff absichern:</b> HTTP-Client modifiziert, dass er das ungültige Zertifikat nur auf der bereitgestellten IP akzeptiert. Mit Wireshark getestet ob Verbindung tatsächlich mit HTTPS/TLS geschieht.	2
2019.12.14	<b>Lokale Daten verschlüsseln:</b> Verwenden von Encryption-Api. MinSDK musste erhöht werden. Files wurden verschlüsselt. Verschlüsselte Datei muss gelöscht werden. Sonst Verschlüsselung nicht möglich!	4
2020.12.15	<b>Lokale Daten verschlüsseln:</b> AES-Verschlüsselung des in SharedPreferences gespeicherten Passwortes.	3
2020.12.16	<b>Lokale Daten verschlüsseln:</b> AES-Verschlüsselung, Schlüsselspeicherung. Verwenden des Android KeyStores.	3
	Meilenstein: Fertigstellung der App	0
2019.12.16	<b>Schriftlicher Teil der Diplomarbeit:</b> Inhaltsverzeichnis	1
2019.12.19	<b>Gespräch mit Betreuer:</b> Vorgehensweise schriftliche Diplomarbeit, Feedback Präsentation	1
2019.12.22	<b>Schriftlicher Teil der Diplomarbeit:</b> Inhaltsverzeichnis	1
2019.12.28	<b>Schriftlicher Teil der Diplomarbeit:</b> Inhaltsverzeichnis	2
2019.12.29	<b>Fehler behoben, Einstellungsmenü:</b> Durch Programm geänderte Einstellungen wurden nicht übernommen.	2
2019.12.29	<b>Schriftlicher Teil der Diplomarbeit:</b> Inhaltsverzeichnis, Literatur	3
2019.12.30	<b>Schriftlicher Teil der Diplomarbeit:</b> Ausarbeitung Inhaltsverzeichnis, Erklärung App	3
2019.12.01	<b>Schriftlicher Teil der Diplomarbeit:</b> Erklärung der App	2
2020.01.02	<b>Schriftlicher Teil der Diplomarbeit:</b> Erklärung Retrofit-Zugriff	3

2020.01.03	<b>Schriftlicher Teil der Diplomarbeit:</b> Persistenz der Daten, Überarbeitung Retrofit-Zugriff	4
2020.01.03	<b>Fehler beheben/verbessern:</b> Speicherung der Daten: Wechsel zu StAX – XMLStreamWriter – da dieser effizienter ist.	3
2020.01.06	<b>Schriftlicher Teil der Diplomarbeit:</b> Webservice, Objekte aus XML-Dokumente laden	2
2020.01.07	<b>Schriftlicher Teil der Diplomarbeit:</b> Intents	1
2020.01.07	<b>Präsentation:</b> Einbauen des Feedbacks	1
2020.01.08	<b>Gespräch mit Betreuer</b>	1
2020.01.10	<b>Schriftlicher Teil der Diplomarbeit:</b> Konzept für die Speicherung der Daten	3
2020.01.11	<b>Schriftlicher Teil der Diplomarbeit:</b> Erstellung von Entitäten	2
2020.01.13	<b>Präsentation:</b> Einarbeiten des Feedbacks aus dem Gespräch	1
2020.01.16	Gespräch mit Betreuer	1
2020.01.18	<b>Schriftlicher Teil der Diplomarbeit:</b> Verbesserung der bestehenden Punkte	2
2020.02.01	<b>Schriftlicher Teil der Diplomarbeit</b>	2
2020.02.12	Gespräch mit Betreuer	1
2020.02.16	<b>Schriftlicher Teil der Diplomarbeit:</b> Einfache Korrekturen einbauen.	2
2020.02.17	<b>Schriftlicher Teil der Diplomarbeit:</b> Einleitung, Zielsetzung und Aufgabenstellung, Ausgangslage.	3
2020.02.18	<b>Schriftlicher Teil der Diplomarbeit:</b> Konzept für die Programmierung, Konzept für die Speicherung	4
2020.02.19	<b>Schriftlicher Teil der Diplomarbeit:</b> Gewährleistung der Sicherheit, Erstellung einer Android App.	4
2020.02.20	<b>Schriftlicher Teil der Diplomarbeit:</b> Überarbeitung des Kapitels ERP-Connect, Activities	4
2020.02.21	<b>Schriftlicher Teil der Diplomarbeit:</b> Layout, Speichern und Laden von Einstellungen	5
2020.02.22	<b>Schriftlicher Teil der Diplomarbeit:</b> Architekturmuster, Testen der Anwendung,	3
2020.02.23	<b>Schriftlicher Teil der Diplomarbeit:</b> Gewährleistung der Sicherheit	3
2020.02.24	<b>Schriftlicher Teil der Diplomarbeit:</b> Zusammenfassung, Herausforderungen.	2
2020.02.25	<b>Schriftlicher Teil der Diplomarbeit:</b> Fehler ausbessern.	1
2020.02.26	<b>Dokumentation (Kommentieren der Anwendung)</b>	4
2020.02.27	<b>Ebenfalls</b>	2

2020.02.28	<b>Testen der Anwendung:</b> Refactoring um die Anwendung für das Testen zu optimieren, Testen der Models. Wegen Asynchronität von Funktionen Probleme. Mit CountDownLatch gelöst!	5
2020.02.29	<b>Testen der Anwendung:</b> Refactoring um die Anwendung für das Testen zu optimieren. Testen der SharedPreferences, Encryption	4
2020.03.02	<b>Testen der Anwendung:</b> Refactoring, Presenter Testen. Mocking notwendig. Probleme mit Version. Auf Kotlin-Mockito (aus Github) ausgewichen. Anpassen der Mockito-Konfiguration, um einen auftretenden Null-Fehler zu vermeiden.	4
2020.03.03	<b>Testen der Anwendung:</b> Activity-Test. Probleme mit dem Starten von Activities. Verwenden von ActivityTestRules	3
2020.03.05	<b>Dokumentation (Kommentieren der Anwendung)</b>	2
2020.03.06	<b>Schriftlicher Teil der Diplomarbeit:</b> Kapitelübersicht behoben, kleine Änderungen, Erklärung Fragment, Sicherheit.	3
2020.03.07	<b>Schriftlicher Teil der Diplomarbeit:</b> Überarbeitung Kapitel Entitäten, Retrofit, Zitate überarbeitet.	4
2020.03.08	<b>Testen der Anwendung:</b> Testfälle hinzugefügt	2
2020.03.09	<b>Einarbeiten Feedback:</b> Nicht verwendbare Buttons grau. Nicht verwendbare farbig.	3
2020.03.14	<b>Testen der Anwendung:</b> Activity-Tests erweitert – Suche getestet. Hinzufügen von Kontonummer – Eingabe bei Suche sowohl bei KontoList als auch KontakteListe.	3
2020.03.15	<b>Testen der Anwendung:</b> Detail-Utility-Tests	2
2020.03.16	<b>Schriftlicher Teil der Diplomarbeit:</b> Formatierung, Danksagung, Abbildungsverzeichnis, Abstract	2
2020.03.17	<b>Schriftlicher Teil der Diplomarbeit:</b> Formatierung, Abstract	3
2020.03.18	<b>Testen der Anwendung:</b> Testfälle, Fehlermeldung hinzugefügt wenn Konto-Detail-Nummer nicht verfügbar.	3
2020.03.19	<b>Testen der Anwendung:</b> Testfälle für Detail-Activities (Darstellung, Buttons). Fehler beheben (Maps-Button immer aktiv, Absturz bei Ansprechpartner-Button Auswahl), Refactoring, Kommentieren der Anwendung	4
2020.03.19	<b>Abgabe vorbereiten für Auftragsgeber:</b>	2

# 10 Projektdokumente

Es wurden folgende Dokumente im Laufe des Projektes erstellt. Diese werden auf den folgenden Seiten dargestellt.

- Pflichtenheft

ERP\_Connect\_Pflichtenheft.odt

- Besprechungsprotokolle

1\_05\_06\_2019\_Besprechungsprotokoll\_Auftragsgeber\_Vorstudie.odt

2\_26\_06\_2019\_Besprechungsprotokoll\_Startbesprechung.odt

3\_29\_08\_2019\_Besprechungsprotokoll\_Projektstart\_Auftragsgeber.odt

4\_05\_09\_2019\_Besprechungsprotokoll\_Auftragsgeber\_Controling.odt

5\_10\_09\_2019\_Besprechungsprotokoll\_Zwischenstand.odt

6\_11\_10\_2019\_Besprechungsprotokoll\_Zwischenstand.odt

7\_26\_11\_2019\_Besprechungsprotokoll\_Zwischenstand.odt

8\_05\_12\_2019\_Besprechungsprotokoll\_Zwischenstand.odt

9\_06\_12\_2019\_Besprechungsprotokoll\_Vorstellung\_Prototyp.odt

10\_19\_12\_2019\_Besprechungsprotokoll\_Zwischenstand.odt

11\_09\_01\_2020\_Besprechungsprotokoll\_Zwischenstand.odt

12\_16\_01\_2020\_Besprechungsprotokoll\_Zwischenstand.odt

13\_12\_02\_2020\_Besprechungsprotokoll\_Zwischenstand.odt

- Projekthandbuch

ERP\_Connect\_Projekthandbuch.odt

- Zeiterfassung

Projektplan\_ERP\_Connect.ods



**HÖHERE TECHNISCHE BUNDESLEHRANSTALT  
für INFORMATIONSTECHNOLOGIE  
und  
BUNDESFACHSCHULE für INFORMATIONSTECHNIK**

Pflichtenheft ERP-Connect  
Version 2.0

HTBLA Traun	Nur für den internen Gebrauch
<b>Ersteller:</b> Name: Finn Dorninger E-Mail: dorninger.fi@gmail.com	<b>Prüfer:</b> Name: E-Mail:
Status: Zum Review	Datum:

## Dokumentenverwaltung

### Dokument-Historie

Ver.	Status	Datum	Verantwortlich	Änderungsgrund
1.1	In Arbeit	16.08.2019	Dorninger	Erstellung.
1.2	In Arbeit	29.08.2019	Dorninger	Überarbeitung mit Arbeitsgeber
1.3	In Arbeit	30.08.2019	Dorninger	Ausformulierung der mit dem Arbeitsgeber besprochenen Punkte.
1.4	In Arbeit	15.02.2020	Dorninger	Verbesserung von Fehlern
2.0	Vollständig	27.03.2020	Dorninger	Beschriftung der Grafiken verbessert. Einfügen des Logos.

### Änderungsberechtigte

Autor/in: Finn Dorninger

### Dokument wurde mit folgenden Tools erstellt:

Microsoft Word Office365 V1908

# 1 Vorgaben an die Projektabwicklung

## 1.1 Zusammenhang mit bereits laufenden Projekten

Es gibt keine Zusammenhänge mit bereits laufenden Projekten.

## 1.2 Zusammenhang mit Vorgänger- und Nachfolgeprojekten

ERP-Connect ist ein Nachfolgeprojekt der Anwendung "Mobile WinLine". Diese Anwendung verwendet das Unternehmen Sysco EDV und dessen Kunden. Das Vorgängerprojekt besitzt keine Offline-Datenspeicherung. Dieses Projekt wird für Android-Geräte entwickelt und eine Offline-Datenspeicherung unterstützen.

## 1.3 Zweck des Produkts

Ziel ist es dem Endbenutzer eine App zur Verfügung zu stellen, um

- die Kundenbindung zu stärken.
- bessere Informationsmöglichkeiten für die Mitarbeiter des Unternehmens zu bieten.
- eine flexiblere Verfügbarkeit durch Offline-Verfügbarkeit zu unterstützen.
- ein innovatives und modernes Hilfsmittel bereit zu stellen.

## 1.4 Abgrenzung und Einbettung des Produkts

Diese App muss auf den Webservice, welcher von dem Auftragsgeber bereitgestellt wird, zugreifen können und diese Daten auf einem Android-Gerät darstellen.

## 1.5 Überblick über die geforderte Funktionalität

Die App soll folgende **Funktionen** bereitstellen:

- Verbindungseigenschaften festlegen über Einstellungsfenster
- Kommunikation mit dem Webservice
- Kunden-/Ansprechpartnerdaten visualisiert
- Daten übersichtlich dargestellt
- Offline-Zwischenspeicherung der Daten

Folgende **nicht-funktionalen Anforderungen** sollen erfüllt werden:

- Sicherheit der Daten gewährleistet

## 1.6 Allgemeine Einschränkungen

Die Anwendung ist nur für Android-Geräte verfügbar.

## 1.7 Vorgaben zu Hardware und Software

Die Anwendung soll lauffähig auf jedem Android Gerät ab der Version 5.0 sein.

## 1.8 Benutzer des Produkts

Die Endbenutzer sind Mitarbeiter und Kunden des Unternehmens SYSc EDV. Diese besitzen ein Android-Gerät und beabsichtigen Kunden-/Ansprechpartnerdaten grafisch auf dem Android-Gerät übersichtlich dargestellt zu bekommen.

## 2 Detaillierte Beschreibung der geforderten Produktmerkmale

### 2.1 Lieferumfang

Der Lieferumfang beinhaltet die APK der App, eine Bedienungsanleitung für die Installation der Anwendung und den Sourcecode.

### 2.2 Abläufe (Szenarien) von Interaktionen mit der Umgebung

Die einzelnen Funktionen werden im Kapitel 2.4 detailliert beschrieben.

### 2.3 Ziele des Benutzers

Benutzer dieser App haben folgende Ziele:

- Der User kann die bereitgestellten Daten jederzeit abrufen.
- Der User ist immer informiert über die zur Verfügung gestellten Daten.

### 2.4 Geforderte Funktionen des Produkts

#### 2.4.1 Startmenü

Das Startmenü beinhaltet ein „Bottom-Menü“ bei dem zwischen Kunden und Kontakte gewechselt werden können. Die Daten werden in einer Liste dargestellt. Mit einer Suche kann nach Kunden/Kontakten mit übereinstimmenden Namen gesucht werden. Durch die Auswahl eines Listeneintrag startet die Detailansicht des Kunden/Kontaktes.

#### 2.4.2 Detailansicht

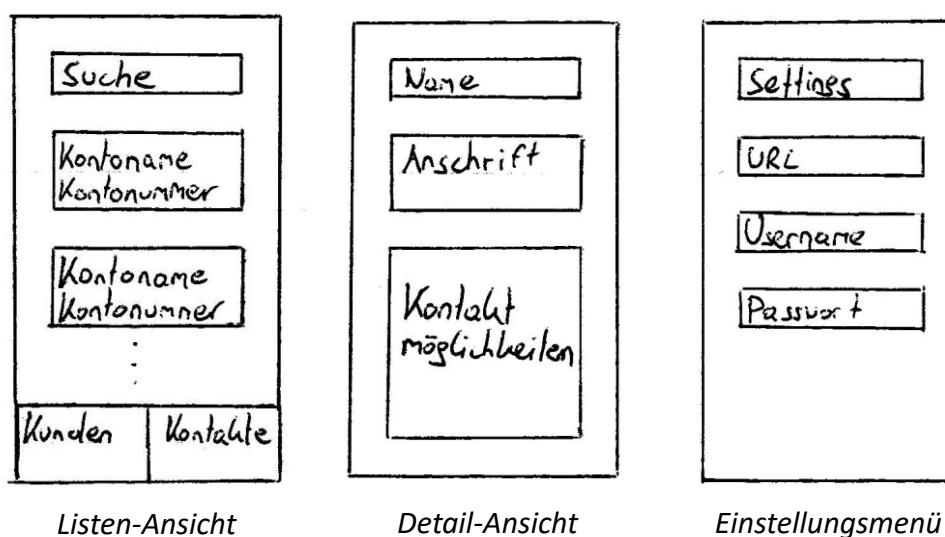
In der Detailansicht werden Informationen, der Kunden (Kontonummer, Kontoname, Staat, Postleitzahl, Ort, Straße, Telefon, Mobiltelefonnummer, E-Mail, WWW-Adresse, Notiz) beziehungsweise der Kontakte (Geschlecht, Name, Vorname, Emailadresse, Telefon-Land, Telefon-Vorwahl, Telefon-Nummer, Mobil Land, Mobil-Vorwahl, Mobil-Nummer) übersichtlich dargestellt.

#### 2.4.3 Einstellungsmenü

Einstellungsmenü in dem die Verbindungseigenschaften (URL, Benutzername und Passwort) zu dem Webservice eingestellt werden können.

### 2.5 Externe Schnittstellen des Produkts

#### 2.5.1 Benutzerschnittstellen (User Interfaces)



## 2.5.2 Systemschnittstellen

Die Applikation verwendet eine Systemschnittstelle zu einem REST-Webservice, welcher von dem Auftragsgeber zur Verfügung gestellt wird. Außerdem soll es möglich sein auch andere REST-Webservices einzubinden.

## 2.6 Sonstige geforderte Produktmerkmale

### 2.6.1 Geschwindigkeitsmerkmale (Performance)

Das lokale Laden der einzelnen Kunden/Kontakte darf maximal 5 Sekunden benötigen.

### 2.6.2 Ressourcenmerkmale (Resource)

Die Anwendung soll bis zu 10000 Kunden speichern und darstellen können. Für die maximale Anwendungsgröße wurden keine Bedingungen gesetzt.

### 2.6.3 Schutzmerkmale (Security)

Die Kommunikation mit dem REST-Webservice muss verschlüsselt stattfinden. Damit wird die Datenübertragen zwischen dem Rest-Webservice und der Anwendung abgesichert.

### 2.6.4 Sicherheitsmerkmale (Safety)

Die lokal gespeicherten Daten müssen verschlüsselt gespeichert.

### 2.6.5 Portabilitätsmerkmale (Portability)

Es gibt keine Portabilitätsmerkmale.

### 2.6.6 Stabilitätsmerkmale (Reliability)

Anwendung darf nicht bei fehlenden Daten, nicht gelungener Webservice-Verbindung und fehlendem Internet abstürzen.

### 2.6.7 Wartungsmerkmale (Maintenance)

Es wurden keine Wartungsmerkmale ausgemacht.

### 2.6.8 Wiederverwendbarkeitsmerkmale (Reuse)

Das Unternehmen kann ihren Kunden die App zur Verfügung stellen.

### 2.6.9 Benutzbarkeitsmerkmale (Usability)

Die darzustellenden Daten sollen übersichtlich dargestellt werden.

### 2.6.10 Benutzbarkeitsmerkmale (Usability)

Es wurden keine Benutzbarkeitsmerkmale ausgemacht.

## 3 Vorgaben an die Projektabwicklung

### 3.1 Anforderungen an die Realisierung

- Hardware
  - Entwicklungsrechner (Eigener Laptop und der Laptop des Unternehmens)
  - Android-Testgerät (Oneplus 5T)
- Software
  - Android Studio IDE
  - Kotlin, Java, XML

### 3.2 Fertige und zugekaufte Komponenten

Es werden keine fertige beziehungsweise zugekaufte Komponenten benötigt.

### 3.3 Unterauftragnehmer

Es gibt keinen Unterauftragnehmer.

### 3.4 Abnahmebedingungen

Das Produkt wird gegen das Pflichtenheft abgenommen.

Die Abnahme ist erfolgreich, wenn:

- der Kunde die App erfolgreich starten und schließen kann
- der Kunde sich zu dem Webservice verbinden kann
- der Kunde bereitgestellte Daten abrufen kann
- der Kunde offline die gespeicherten Daten abrufen kann.

### 3.5 Lieferbedingungen

Die Lieferkomponenten werden nach dem Abschluss des Projekts in Form einer APK und einem Zip-File mit dem Projekt-Sourcecode geliefert. Lieferform ist dabei elektronisch.

### 3.6 Anforderungen an den Einsatz

Die Produktübergabe erfolgt persönlich bei dem Unternehmen.

### 3.7 Gewährleistung

Es wurde keine Gewährleistung geregelt.

## 4 Verpflichtungen des Auftraggebers

Der Auftraggeber stellt die Webservice-Schnittstelle und damit die Daten der App zur Verfügung.

**ERP-Connect**



**Besprechungsprotokolle**

# **Projekt: ERP-Connect**

## **1.) Protokoll zur Vorstudie mit dem Auftragsgeber**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
SYSco EDV, 4311 Schwertberg	Peter Wurm, Alois Gaisberger, Finn Dorninger	05.06.2019 09:15-09:45

### **TOP's:**

1. Inhalt des Projektes
2. Wichtige Tools
3. Praktikum

### **TOP 1: Inhalt des Projektes**

Ziel wird es sein eine Android-App zu programmieren. Diese soll Daten (XML) von einem Webservice abfragen und darstellen. Diese Daten sind Kundendaten und Ansprechpartnerdaten. (Müssen aus Datenschutz-Gründen geschützt werden). Diese Daten sollen auch lokal gespeichert werden, um eine Offline-Darstellung zu ermöglichen. Auch die Sicherheit der Daten (Https-Verschlüsselung, Sichere Kommunikation mit der API, sicheres Speichern der Daten auf dem Gerät) soll gewährleistet sein.

### **TOP 2: Wichtige Tools**

SOAP UI eignet sich für das Testen von SOAP-/REST-APIs.

### **TOP 2: Praktikum**

Es besteht die Möglichkeit bei dem Unternehmen SYSco EDV die Vorbereitung der Diplomarbeit mit einem Praktikum zu kombinieren.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Peter Wurm über Praktikum informieren	12.06.2019

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **2.) Protokoll zur Startbesprechung**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	26.06.2019 11:00-11:45

### **TOP's:**

1. Vorgehensweise
2. Benötigte Unterlagen
3. Speicherung der Daten
4. Präsentationen und Korrekturlesen

### **TOP 1: Vorgehensweise**

Für eine erfolgreiche Absolvierung der Diplomarbeit ist folgendes zu beachten:

- Inhaltsverzeichnis für einen roten Faden (grobe Planung diesen Sommer, Punkte werden später gefüllt)
- Quellen sammeln
- Zitier-Regeln wiederholen
- Mindestens 4 bis 5 Bücher für die Handreichung
- Geachtet wird auf: Termin-Einhaltung, Präsentationen, Fachkompetenz, etc.
- Vorbereitung auf Besprechungen wird erwartet -> Verhinderungen müssen begründet werden.

### **TOP 2: Benötigte Unterlagen**

Folgende Unterlagen sind bis Herbst zu erstellen:

- Pflichtenheft
- Projektplan
- Zeitplanung
- Planung des Inhaltverzeichnisses

Durchgehend:

- Führung von Besprechungsprotokollen
- Führung von Begleitprotokollen

## **TOP 3: Speicherung der Daten**

Sourcecode und Projektordner sollen dem Betreuer bereitgestellt werden mit folgenden Tools:

- Dokumentation: Office 365/One Drive
- Software: Git

## **TOP 4: Präsentation und Korrekturlesen**

Korrekturlesen ist selbst zu organisieren. Mindestens 2 Präsentationen verpflichtend. Empfohlen: ab der Zweite mit Anzug!

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Pflichtenheft, Projektplan, Zeitplanung, Planung Inhaltsverzeichnis Source-Code auf Git	Herbst
Finn Dorninger	Führung Begleitprotokoll	Immer

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **3.) Protokoll – Projektstart Besprechung mit Auftragsgeber**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
SYScos EDV, 4311 Schwertberg	Peter Wurm, Finn Dorninger	29.08.2019 15:00-15:45

### **TOP's:**

1. Besprechung Pflichtenheft
2. Kontaktmöglichkeiten

#### **TOP 1: Besprechung Pflichtenheft**

- Zusammen mit Herr Wurm wurde das Pflichtenheft überarbeitet. Die genauen Anforderungen des Projektes wurden festgelegt. (Siehe Pflichtenheft)

#### **TOP 2: Kontaktmöglichkeiten**

Der Kontakt geschieht E-Mail oder Telefon. Dafür wurde eine Visitenkarte übergeben.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Keiner	Keine	Nie

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **4.) Protokoll zur Startbesprechung**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
SYSco EDV, 4311 Schwertberg	Thomas Rafetseder, Finn Dorninger	05.09.2019 13:-13:30

### **TOP's:**

1. Projektcontrolling
2. Kontaktmöglichkeiten

### **TOP 1: Projektcontrolling**

Es soll:

- alle 2-3 Wochen soll ein telefonisches Update geschehen.
- alle 4-6 Wochen soll ein Treffen bei dem Unternehmen stattfinden.

### **TOP 2: Kontaktmöglichkeiten**

Der Kontakt geschieht über E-Mail und Telefon. Ein Termin soll möglichst eine Woche davor angekündigt werden. Am Besten eignet sich dafür der Freitag.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Updates an das Unternehmen geben.	regelmäßig

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **5.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	10.09.2019 09:15-10:15

### **TOP's:**

1. Projektstand und Probleme

### **TOP 1: Projektstand und Probleme**

Bei der Vorbereitung auf die Diplomarbeit sind Probleme bei der Programmierung entstanden. (Inkompatibilitäten zwischen Libraries, Schwierigkeiten beim Darstellen der bezogenen Daten, Übersicht über die Anwendung verloren). Es wurde ausgemacht, das Projekt in Teile einzuteilen. Dies schafft einen besseren Überblick bei der Entwicklung. Außerdem werden Teile der Entwicklung voneinander separiert, was die Entwicklung erleichtert.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Keine	/

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **6.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	11.10.2019 12:00-13:05

### **TOP's:**

1. Projektstand
2. Schriftliche Ausarbeitung der Diplomarbeit, Präsentation
3. Benötigte Unterlagen

### **TOP 1: Projektstand**

Review der Projektpläne. Ein Gantt-Balkendiagramm ist anzufertigen. Dort sollen auch die Präsentationen hinzugefügt werden.

### **TOP 2: Schriftliche Ausarbeitung und Präsentation**

Für die schriftliche Ausarbeitung empfiehlt sich als erster Schritt das Schreiben eines Inhaltverzeichnisses. Beim Schreiben der Diplomarbeit soll darauf geachtet werden, dass man die verschiedenen Möglichkeiten zu einer Problemlösung darstellt und Entscheidungen bekräftigt.

Für die Präsentationen gibt es eine Powerpoint-Vorlage. Die Präsentation soll spätestens zwei Wochen vor der Präsentation zur Überprüfung abgeben werden.

### **TOP 3: Benötigte Unterlagen**

Es wird noch benötigt:

- Projektordner mit Besprechungsprotokolle
- GIT-Zugriff
- Gantt-Balkendiagramm

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Projektordner mit Besprechungsprotokolle, Git-Zugriff, Gantt	20.10.2019
Finn Dorninger	Präsentation, Inhaltsverzeichnis	27.11.2019

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **7.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	26.11.2019 11:00-11:45

### **TOP's:**

1. Projektstand
2. Feedback - Prototyp
3. Vorgehensweise

#### **TOP 1: Projektstand**

Prüfung des aktuellen Projektstandes. Problem, dass die gesamte Kontakt-Liste der App erst lokal verfügbar ist, wenn diese einmal aufgerufen wurden.

#### **TOP 2: Feedback - Prototyp**

Es wurde vorgeschlagen, dass die Kontakte automatisch beim Start der Anwendung synchronisiert werden sollen. Außerdem könnte man dies mit einem Einstellungsmenü-Eintrag verknüpfen, bei dem der Benutzer diese Funktion auch deaktivieren kann.

#### **TOP 3: Vorgehensweise**

Fertigstellung des Prototyps, um sich danach auf die Präsentation und auf das Schreiben der Diplomarbeit konzentrieren zu können.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Hinzufügen der automatischen Synchronisation	05.12.2019
Finn Dorninger	Neuer Termin für ein Gespräch	25- 28.12.2019

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **8.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	05.12.2019 13:30-14:10

### **TOP's:**

1. Projektstand und Präsentation
2. Vorgehensweise: Erstellung Enddokumentation

### **TOP 1: Projektstand und Präsentation**

Pflichten des Projektes sind alle erfüllt, in den folgenden Wochen verbessern. Für die Präsentation wird eine Präsentationsfernbedienung zur Verfügung gestellt. Verbesserung der Schlussfolie notwendig. Auch muss die Präsentation gekürzt werden.

### **TOP 2: Vorgehensweise bei der Erstellung der Enddokumentation**

Für das Schreiben empfiehlt es sich ein Inhaltsverzeichnis zu schreiben und diese mit voraussichtlichen Quellenangaben zu füllen.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Überarbeitung der Präsentation	11.12.2019

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **9.) Protokoll zur Vorstellung des Prototyps**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
Sysco EDV, 4311 Schwertberg	Peter Wurm, Finn Dorninger	06.12.2019 13:00-13:50

### **TOP's:**

1. Vorstellung des Prototyps
2. Feedback

### **TOP 1: Vorstellung des Prototyps**

Präsentation des Prototyps und dessen Funktionen inklusive der folgenden Schritte.

### **TOP 2: Feedback**

Für die Verschlüsselung ist es wichtig die lokalen Daten und die Webservice-Übertragung zu verschlüsseln. Mindest-Android Version darf dafür auf Jetpack erhöht werden. Der Code soll mit Kommentaren kurz beschrieben werden. Der Auftragsgeber würde sich über die schriftliche Diplomarbeit inklusive einer Dokumentation der benötigten Zeit freuen.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Handbuch über die Vorgehensweise bei der Installation der Anwendung, Schriftliche Diplomarbeit, Zeiterfassung.	Nach Fertigstellung.

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **10.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	19.12.2019 13:35-14:35

### **TOP's:**

1. Vorgehensweise schriftliche Diplomarbeit
2. Feedback Präsentation
3. Abgabefristen

### **TOP 1: Vorgehensweise schriftliche Diplomarbeit**

Inhaltlich besonders auf Webserver, Kommunikation und Encryption eingehen. Eigene Vorgehensweise immer begründen. Möglichkeit zu verweisen, um den roten Faden nicht zu verlieren. Bei den IST/Soll-Fragen noch nicht in die Tiefe gehen und gegebenenfalls auf ein späteres Kapitel verweisen.

### **TOP 2: Feedback Präsentation**

Anfang Januar steht eine neue Präsentation an. Dort ist es wichtig mehr zu erklären was man macht und weniger auf Details eingehen.

### **TOP 3: Abgabefristen**

Spätestens 23 März finale Version. (30. April muss diese bereits gebunden sein). Mitte Februar – Abgabe damit es sich ausgeht, dass Feedback des Betreuers noch eingearbeitet werden kann.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Fertige Version Diplomarbeit	Mitte Feb.

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **11.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	09.01.2020 11:00-11:10

### **TOP's:**

1. Feedback Präsentation
2. Anpassen des Zeitplanes

### **TOP 1: Feedback Präsentation**

Präsentation anpassen: Redundanzen vermeiden, mehr Symbole einsetzen, Zweck der Anwendung hervorheben.

### **TOP 2: Anpassen des Zeitplanes**

Anpassen des Zeitplanes damit man sich auf das Schreiben der Diplomarbeit konzentrieren kann. Nach dem Schreiben kann das Testen der Anwendung im Laufe des Projektes nachgeholt werden.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Anpassen des Zeitplanes	/

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **12.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	16.01.2020 13:45-14:25

### **TOP's:**

1. Feedback zur zweiten Zwischenpräsentation
2. Schriftlicher Teil der Diplomarbeit

### **TOP 1: Feedback zur zweiten Zwischenpräsentation**

Bei der nächsten Zwischenpräsentation soll vor allem die Rhetorik verbessert werden. Weniger Allgemeines dafür den Zweck mehr hervorheben.

### **TOP 2: Schriftlicher Teil der Diplomarbeit**

Bei dem schriftlichen Teil keine „man“-Sätze verwenden und gegebenenfalls ein Glossar verwenden. Bis Ende dieser Woche-Abgabe erste Version der Diplomarbeit. Ende Semesterferien schriftlicher Teil fertigstellen, damit die Diplomarbeit Mitte März freigegeben/abgegeben werden kann.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Abgabe erste Version der Diplomarbeit	19.01.2020
Finn Dorninger	Neues Gespräch	12.02.2020

Für das Protokoll:

Finn Dorninger

# **Projekt: ERP-Connect**

## **13.) Protokoll – Zwischenstand mit Betreuer**

<b>Ort</b>	<b>Teilnehmer</b>	<b>Datum/Dauer</b>
HTL Traun, 4050 Traun	Alois Gaisberger, Finn Dorninger	12.02.2020 13:35-14:25

### **TOP's:**

1. Projektzwischenstand
2. Feedback zu dem schriftlichen Teil der Diplomarbeit

### **TOP 1: Projektzwischenstand**

Betreuer über den aktuellen Projektstand informiert, das Testen der Anwendung soll niedriger priorisiert werden, im Vergleich zu dem schriftlichen Teil der Diplomarbeit.

### **TOP 2: Feedback bezüglich schriftlichen Teils der Diplomarbeit**

Immer zuerst die Möglichkeiten für die Lösung eines Problems fertig schreiben und erst dann die Entscheidung begründen. Hinweise (sprachlich und fachliche Unklarheiten) werden im Word-Dokument als Kommentar markiert und sind auszubessern. Die Arbeit soll verbessert und in den Ferien fertiggestellt werden, damit noch genug Zeit für mögliche Verbesserungsvorschläge und Formatierung offen ist.

### **Vereinbarungen**

<b>Wer</b>	<b>Vereinbarung</b>	<b>Bis wann</b>
Finn Dorninger	Abgabe der ausgebesserten, vollständig ausgefüllten Arbeit.	27.12.2020

Für das Protokoll:

Finn Dorninger

**HÖHERE TECHNISCHE BUNDESLEHRANSTALT für  
INFORMATIONSTECHNOLOGIE  
und BUNDESFACHSCHULE für INFORMATIONSTECHNIK**

# Projekthandbuch

## ERP-Connect

VerfasserInnen: Finn Dorninger

Projektleiter: Finn Dorninger

Datum: 27.03.2020

Bahnhofstraße 52, 4050 Traun  
Tel: +43 7229 623 11, Fax: +43 7229 623 11-41  
E-Mail: office@htltraun.at  
Web: <http://www.htltraun.at>  
Schulkennzahl: 410457

# Inhaltsverzeichnis

1	Änderungsverzeichnis .....	3
2	Ansprechpartner .....	3
3	Projektpläne .....	3
3.1	Projektfeld-Analyse .....	3
3.2	Datenschutzrechtliche und sicherheitstechnische Aspekte .....	5
3.3	Objektstrukturplan .....	5
3.4	Projektstrukturplan .....	5
3.5	Projektmeilensteinplan .....	6
3.6	Projektbalkenplan (Gantt) .....	6
3.7	Projektrisikoanalyse .....	7
3.8	Qualitätsmanagement .....	8
4	Projektcontrolling .....	8
4.1	Präsentation .....	8
4.2	Meilenstein-Trendanalyse .....	9
4.3	Reviews .....	9
5	Projektabschluss .....	10

# 1 Änderungsverzeichnis

Version	Datum	Änderung	Ersteller
1.0	2019.08.19	Neuerstellung, Projektumfeld, Objektstrukturplan	Dorninger
1.1	2019.08.20	Projektstrukturplan, Datenschutzaspekte	Dorninger
1.2	2019.09.06	Meilensteinliste, Projektrisikoanalyse, Qualitätsmanagement, Projektcontrolling	Dorninger
1.3	2019.10.19	Gantt hinzugefügt	Dorninger
1.4	2020.02.15	Sicherheitsaspekte angepasst	Dorninger

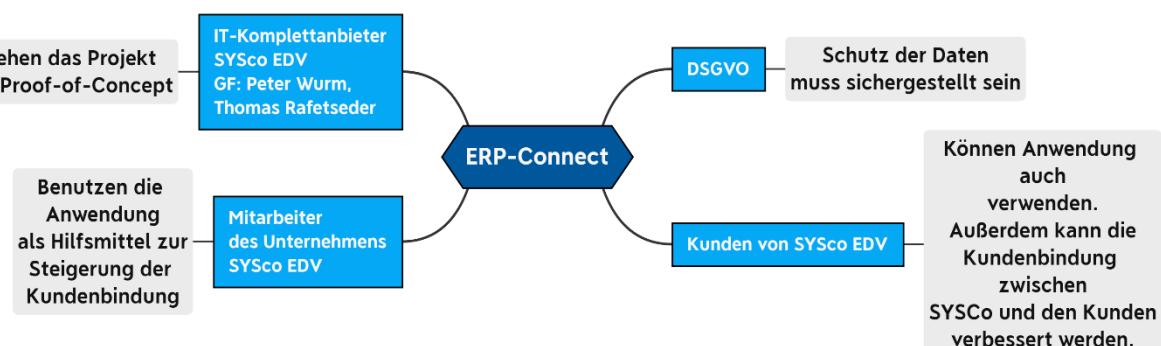
## 2 Ansprechpartner

Name	Organisationseinheit	Rolle im Projekt	Erreichbarkeit (Tel, Mail)
Finn Dorninger	HTL-Traun	PL, Programmierung, Testen	dorninger.fi@gmail.com
Alois Gaisberger	HTL-Traun	Betreuer	Alois.gaisberger@htltraun.at
Peter Wurm	Geschäftsführer SYSCO EDV	Auftraggeber	p.wurm@sysco.at

## 3 Projektpläne

### 3.1 Projektumfeld-Analyse

Projektumfeld–Graphik



## Projektumfeld-Beziehungen

Stakeholder	Einfluss auf das Projekt	Konflikt-Potenzial	Sympathie, Erwartungen	Antipathie, Befürchtungen	Strategische Vorkehrungen
Prof. Gaisberger Diplomarbeit-Betreuer	mittel	mittel	<p>Prof. Gaisberger erwartet eine selbstständige und termingerechte Projektdurchführung. Außerdem erwartet er ein gut programmiertes Endprodukt.</p> <p>Das Projekt-Team erwartet Unterstützung bei der Durchführung der Diplomarbeit.</p>	<p>Befürchtung: Das größtenteils autodidaktische Programmierkenntnis Prof. Gaisberger nicht genügen werden.</p> <p>Unsicherheit ob alle Termine termingerecht durchgeführt werden können.</p> <p>Befürchtung das etwas Ausgemachtes übersehen werden könnte.</p>	<p>Sorgfältige Programmierung.</p> <p>Ausführliche Protokoll-Führung bei Gesprächen.</p> <p>Regelmäßige Updates über den Projektstand.</p>
Peter Wurm Auftraggeber Geschäftsführer SYScos EDV	sehr groß	gering	Hr. Wurm sieht die Diplomarbeit als „Proof-of-Concept“ und ist sehr interessiert an dem Endergebnis.	Keine.	<p>Hr. Wurm die Möglichkeit geben Wünsche zu äußern.</p> <p>Sehr genaue End-Dokumentation.</p>
Thomas Rafetseder Geschäftsführer SYScos EDV	sehr groß	mittel	Hr. Rafetseder hat sehr hohe Erwartungen an der Anwendung und erwartet eine einwandfreie Anwendung.	Befürchtung, dass das Endprodukt Hr. Rafetseder nicht genügen wird.	<p>Genaue Recherche vor dem Programmieren.</p> <p>Genaues Testen des Endproduktes.</p> <p>Hr. Rafetseder die Möglichkeit geben Wünsche zu äußern.</p>

## 3.2 Datenschutzrechtliche und sicherheitstechnische Aspekte

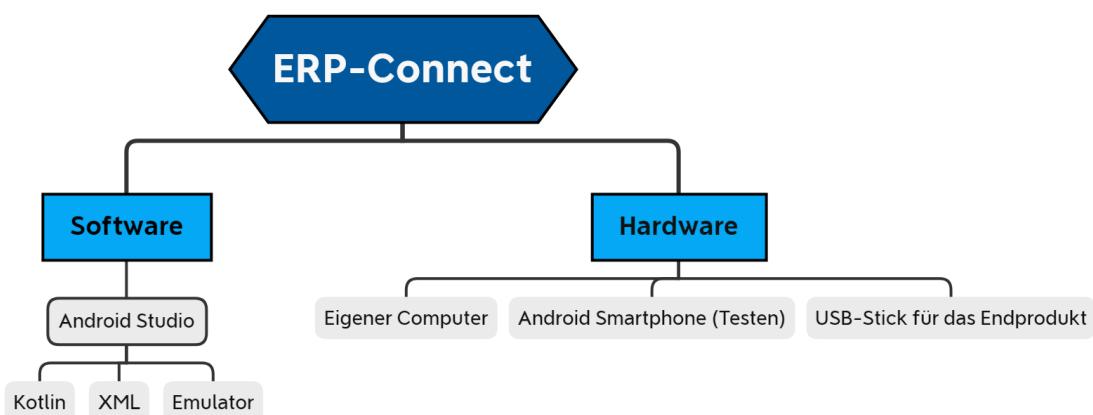
- **Datenminimierung:**

Die Anwendung ERP-Connect verarbeitet personenbezogene Daten. Diese werden durch einen WinLine-Webservice bereitgestellt. Die Anwendung verarbeitet und speichert nur jene Daten, welche auch dargestellt werden sollen.

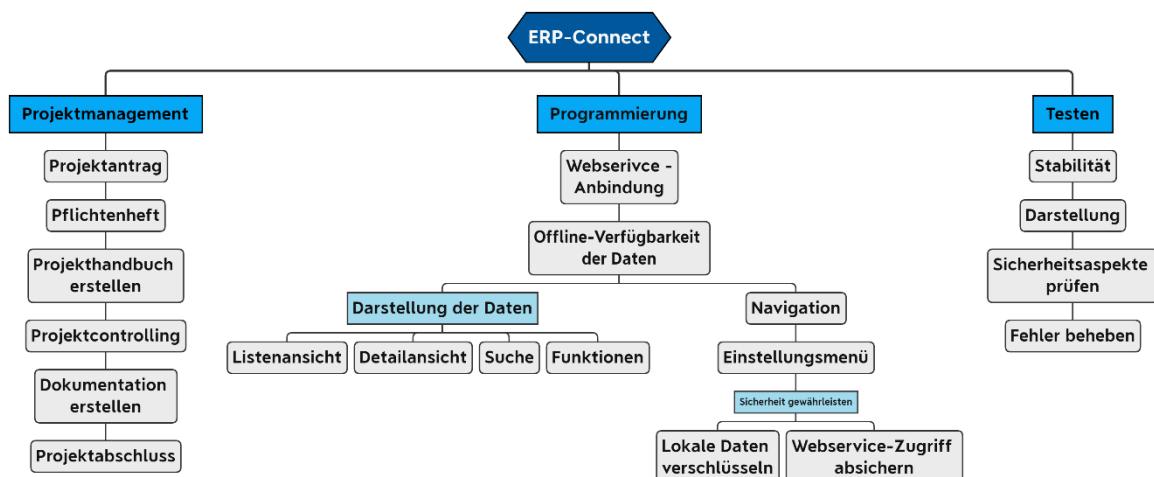
- **Schutz der Daten:**

Der Webservice-Zugriff geschieht über HTTPS, damit wird die Kommunikation zwischen Webservice und der Anwendung abgesichert. Für die Offline-Verfügbarkeit werden Daten lokal gespeichert. Diese Daten müssen verschlüsselt gespeichert werden, um vor einem fremden Zugriff zu schützen.

## 3.3 Objektstrukturplan



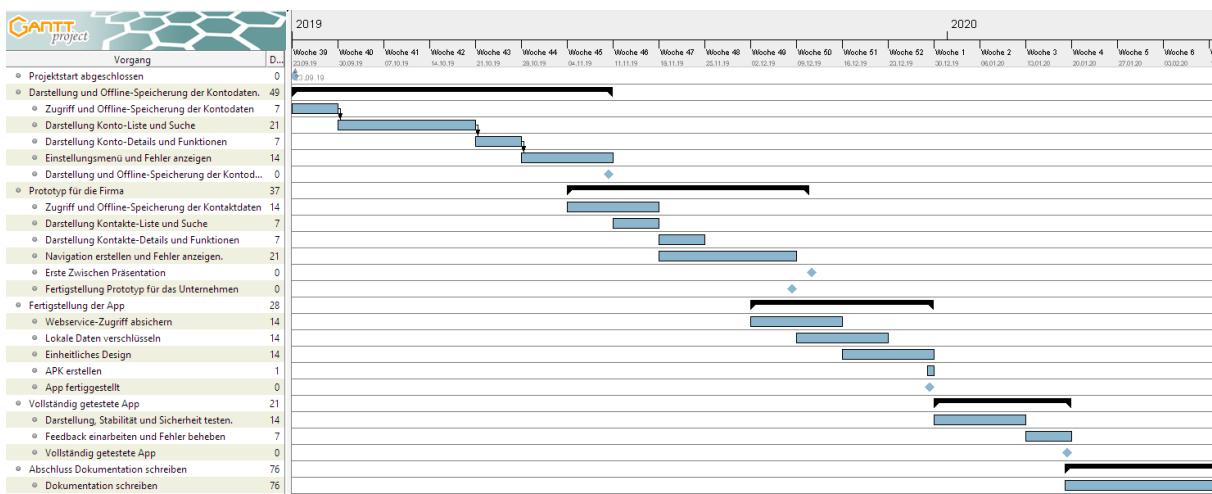
## 3.4 Projektstrukturplan



### 3.5 Projektmeilensteinplan

PSP-Code	Meilenstein	Soll-Termin	Ist-Termin
--	Projektstart abgeschlossen	24.09.2019	24.09.2019
1.2	Darstellung und Offline-Speicherung der Kontodaten	01.11.2019	10.11.2019
1.3	Fertigstellung Prototyp für das Unternehmen	06.12.2019	06.12.2019
--	Fertigstellung der App	28.12.2019	16.12.2019
2	Vollständig getestete App	18.01.2019	14.03.2020
--	Schriftlicher Teil der Diplomarbeit abgeschlossen	27.03.2020	27.03.2020

### 3.6 Projektbalkenplan (Gantt)



### 3.7 Projektrisikoanalyse

MS-Code	Arbeitspaket - Bezeichnung	Risiko-Beschreibung, Ursache	Priorität	Risiko-kosten	Wahr-scheinli-chkeit	Risikowert	Verzögerung	Präventive und Korrektive Maßnahmen	Risiko-Minimierungskosten
		Text	Auswahl	Euro	Prozent	Euro	Wochen	Text	Euro
MS 2	Darstellung und Offline-Speicherung der Kontodaten	1.) Suche schwer zu programmieren. 2.) Darstellung funktioniert nicht, weil Daten nicht passen.	2	0	30	0	2	Ein Architekturmuster verwenden, um ein gutes Modell zu haben.	0
MS 3	Fertigstellung Prototyp für das Unternehmen	Keine sonderlichen Risiken.	4	0	0	0	0	/	0
MS 4	Fertigstellung App	1.) Es könnten Schwierigkeiten (zB. Offline-Speicherung muss komplett neu programmiert werden) bei dem verschlüsseln der Daten entstehen. 2.) Das Unternehmen äußert viele Änderungswünsche.	1	0	40	0	3	1.) Bereits davor mit Verschlüsselung vertraut/recherchieren machen. 2.) Regelmäßige Reviews mit dem Arbeitsgeber.	0
MS 5	Vollständig getestete App	Beim Testen sind Fehler entdeckt worden welche verbessert werden müssen.	3	0	30	0	2	Während des ganzen Projektes auf sorgfältige Programmierung achten.	0

## **3.8 Qualitätsmanagement**

### **Art des Vorgehensmodells**

Wasserfallmodell

### **Qualitätsziele für das Produkt**

Die Qualitätsziele für das zu erstellende Produkt sind die Folgenden:

- Die Anwendung soll auf sämtlichen Bildschirmen ähnlich aussehen.
- Die Anwendung darf nicht bei fehlender Internetverbindung oder fehlenden Daten abstürzen.
- Die Anwendung soll leicht bedienbar und übersichtlich sein.

### **Qualitätsziele für das Projekt**

Die Qualitätsziele für den Projektverlauf sind die Folgenden:

- Regelmäßige Reviews mit dem Auftragsgeber.
- Vorschläge des Betreuers und des Auftragsgebers umsetzen.

### **Testfälle**

1.) Die Anwendung stürzt bei der Bedienung ab.

Testfall: Programmcode ist schlecht geschrieben, Programmfehler wurden nicht ausgebessert.

Lösung: Überarbeitung des Programmcodes, sorgfältigeres testen.

2.) Die Darstellung ist unübersichtlich.

Testfall: Das Design der Anwendung wurde nicht durchdacht und ist unübersichtlich.

Lösung: Design überarbeiten, gegebenenfalls auch neu konzeptionieren.

Bei Reviews mit dem Auftragsgeber nach Verbesserungsvorschlägen fragen.

## **4 Projektcontrolling**

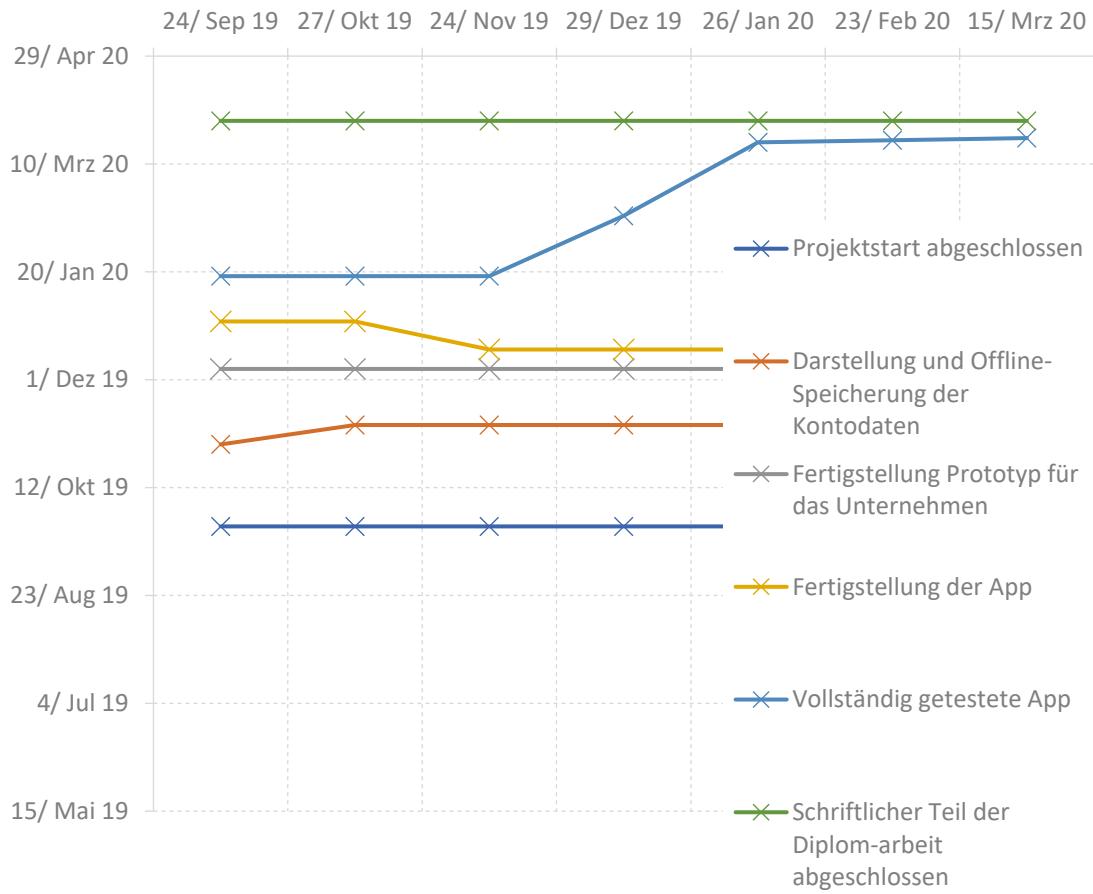
### **4.1 Präsentation**

Die Präsentationen des Projektstatus zu bestimmten Terminen erfolgt durch Folien in einem Präsentationsprogramm.

Folgende Punkte werden zu den Präsentationsterminen behandelt:

- Gesamtstatus
- Status Ziele
- Status Leistungsfortschritt
- Status Termine

## 4.2 Meilenstein-Trendanalyse



### Trendanalyse: ERP-Connect mit Stand vom 15.03.2020

Meilenstein-Titel	Plan-Termin Neue Abschätzung der Termine zu den jeweiligen Berichtsterminen						
	24/09/19	27/10/19	24/11/19	29/12/19	26/01/20	23/02/20	15/03/20
Projektstart abgeschlossen	24/09/19	24/09/19	24/09/19	24/09/19	24/09/19	24/09/19	24/09/19
Darstellung und Offline-Speicherung der Kontodaten	01/11/19	10/11/19	10/11/19	10/11/19	10/11/19	10/11/19	10/11/19
Fertigstellung Prototyp für das Unternehmen	06/12/19	06/12/19	06/12/19	06/12/19	06/12/19	06/12/19	06/12/19
Fertigstellung der App	28/12/19	28/12/19	15/12/19	15/12/19	15/12/19	15/12/19	15/12/19
Vollständig getestete App	18/01/20	18/01/20	18/01/20	15/02/20	20/03/20	21/03/20	22/03/20
Schriftlicher Teil der Diplomarbeit abgeschlossen	30/03/20	30/03/20	30/03/20	30/03/20	30/03/20	30/03/20	30/03/20

## 4.3 Reviews

Mit dem Auftragsgeber wurde ausgemacht, dass alle 3 Woche ein Review (telefonisches oder über E-Mail) über den Projektstand gehalten wird.

Des Weiteren wird alle 5-6 Woche ein Review bei der Firma SYSCO EDV gehalten, um dem Auftragsgeber die Möglichkeit zu geben Verbesserungsvorschläge zu geben und den aktuellen Stand der Anwendung zu sehen.

# 5 Projektabschluss

## Projektabschlussbericht

<b>Gesamteindruck</b> Die Entwicklung der Anwendung wurde erfolgreich mit der Erfüllung aller Projektziele abgeschlossen. Trotz vor dem Projektstart fehlenden Kenntnissen über das Android-Framework und der Programmiersprache Kotlin, wurde eine leicht bedienbare, übersichtliche und stabile Android-App entwickelt.	<b>Reflexion: Zielerreichung</b> Die Ziele, vor allem bei der Funktionalität der Anwendung wurden vollständig erreicht. Auch den Zweck des Produktes kann die Anwendung erfüllen vor allem mit den praktischen Funktionen der Detail-Ansichten. Ob die Kundenbindung verbessert werden kann, zeigt sich erst nach dem Einsatz der Anwendung.
<b>Reflexion: Leistungen/Termine</b> Die Termine wurden sehr gut eingehalten. Es kam jedoch zu einem Verzug, beim Testen der Anwendung und dem Fertigstellen der schriftlichen Dokumentation. Aufgrund von privaten Gründen und einer höheren Priorisierung des Schreibens der schriftlichen Diplomarbeit, musste das Testen verschoben werden. Das Testen konnte gegen Ende des Projektes fertiggestellt werden.	
<b>Reflexion: Ressourcen/Kosten</b> Es kamen keine Kosten bei der Programmierung auf. Es wäre möglich gewesen, eine kostenpflichtige verschlüsselnde Datenbank zu verwenden, da jedoch entschieden wurde Daten in einem XML-Dokument zu speichern konnten diese Kosten vermieden werden. Kosten entstanden nur für die Literatur.	
<b>Reflexion: Interne Organisation/ Umweltbeziehungen</b> Die Beziehung zwischen Auftragsgeber und dem Projektteam war sehr gut. Es sollte jedoch bei einem neuen Projekt darauf geachtet werden, den Auftragsgeber öfters über den Projektstand zu benachrichtigen.	
<b>Leistungsbeurteilung</b> Es wurde sehr viel Freizeit für diese Arbeit verwendet. Durch Perfektionistische Veranlagung war es schwierig Tätigkeiten als abgeschlossen anzuerkennen, wodurch zu viel Zeit in Aufgaben geflossen sind. Trotzdem entstand vor allem deswegen eine gute Erfüllung aller Projektziele. Wegen fehlenden Kenntnissen kam es zu Startschwierigkeiten (verlorener Überblick, Inkompatibilität zwischen Libraries mit der Kotlin-Programmiersprache), welche sich nach einer langen „Kennenzel-Phase“ mit der Entwicklungsumgebung inkl. der Programmiersprache besserte. Durch durchgängiges Mitlernen und dem Recherchieren mit verschiedenen Medien verbesserten sich die Kenntnisse über die Programmiersprache und dem Android-Framework laufend. Gegen Mitte und dem Ende des Projektes waren diese Kenntnisse schon sehr gut und erleichterten die Entwicklung.	<b>„Lessons learned“</b> Es war eine wertvolle Erfahrung ein großes Projekt durchzuführen. Eine Einteilung des Projektes in Teilaufgaben ermöglichte einen Überblick über das Projekt und schaffte ein System in die Programmierung. Die Kommunikation mit dem Auftragsgeber sollte bei dem nächsten Projekt verbessert werden, dadurch kann die Entscheidungsfindung erleichtert werden. Auch wurde gelernt, dass durch das Vermeiden von Stress und dem Anwenden des Pareto-Prinzips ein deutlich besseres Ergebnis erzielt werden kann. Ein Projekt allein durchzuführen erschafft Schwierigkeiten in der Entscheidungsfindung und dem Anerkennen von Aufgaben als abgeschlossen. Auch soll weniger darauf geachtet werden was noch alles offen ist oder noch nicht funktioniert, sondern lohnt es sich mehr auf die aktuelle Aufgabe zu konzentrieren. Einarbeitungszeit in Framework und Programmiersprache sollte nicht unterschätzt werden.
<b>2DO</b> Persönliche Übergabe, Beantwortung von offenen Fragen.	<b>WHO</b> Dorninger  <b>WHEN</b> Nach der Abgabe der Diplomarbeit.

**ERP-Connect**



**Zeiterfassung**

Projektplan_ERP_Connect.ods																		
Zeiterfassung: Finn Dorringer - ERP-Connect																		
Finn Dorringer																		
<b>Phasenübergreifende Aktivitäten</b>																		
Kontakt mit Arbeitgeber		9	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
<b>Idee+Vorstudie</b>																		
Projektauftrag erstellen, Startbesprechung		4	3	1														
<b>Definition+Planung+Analyse</b>																		
Einlesen: Android Studio, Kotlin		21	21															
Pflichtenheft, Projekthandbuch, Projektplan		26			10	2	6	5	3									
Einarbeitung: Einstellungs menü erstellen/Speichern von Daten, Android Studio		31			10	21												
<b>Projektstart</b>																		
Zugriff Webservice		20			6	14												
Speichern der Daten in Datenbank/Darstellung		14				14												
Architekturmuster		8				8												
Speichern/Laden von Objekten in XML-Datei		8				8												
<b>Darstellung und Offline-Sicherung der Kontodaten</b>																		
Webservice-Zugriff, Konten lokal speichern		13				8	3	1										
Darstellung der Konto-Liste		5						5										
Konto-Liste Suche erstellen		6						4	2									
Funktionen Konto		9						4	5									
GUI-Fehler anzeigen		7							6	1								
Einstellungs menü		8							8									
Darstellung der Konto-Details erstellen		9							5	4								
<b>Prototyp für die Firma</b>																		
Webservice Zugriff (Kontakte-Liste/Details)		3														1	2	

Kontakte lokal speichern	1
Darstellung der Kontakte-Liste	5
Verknüpfung mit Kontakt-Details	3
Kontakte-Liste Suche erstellen	3
Darstellung Kontakt-Details, Funktionen	6
GUI-Fehler anzeigen	2
Einstellungsmenü einbinden	5
Bottom-Navigation-Menü erstellen	7
<b>Fertigstellung App</b>	
WS-Zugriff absichern/Lokale Daten sichern	12
<b>Vollständig getestete App</b>	
Testen der Anwendung, Fehler beheben	42
<b>Abschluss</b>	
Abschluss Dokumentation schreiben	78

Projektplan\_ERP\_Connect.ods