ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Entwicklung einer dynamischen und prozessorientierten Augmented Reality Anwendung am Beispiel der AristaFlow BPM Suite und Android.

Masterarbeit an der Universität Ulm

Vorgelegt von:
Martin Fröchtenicht
martin.froechtenicht@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Prof. Dr. Peter Dadam

Betreuer:
Dipl. Inf. Marc Schickler

2014

Fassung 15. September 2014

Kurzfassung

Diese Masterarbeit beschäftigt sich mit dem sehr aktuellen Themengebiet **A**ugmented **R**eality (AR) verknüpft mit prozessorientierten mobilen Anwendungen. Anhand eines ausgewählten Szenarios wird gezeigt, dass diese Kombination neue innovative Methoden zum Anzeigen und Bearbeiten von Prozessinformationen ermöglicht. Aktivitäten, welche mit Positionsdaten verbunden sind (z.B. eine Adresse oder Koordinaten), werden direkt in die reale Welt projiziert und sind für den Nutzer über eine Smartphone-Kamera oder ein Head-up-Display visuell zugänglich. Auf Grundlage dieser Idee wird eine Android-Anwendung konzipiert, welche wesentliche Funktionen implementiert, um den Nutzer bei der Prozessbearbeitung gezielt zu unterstützen. Diese werden aus technischer und konzeptioneller Sicht im Detail erläutert.

Als Szenario für diese Anwendung wurde ein Kurierdienst ausgewählt. Sämtliche damit verbundenen Abläufe, wie das Erfassen einer Kundenunterschrift, die Auswahl eines Abholdatums oder die Authentifizierung des Kuriers, werden durch die Anwendung unterstützt. Da die Anwendungsfälle, wie z.B. ein Abhol- und Lieferauftrag, sehr prozesslastig sind, eignet sich die AristaFlow BPM Suite optimal als Grundgerüst. Zur Navigation bei einer Paketlieferung, kann der Kurier die AR-Sicht benutzen, welche die wichtigsten Prozessinformationen und zusätzlich den Zielort in der realen Welt anzeigt. Navigation und Prozessbearbeitung sind damit fließend vereint. Darzustellende und zu erfassende Inhalte werden durch die AristaFlow Process-Engine gesteuert. Die Abläufe sind hier in Form eines Prozesstemplates hinterlegt und können bei Bedarf geändert bzw. erweitert werden. Des Weiteren profitiert man von den umfangreichen Organisations- und Prozessmodellierungsmöglichkeiten. Über den AristaFlow Webservice erhält die mobile Anwendung Prozessdaten und generiert für deren Anzeige und Bearbeitung dynamisch GUI-Elemente.

Inhaltsverzeichnis

1

Einleitung

Mobilität spielt in unserer heutigen Zeit eine entscheidende Rolle. Moderne Smartphones haben sich im Alltag etabliert und sind kaum noch wegzudenken. Nicht nur im privaten sondern vor allem im beruflichen Bereich, wird das Smartphone viel genutzt und ersetzt oftmals bereits den Desktop-PC. Zu den Anforderungen gehören nicht nur das einfache Abrufen von Emails oder das Erstellen eines Termins im Kalender. Da die Rechenleistung für graphisch komplexe und anspruchsvolle Anwendungen keine Barriere mehr darstellt, ist der Startschuss für neue innovative Herangehensweisen, um Informationen anzuzeigen (siehe Abbildung 1.2(a)), gegeben. Inhalte können über die Smartphone-Kamera oder Datenbrille (Google Glass[1]) in die reale Welt hineinprojiziert werden. Bei der großen Menge an Daten ist es wichtig diese geeignet zu filtern [10], da der Platz für deren Anzeige sich auf den Bildschirm des Smartphones begrenzt. Die Position des Nutzers kann dazu verwendet werden Datenbanken spezifisch nach

[1] http://www.google.com/glass/start/

Informationen zu durchsuchen, welche für seinen aktuellen Standort relevant sind. Weiterhin sind viele Informationen Bestandteil eines komplexeren Prozesses. Somit ergibt sich zusätzlich die Anforderung diese auf einem mobilen Client ausführen [21, 22, 23] und in der **A**ugmented **R**eality (AR) geeignet abbilden zu können. Über eine Prozess-Engine werden die Aktivitäten des Nutzers gesteuert, die für die Interaktion notwendigen GUI-Elemente müssen von der mobilen Anwendung dynamisch bereitgestellt werden. Je nach Nutzeraktion kann daraufhin entschieden werden, welche Inhalte in der AR abgebildet werden und welcher Prozesszweig als nächstes gewählt wird. Abbildung 1.1 veranschaulicht diesen Vorgang exemplarisch am Beispiel einer Safari Tour. Diese liegt als Prozesstemplate vor und wird über die AristaFlow Prozess-Engine ausgeführt. Je nach erhaltener Nutzerbewertung zu den Tieren, können die nächsten Stopps im Voraus geplant werden. Auf diese Weise können dynamische, auf die Umgebung und den Benutzer spezifisch angepasste Informationen in der AR angezeigt werden. In Abbildung 1.2 wird anhand zwei Mockups gezeigt, wie der Nutzer einer solchen Anwendung auf einer Safari Tour Informationen zu Tieren per AR beziehen kann.

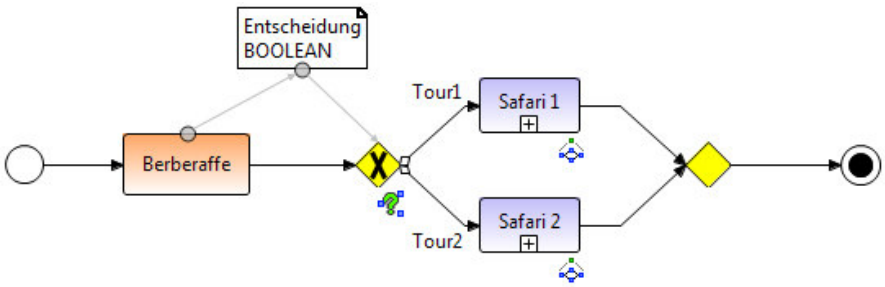
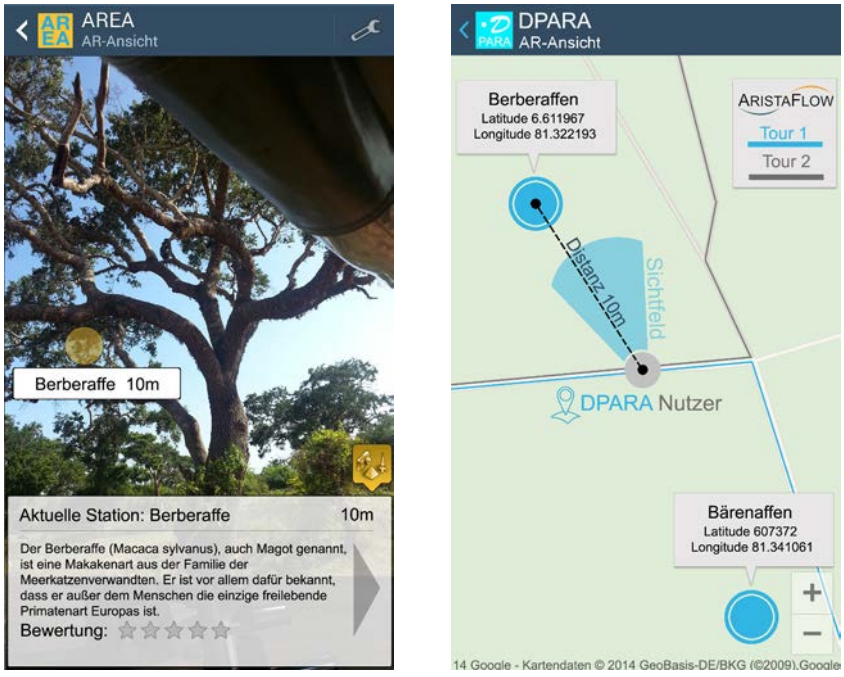


Abbildung 1.1.: Entscheidungszweig im Safari Tour Prozess

Die Position des Nutzers kann Auslöser für einen Prozess sein oder ist die Bedingung, um diesen erfolgreich abzuschließen. Die Einsatzgebiete von AR sind weitreichend, so kann man z.B. auch den Einbau eines Motors graphisch unterstützen [19], oder bei einem Paketlieferdienst die Navigation und Orientierung bei der Paketannahme und Auslieferung entscheidend optimieren.



(a) AR-Sicht mit Positionsanzeige

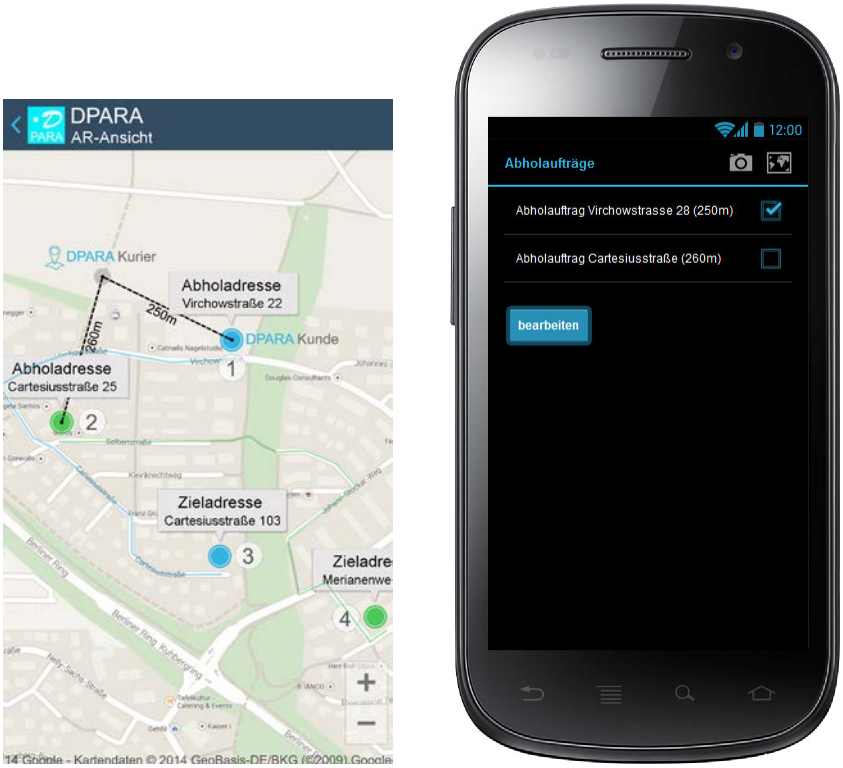
(b) Safari Tour wird durch einen Prozess gesteuert

Abbildung 1.2.: Safari Tour mit Augmented Reality auf dem Smartphone

1.1. Zielsetzung

Ziel dieser Arbeit ist es eine mobile Anwendung zu entwickeln, welche wesentliche Konzepte und Ideen umsetzt, Prozessaktivitäten mit Positionsdaten (z.B. eine Adresse oder Koordinaten eines Ortes) in einer AR anzeigen und bearbeiten zu können. Die Anwendung heißt **D**ynamic **P**rocess-oriented **A**ugmented **R**eality **A**pplication (DPARA) und basiert auf dem mobilen Betriebssystem Android (Version 4.4.2 „Kit Kat“). Sie wird exemplarisch am Beispiel eines Kurierdienstes implementiert (siehe Kapitel 3.2.2). Für dieses Szenario werden Features entworfen, die die Benutzer des Systems (Kunde und Kurier) optimal in ihren Aufgaben unterstützen. Ein Kurier kann über DPARA Abholaufträge eines Kunden direkt annehmen und bearbeiten. In einer Arbeitsliste sieht er alle notwendigen Informationen zu Aufträgen und die Entfernung zu diesen. Auf diese Weise

kann er immer Aufträge bearbeiten, die direkt auf seiner Wegstrecke liegen. Abbildung 1.3(a) zeigt, wie solch eine Planung aussehen könnte. In diesem Beispiel würde der Kurier zwei Aufträge akzeptieren und in der angegebenen Nummerierung abarbeiten. Für die Orientierung und Navigation zu Auftragsadressen verwendet der Kurier die von DPARA angebotene Karten- und AR-Ansicht.



(a) Kartenansicht (b) Listenansicht

Abbildung 1.3.: Auftragsdarstellung in Karten- und Listenansicht

Die Prozessausführung und Datenbereitstellung geschieht über einen Server mit einer umfangreichen AristaFlow BPM Suite [2]. Prozessvorlagen werden hier initiiert und über die AristaFlow Prozess-Engine ausgeführt. Die Applikation kommuniziert über ausgewählte Webservice-Methoden und präsentiert dann die erhaltenen Informationen dem Nutzer. Schon bei der graphischen Modellierung von Prozessen im AristaFlow Process Template Editor kann der Nutzer angeben, ob in einer Aktivität eine Position

erfasst werden soll (siehe Abbildung 1.4). DPARA fungiert als mobiler Workflow-Client und bietet alle dafür notwendigen Funktionen an. Des Weiteren wird der AristaFlow OrgModel Editor verwendet, um ein Organisationsmodell zu erstellen, welches die in DPARA notwendigen Rollen und eine komplexe Bearbeiterzuordnung für Aktivitäten in der Prozessvorlage ermöglicht.

Abbildung 1.4.: Konzeptübersicht

Nutzer von DPARA können Prozesse starten, pausieren oder unter Angabe eines Grundes beenden. Die Prozessliste wird bei Bedarf automatisch aktualisiert und kann nach Name, Aktivierungszeit, Distanz und Priorität sortiert werden. Ändert sich der Zustand einer Prozessaktivität, wird der Nutzer über diesen benachrichtigt. Für die visuelle Darstellung wird die **A**ugmented **R**eality **E**ngine **A**pplication (AREA) [4, 5] verwendet, eine an der Universität Ulm eigens entwickelte AR-Engine. Diese ermöglicht das Anzeigen von Interessenspunkten aus der Umgebung eines Benutzers, anhand des Blickwinkels, seiner Ausrichtung und Position. Für die Verwendung in DPARA muss diese so angepasst werden, dass alle für den Nutzer wichtigen Informationen zu einer Prozessaktivität in der AR-Sicht abrufbar sind und dabei die Übersicht immer gewahrt bleibt. Zu diesem Zweck ist es notwendig ein Konzept zu entwickeln, dass die Vorteile von AR mit denen einer klassischen Ansicht (z.B. Listenansicht) verbindet. Der Wechsel zwischen den Ansichten soll fließend sein. Die aktuell zu bearbeitende Prozessaktivität muss in der AR-Sicht schnell gefunden werden und wird aus diesem Grund auf geeignete Weise hervorgehoben. Beim Navigieren in der AR-Sicht kann der Nutzer die zu bearbeitende Aktivität aus dem Sichtfeld verlieren. Es gilt ein Konzept zu entwickeln, welches ihm die Richtung anzeigt, in die er das Smartphone bewegen soll, um die Aktivität zurück ins Sichtfeld zu holen. Die wichtigsten Prozessdetails sind für den Nutzer immer sichtbar. Aktivitäten, welche sich in der AR-Ansicht überlagern und dadurch schlecht über den Touchscreen selektierbar sind, können über eine alternative Selektionsmethode ausgewählt werden.

1.2. Related Work

Es gibt bereits eine Vielzahl an Apps zum Thema AR im Android Market sowie Apple Store. Diese können grob in zwei Kategorien unterteilt werden: AR-Browser und Bilderkennungs-Anwendungen [18]. Die Kombination mit Prozessen findet man zum aktuellen Zeitpunkt noch selten. In diesem Kapitel werden zwei Vertreter, welche AR in Prozessanwendungen integrieren, näher beschrieben. Im Video von Atos & RS [25] wird eine Anwendung vorgestellt, die es dem Kunden ermöglicht von Werbeanzeigen direkt mit dem Smartphone Artikelinformationen einzuholen und sogar Bestellvorgänge zu täti-

gen. Zusätzlich bekommt er eine 3D-Ansicht zum Artikel, welche durch eine AR-Engine auf die Anzeige virtuell projiziert wird (siehe Abbildung 1.5). Die Artikelinformationen werden direkt von einem Datenservice bezogen und sind damit immer aktuell. Bestandskunden können über ihr Kundenprofil Massenbestellungen aufgeben und bekommen auf bestellte Artikel eventuell einen Preisnachlass. Wird ein Bestellprozess gestartet, bekommt der zugeteilte Handelsvertreter eine Benachrichtigung und kann den Vorgang mit den Kundendaten überprüfen. Die Daten werden über ein SAP CRM (Customer Relationship Management) System bezogen. Auf diese Weise können umfangreiche Analysen und Statistiken zum Kunden abgerufen werden. Wird das System zusätzlich mit Outlook verknüpft, kann auch auf Terminlichkeiten des Kunden bei der Überprüfung eingegangen werden.

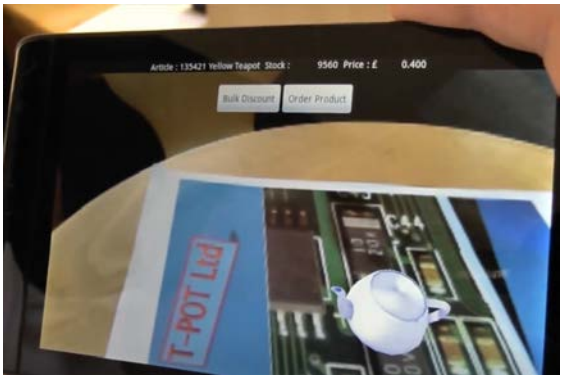


Abbildung 1.5.: Artikel über AR bestellen [25]

Nachdem der Antrag an einen Manager weitergeleitet und durch diesen überprüft wurde, bekommt der Kunde eine Benachrichtigung über den bestätigten Preisnachlass und kann den Bestellvorgang abschließen. Diese Funktionen sind über die native Anwendung nicht erreichbar. Der Nutzer wird auf eine Webseite weitergeleitet, welche die benannten Informationen mit Hilfe von HTML5 und jQuery darstellt. Die Anzeige von Informationen in der AR wird in diesem Beispiel durch ein Plakat mit einem produktspezifischen Erkennungscode ermöglicht, diese werden durch den Bestellprozess nicht beeinflusst. Die AR-Funktion dient lediglich als visueller Einstieg. In DPARA sollen, die in der AR

angezeigten Informationen, durch den Prozess gesteuert werden.
Das zweite Beispiel findet Anwendung im Industriesektor. Die Firma Augmensys setzt auf den Einsatz von Augmented Reality bei produzierenden Unternehmen und deren Zulieferern [15]. Das österreichische Unternehmen kooperiert dabei mit Microsoft und im Zuge dessen wird die Applikation auf Windows 8 basierende Endgeräte optimiert. Der Vorteil von AR in diesem Bereich ist, dass Maschinen und Bauteile identifiziert werden können und Informationen dazu sofort abrufbar sind [12]. Das Smartphone oder Tablet wird dazu einfach auf den Gegenstand gerichtet (siehe Abbildung 1.6).

Abbildung 1.6.: Augmented Reality Anwendung des Startup Augmensys [15]

Zusätzlich soll das System Mitarbeiter motivieren, Daten häufiger einzutragen als bisher (z.B. Foto oder Sprachnotiz). Mit diesem Konzept können auch Laufwege über die AR-Sicht eingeblendet werden, um Gefahrenzonen in einer Fabrik zu umgehen. Dadurch soll ein höheres Maß an Sicherheit für die Mitarbeiter gewährleistet werden. Eine größere Herausforderung bei diesem Ansatz ist laut Jürgen Kneidinger, Geschäftsführer der Augmensys GmbH, die Indoor-Navigation. Es gibt in Fabrikgebäuden viele Störfaktoren wie Betonmauern, schwere Maschinen und magnetische Strahlung, welche das Positionssignal negativ beeinflussen. Somit ist die alleinige Positionsangabe über das GPS des Tablets nicht ausreichend zuverlässig. Augmensys arbeitet hier an einer eigenen Lösung, um diesen Qualitätsfaktor selbst steuern zu können.

1.3. Aufbau der Arbeit

In diesem Abschnitt wird der Aufbau dieser Arbeit beschrieben. Diese unterteilt sich insgesamt in neun Kapitel (siehe Abbildung 1.7). Nach einer kurzen Motivation zum Thema „Augmented Reality in prozessorientierten mobilen Anwendungen“ und der Zielsetzung der Arbeit, werden in der Einleitung (Kapitel 1) im Abschnitt Related Work zwei Anwendungen vorgestellt, welche AR in mobilen Prozessanwendungen integrieren. Kapitel 2 listet die Anforderungen an DPARA, hinsichtlich seiner Funktion als mobiler Workflow-Client und des gewählten Anwendungsszenarios auf und erläutert diese im Detail. Am Schluss des Kapitels wird auf die verwendeten Werkzeuge und die Entwicklungsumgebung eingegangen. Im Anschluss folgt ein Kapitel über den Entwurf des Projekts (Kapitel 3). In diesem wird zunächst der Architekturentwurf vorgestellt. Des Weiteren werden drei Szenarios aus der Vorauswahl und das letztlich gewählte Kurierdienstszenario vorgestellt. Am Schluss des Kapitels werden die Anwendungsfälle, ein genereller Ablauf und der Prozessentwurf im Detail erläutert. Kapitel 4 zeigt die finale Architektur der entstandenen Anwendung DPARA und dessen Paket-, Klassen- und Dialogstruktur. Im Anschluss wird in Kapitel 5 auf die Implementierung der Anwendung im Detail eingegangen. Dies umfasst die Kommunikation mit dem AristaFlow Webservice, das Datenmanagement, die Datenstrukturen von DPARA und die Präsentationsschicht bestehend aus der Aktivitäten-, Listen-, Karten- und Kameraansicht. Die erstellte Anwendung wird in Kapitel 6 anhand eines Ausschnitts aus dem Kurierdienstszenario mit Hilfe von Screenshots und Fotos vorgestellt. Kapitel 7 gleicht die gestellten Anforderungen an das Projekt ab und verweist auf die jeweiligen Stellen im Dokument. Daraufhin werden in Kapitel 8 die im Projektverlauf aufgetretenen Herausforderungen und Problemstellungen und deren Lösungen beschrieben. Das letzte Kapitel 9 gibt eine Zusammenfassung über das ganze Projekt und gibt einen Ausblick hinsichtlich der Zukunft von Anwendungen, welche AR zur Anzeige und Bearbeitung von Prozessdaten verwenden. Zusätzlich wird auf zukünftig mögliche Features von DPARA eingegangen.

Abbildung 1.7.: Aufbau der Arbeit

2

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an DPARA aufgelistet und im Einzelnen erläutert. Dabei werden diese in **funktionale** und **nichtfunktionale** gegliedert. DPARA soll die Möglichkeiten eines mobilen Workflow-Clienten [20] mit der von AREA bereitgestellten AR-Funktionalität verbinden. Zusätzlich ergeben sich weitere spezifische Anforderungen, da die Anwendung am Beispiel eines Kurierdienstes implementiert wird und dafür einige Features bereitstellen muss. Dementsprechend wurden die Anforderungen in zwei Kategorien unterteilt.

2.1. Anforderungen

Prozesse, die mit einem **P**rocess **L**ocation **I**nput (PLI) (siehe Kapitel 5.3.5) verknüpft sind, können auf einer Karte und im Kameramodus des Smartphones angezeigt werden.

AREA muss für die Anzeige von Prozessinformationen angepasst werden. Alle Informationen, die für den aktuellen Prozessschritt relevant sind, sollen für den Nutzer abrufbar sein, ohne dabei die Übersicht zu verlieren. Neben der Karten- und Kameraansicht soll der Nutzer die Möglichkeit besitzen, Prozesse in einer klassischen Listenansicht betrachten zu können. Diese Kombination erzielt ein Maximum an Effizienz und Benutzerfreundlichkeit und erleichtert den Einstieg in die AR-Sicht. Weiterhin wird auf diese Weise gewährleistet, dass dem Nutzer das vollständige Funktionsspektrum zur Anzeige und Bearbeitung der Prozessaktivität zur Verfügung steht. In allen Ansichten zeigt DPARA die Entfernung des Nutzers zum jeweils hinterlegten PLI an. Neben der Prozessaktivierungszeit und der Prozesspriorität, kann auch nach der Distanz sortiert werden. In der AR-Sicht soll der Nutzer den maximalen Radius einstellen können, in dem Objekte für ihn sichtbar sind. Bei vielen Informationen ist dies essentiell um nicht den Überblick zu verlieren. Process Location Inputs sollen auf einer Karte mit einem Marker, welcher für den Prozess eine eigens bestimmte Farbe besitzt, gekennzeichnet werden. Bei mehreren PLIs wird zusätzlich eine Route gezeichnet, welche diese miteinander verbindet. Die Route orientiert sich hierbei am Straßenverlauf. Ein PLI soll über Konfigurationsvariablen im Verhalten modifizierbar sein. Diese werden über den AristaFlow Process Template Editor angegeben. Es soll die Möglichkeit bestehen Prozesse, die nicht im direkten Umfeld des Nutzers sind, zunächst für die Bearbeitung sperren zu können. Zudem soll der Nutzer auf Prozesse, welche in seiner Nähe sind, hingewiesen werden. DPARA soll auf ähnliche Anwendungsszenarien, welche Positionsangaben im Prozess enthalten, leicht adaptiert werden können. Die Anzeige von Ein- und Ausgabeparametern wird dynamisch generiert. Da die Daten über einen Webservice bezogen werden, muss DPARA die Abfragen so steuern, dass maximale Performance gewährleistet ist. AristaFlow bietet hierfür die Möglichkeit Prozessinformationen mit Revisionsnummern abzugleichen. Die Revisionsnummer wird in DPARA verwaltet und bei jeder Abfrage mitgesendet. Auf dieses Weise kann die Menge der zu transferierenden Daten minimiert werden. Tabelle 2.1 zeigt weitere Anforderungen an DPARA in seiner Funktion als mobiler Workflow-Client in der Übersicht.

Nr.	Anforderung	Beschreibung
1	Anzeige und Modifizierung von Prozessen in der Listen-, Karten und AR-Ansicht **(funktional)**	DPARA kann Prozesse mit PLIs auf einer Karte, im Kamera- und Listenmodus anzeigen. Für die Kameraansicht wird die AR-Engine AREA verwendet.
2	Übersichtlichkeit in der AR-Sicht **(nichtfunktional)**	Da der Anzeigeplatz in der AR-Ansicht stark begrenzt ist, können nicht alle Prozessinformationen angezeigt werden, ohne dass sich dabei die Übersichtlichkeit bzw. Bedienbarkeit verschlechtert. Eine detaillierte Ansicht / Bearbeitung kann in der Listenansicht erfolgen.
3	Schema zur Kennzeichnung bzw. Zuordnung von Positionsdaten einer Aktivität im AristaFlow Process Template Editor **(funktional)**	DPARA soll automatisch erkennen, ob in einer Aktivität eine Positionsangabe als Eingabe benötigt wird oder zur Ausgabe bereit steht. Zu diesem Zweck muss ein Parameter speziell gekennzeichnet werden können. Der Parameter ist zusätzlich über Konfigurationsvariablen im Anzeigeverhalten modifizierbar.
4	Nachrichtenaustausch zwischen Smartphone und dem AristaFlow Webservice **(funktional)**	In DPARA soll eine Schnittstelle für die Kommunikation mit einem AristaFlow Webservice bereitgestellt werden. Diese soll leicht erweiterbar sein, um neue Funktionen zur Prozessverarbeitung zur Verfügung zu stellen.
5	Anzeige der Distanz zu einem PLI **(funktional)**	DPARA soll die Distanz zwischen einem PLI und der aktuellen Nutzerposition anzeigen können.

6	Filterung der Prozessinstanzen nach Distanz, Aktivierungszeit und Priorität **(funktional)**	Über ein Menü kann der Nutzer die Prozessinstanzen sortieren, um eine bessere Übersicht zu erhalten.
7	Stabilität und Eingabevalidierung **(nichtfunktional)**	Tritt in DPARA ein Eingabefehler auf bzw. wird eine nichtoptionale Eingabe vergessen, wird der Nutzer vom System darauf hingewiesen. Auch bei Netzwerkproblemen oder spezifischen Webservicefehlern bekommt der Nutzer die für ihn notwenigen Hinweise.
8	Anpassbarkeit **(nichtfunktional)**	DPARA soll so dynamisch programmiert werden, dass es auf ähnliche Anwendungsszenarien leicht adaptiert werden kann.
9	Gute Bedienung, Verständlichkeit, Erlernbarkeit **(nichtfunktional)**	Viele Nutzer sind mit der AR-Sicht nicht vertraut. Um den Einstieg zu erleichtern, werden bereits bekannte Elemente (z.B. Listenansicht) als Einstiegs- und Orientierungshilfe angeboten. Bei der Prozessbearbeitung soll der Nutzer bei wichtigen Schritten das notwenige Feedback erhalten, um seine Aufgabe positiv abschließen zu können.
10	Orientierungsunterstützung in der AR-View **(funktional)**	Der Nutzer soll in der Kameraansicht beim Auffinden eines PLIs von DPARA unterstützt werden.
11	Unterstützung von Subprozessen **(funktional)**	DPARA soll Prozessvorlagen mit Subprozessen ausführen können.

12	Möglichkeit in der AR-View zwischen PLIs zu wechseln **(funktional)**	Der Nutzer soll nicht die Kameraansicht verlassen müssen, um sich verschiedene PLIs der für ihn zugeteilten Aktivitäten anzuschauen.
13	Minimalistisches Design **(nichtfunktional)**	Das Design soll so gewählt werden, dass es für mehrere Anwendungsszenarien nutzbar ist.

Tabelle 2.1.: Anforderungen an DPARA als mobiler Workflow-Client mit AR-Funktion

In DPARA kann ein Kunde Abholaufträge erstellen, die von einem Kurier direkt bearbeitet werden. Die Erfassung der Adressinformationen geschieht in der Anwendung über ein Karten-Widget. Bei der Paketannahme wird die Unterschrift des Kunden gleichermaßen über ein von DPARA gestelltes Widget erfasst. Ein vom System generierter Authentifizierungscode sorgt zusätzlich bei der Annahme und Abgabe für die notwenige Sicherheit. Sollte der Kunde mit dem vom Kurier vorgeschlagenen Abholdatum nicht zufrieden sein, besitzt er die Möglichkeit jenes unter Angabe eines Grundes und eines Wunschabholdatums, neu zu verhandeln. Damit der Kurier seine Aufträge optimal planen kann, wird von DPARA eine Route bei Aufträgen eingezeichnet. Tabelle 2.2 zeigt diese Anforderungen an DPARA in seiner Funktion als Kurierdienst Anwendung in der Übersicht.

Nr.	**Anforderung**	**Beschreibung**
1	Auswahl der Adressen über eine Kartenansicht **(funktional)**	Ein Kunde soll die Möglichkeit besitzen Abhol- und Zieladresse über ein eine Kartenansicht auszuwählen. DPARA soll für dieses Feature ein Widget stellen.
2	Erfassen einer Kundenunterschrift **(funktional)**	Um eine Paketabgabe bestätigen zu können, muss der Kurier über DPARA eine Kundenunterschrift erfassen können. Zu diesem Zweck wird ein Unterschriften-Widget bereitgestellt.

3	Authentifizierung des Kuriers **(funktional)**	Damit der Kunde einen Kurier identifizieren kann, erstellt DPARA dynamisch einen Authentifizierungscode.
4	Neuverhandeln des Abholdatums **(funktional)**	Wurde ein Abholauftrag von einem Kurier akzeptiert, schlägt dieser ein Abholdatum vor. Der Kunde kann dieses, unter Angabe eines Grundes und eines Wunschabholdatums, ablehnen. Dieser Vorgang wird als Subprozess umgesetzt.
5	Anzeige der Route zwischen PLIs **(funktional)**	Damit der Kurier die Distanz bei einem Auftrag besser abschätzen kann, soll DPARA die Route zwischen zwei PLIs oder einem PLI und der aktuellen Nutzerposition anzeigen können.

Tabelle 2.2.: Anforderungen bezüglich des Kurierdienst Szenarios

2.2. Werkzeuge und Entwicklungsumgebung

Als Entwicklungsumgebung wird Eclipse Java EE IDE für Web Entwickler in der Version Juno Service Release 2 verwendet. Die Applikation wird für das Android Framework in der Version 4.4.2 (Codename „Kit Kat“ – API 19) entwickelt und optimiert, wobei mindestens das API Level 16 vorausgesetzt wird. Für die Prozesserstellung, Verwaltung und das Prozess Monitoring wird die AristaFlow BPM Suite verwendet [2]. Die Applikation kommuniziert über das **S**imple **O**bject **A**ccess **P**rotocol (SOAP) mit der AristaFlow Webservice Schnittstelle, hierfür wird clientseitig die kSOAP2 Bibliothek in der Version 2.6.1 verwendet [16]. Um SOAP Nachrichten analysieren zu können, wird in Eclipse ein TCP/IP Monitor eingerichtet und zusätzlich das Programm SOAP-UI verwendet [24]. Dieses ermöglicht eine detaillierte Ansicht der Webservicespezifikation und das Testen spezifischer Operationen des Webservice.

3

Entwurf

In diesem Kapitel wird der Entwurf der Applikation DPARA vorgestellt, welcher die wichtigsten Bauteile und deren Zusammenspiel beschreibt. Danach wird aufgeführt, welche bereits existierenden Komponenten in DPARA zum Einsatz kommen. Des Weiteren wird das Prozesstemplate erläutert, welches DPARA von der Serverseite aus steuert. Am Schluss werden ausgewählte Szenarien untersucht, die für den Einsatz von AR in der Prozessbearbeitung geeignet sind und anhand von Mockups veranschaulicht.
Zu Beginn der Arbeit wurden ausführliche Gespräche mit Fabian Maier und Philip Geiger, zwei Studenten der Universität Ulm, geführt. Fabian konnte hinsichtlich seiner Bachelorarbeit [13] zur Thematik „Kommunikation mit dem AristaFlow Webservice“ wertvolle Tipps geben. Er verfolgt in seiner Arbeit den Ansatz, über einen XML-Parser Antworten des Servers manuell zu verarbeiten. Für DPARA soll zu diesem Zweck die effiziente Simple Object Access Protocol Client Bibliothek `ksoap2` verwendet werden [16]. Diese wurde bereits erfolgreich in meiner Bachelorarbeit zur Kommunikation mit einem `Axis2`

Webservice verwendet [3]. Im Gespräch mit Philip Geiger konnten wichtige Fragen zum Funktionsumfang, der in seiner Bachelorarbeit entstandenen AR-Engine AREA [4, 5], geklärt werden. Dieser ist für DPARA optimal und deckt alle gestellten Anforderungen (siehe Kapitel 2) ab.

3.1. Architekturentwurf

Abbildung 3.1.: Architekturentwurf

Der Architekturentwurf von DPARA beschreibt drei wichtigen Komponenten. Die erste Komponente ist das Kommunikationsmodul, welches DPARA zum Nachrichtenaustausch mit dem AristFlow Webservice benötigt. Die zweite Komponente sind die DPARA-Klassen selbst, welche wichtige Funktionen wie Login, Filterung von Prozessen etc. zur Verfügung stellen. Die letzte Komponente ist AREA, welche dafür zuständig ist PLIs auf der

Karten- und Kameraansicht darzustellen. Alle Module müssen untereinander Informationen austauschen können und besitzen dementsprechend eine gemeinsam nutzbares Datenmodell. DPARA wird nach dem Model View Controller Muster strukturiert werden.

3.2. Auswahl eines Szenarios

Bei der Suche nach einem geeigneten Anwendungsszenario dienen folgende Punkte als Auswahlkriterium:

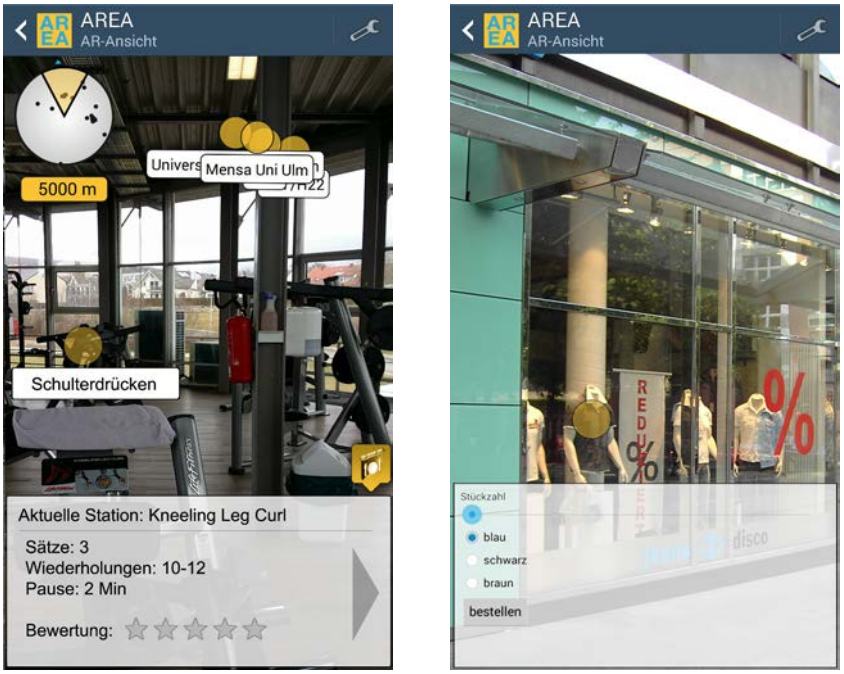
- Szenario ist prozessorientiert
- Positionsangaben sind wesentlicher Bestandteil einer zu bearbeitenden Aktivität

Es werden vier konkrete Szenarios und deren Mockups (siehe Abbildungen 3.2, 3.3) vorgestellt. Diese zeigen die jeweilige Anwendung in der von AREA generierten Kameraansicht.

3.2.1. Vorauswahl

Abbildung 3.2(a) zeigt das Szenario **Fitnessstudio**. In diesem kann der Nutzer mit Hilfe von DPARA einen Trainingsplan, der über ein Prozesstemplate in AristaFlow umgesetzt wird, interaktiv durchlaufen. Sämtliche Trainingsgeräte erhalten eine Positionsangabe (z.B GPS-Koordinate) und können somit in der AR-Sicht angezeigt werden. Der Nutzer bekommt zu jeder Trainingsstation Informationen, wie den Name der Station, Anzahl der Sätze, Anzahl der Wiederholungen und eine Pausenempfehlung angezeigt. Am Schluss kann er die Übung bewerten. Dieses Szenario wurde nicht gewählt, da die Navigation mit dem Smartphone in einem Gebäude, ohne weitere Technik, noch zu ungenau ist. Bei fehlendem GPS-Signal können in der AR-Sicht die Markierungen für die PLIs nicht richtig gesetzt werden (siehe Mensa Uni Ulm Markierung in Abbildung 3.2(a)). Für die Indoor-Navigation gibt es bereits Lösungen. Apples iBeacon ist ein proprietärer Standard für die Navigation in geschlossenen Räumen, basierend auf Bluetooth Low Energy. Mit diesem Konzept kann der Abstand zum Nutzer in vier Kategorien unterteilt

werden (größer 30m, bis 30m, bis 2m, bis 50cm) [11]. Für DPARA wäre aber gerade bei einem Abstand von zwei bis 30 Metern ein genauer Distanzwert wichtig, damit AREA die PLIs richtig platzieren kann.

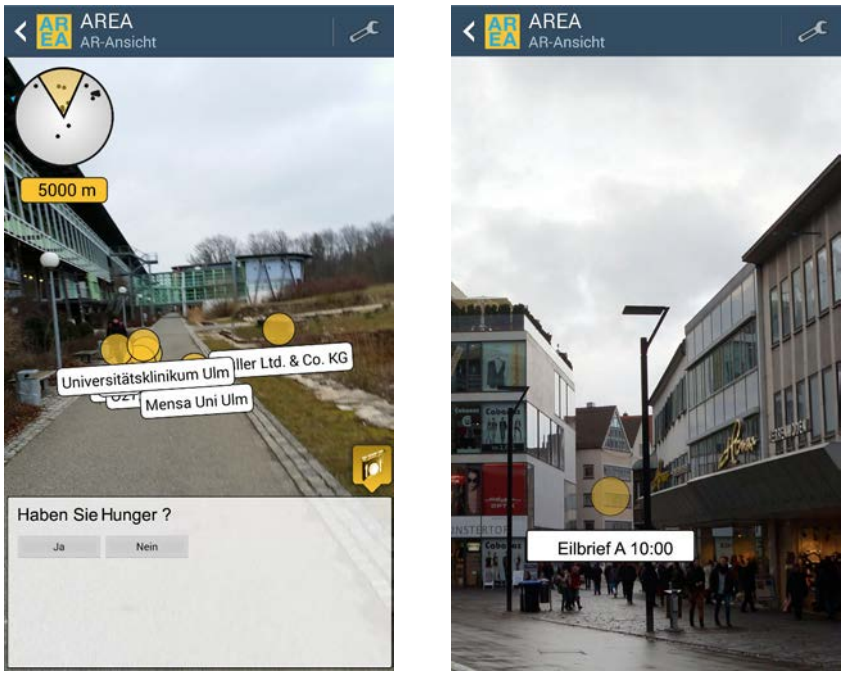


(a) Fitnessstudio (b) AR-Einkaufen

Abbildung 3.2.: Mockups zum Fitnessstudio und AR-Einkaufen

Abbildung 3.2(b) beschreibt einen klassischen **Einkaufvorgang**. Der Nutzer kann über sein Smartphone Schaufensterartikel in der AR-Sicht inspizieren und bei Bedarf bestellen, ohne dabei selbst den Laden betreten zu müssen. Der Kunde kann den Artikel einsehen, speichern, mit anderen Artikeln vergleichen und mit Onlinebezahlungsmethoden kaufen. Das bezahlte Produkt kann nun, falls es in entsprechender Farbe und Größe vorhanden ist, direkt aus dem Laden mitgenommen, oder per Post an die Heimadresse verschickt werden.

Abbildung 3.3(a) beschreibt einen **Mensaradar**. Der Nutzer kann in der Applikation angeben zu welcher Zeit er essen gehen möchte und spezielle Vorlieben angeben (z.B. Italienisch). Das System prüfte mit den gegebenen Parametern, ob sich eine geeignete Mensa in seiner Umgebung befindet.



(a) Mensaradar (b) Kurierdienst

Abbildung 3.3.: Mockups zum Mensaradar und Kurierdienst

3.2.2. Kurierdienst Szenario

Abbildung 3.3(b) zeigt einen **Kurierdienst**. Nutzer von DPARA können als **Kunde** Abholaufträge erstellen. Diese können von einem **Kurier** direkt eingesehen und angenommen werden. Die Rechteverwaltung übernimmt dabei AristaFlow, welches die zwei genannten Rollen Kunde und Kurier im Organisationsmodell enthält. DPARA erfasst zu diesem Zweck in einer Login-Prozedur Benutzernamen und Password und schickt diese an den Server. Dieser prüft die Aufgaben und Rechte des Nutzers und stellt daraufhin alle vor-

handenen Informationen zur Verfügung. Beim Erstellen eines Abholauftrags werden der Aktivität durch den Nutzer Positionsdaten hinterlegt, welche von DPARA genutzt werden um diese in der Karten- und der AR-Sicht darstellen zu können. Für den Abholauftrag werden folgende Informationen zwingend benötigt:

- Abholadresse
- Zieladresse
- Abholdatum
- Lieferdatum

Der Kurier kann nun mit Hilfe der AR-Funktion zum Abholort navigieren. Dort übergibt der Kunde dem Kurier das Paket und signiert die Abholbestätigung mit seiner Unterschrift. DPARA bietet hierfür ein Unterschriften Widget an. Der Kurier kann nun entweder über die Karten, oder über die AR-Ansicht zum Zielort navigieren (siehe Abbildung 3.3(b)). Nachdem die Zielperson das Paket entgegengenommen hat, kann der Kurier den Auftrag positiv abschließen. In DPARA kann eingestellt werden, dass der Kurier sich in der direkten Nähe des Zielorts befinden muss, um die Aktivität beenden zu können. Der Kunde bekommt nun eine Bestätigung mit der Unterschrift der Zielperson angezeigt. Das Paket wurde erfolgreich ausgeliefert. Da die AR-Ansicht nicht alle Prozessinformationen zur gleichen Zeit darstellen kann, erhält der Nutzer zusätzlich die Möglichkeit in eine Listen- oder Kartenansicht zu wechseln, dies garantiert nicht nur die Übersicht, sondern hilft dem Nutzer sich zu orientieren. Jede der drei Ansichten unterstützt mit ihren spezifischen Möglichkeiten die Prozessbearbeitung, dabei ist der Übergang fließend. Die Datenhaltung bei DPARA geschieht zum größten Teil extern und wird über AristaFlow geregelt. Lediglich wenige Informationen werden auf dem mobilen Gerät selbst gespeichert (z.B. Filtereinstellungen, Benutzername). DPARA wurde so programmiert, dass ähnliche positionsbasierende Szenarien mit wenig Änderungsaufwand umgesetzt werden können.

3.3. Anwendungsfälle und genereller Ablauf

Auf Grundlage der in Kapitel 2.1 definierten Anforderungen, werden nun die wichtigsten Anwendungsfälle konkretisiert (siehe Abbildung 3.4). Die Hauptakteure sind der Kunde und der Kurier. Der Kunde kann im DPARA-System Abholaufträge erstellen, welche daraufhin vom Kurier eingesehen und bearbeitet werden. Als Eingabe wird zwingend eine Abhol- und Zieladresse benötigt. Es können mehrere Prozesse parallel existieren. Akzeptiert ein Kurier einen Auftrag, wird dieser für ihn reserviert. Abbildung 3.5 zeigt wie eine typische Interaktion zwischen Kunde und Kurier aussieht.

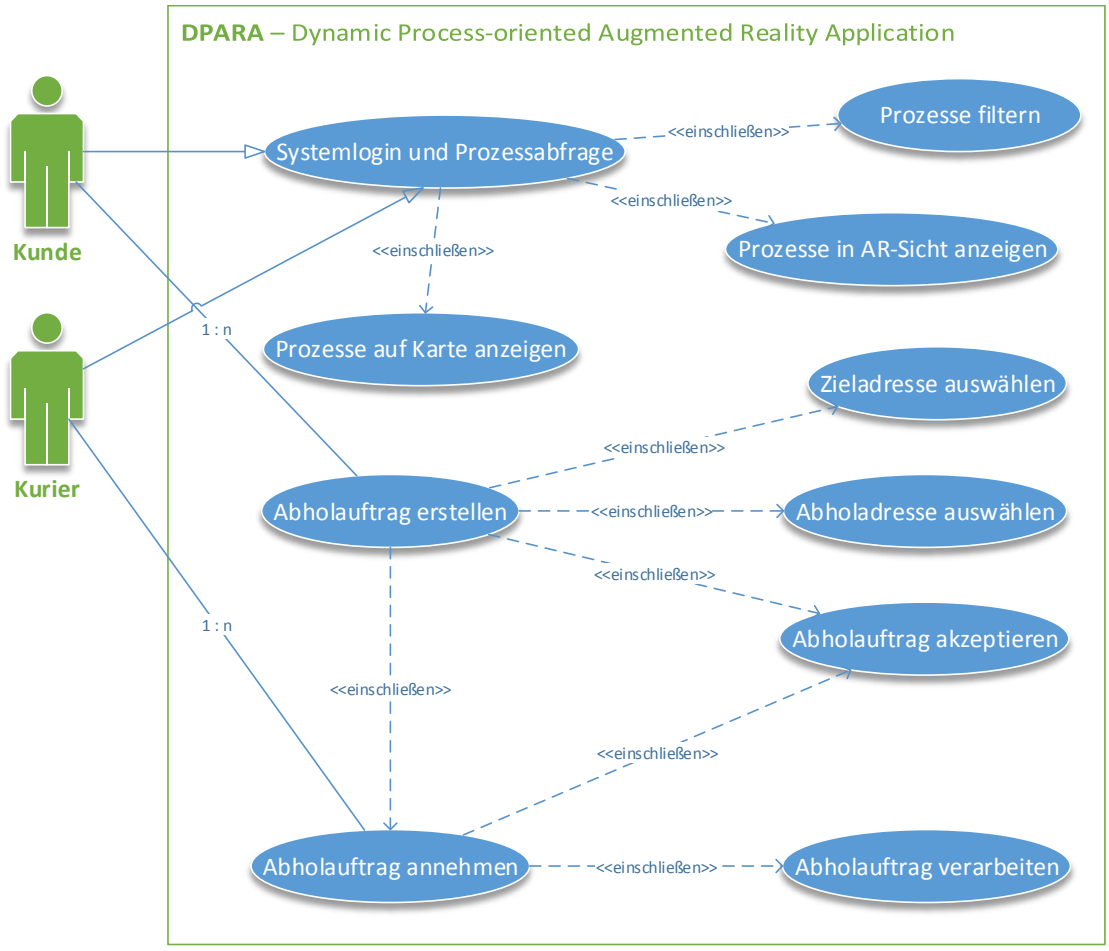


Abbildung 3.4.: Anwendungsfalldiagramm

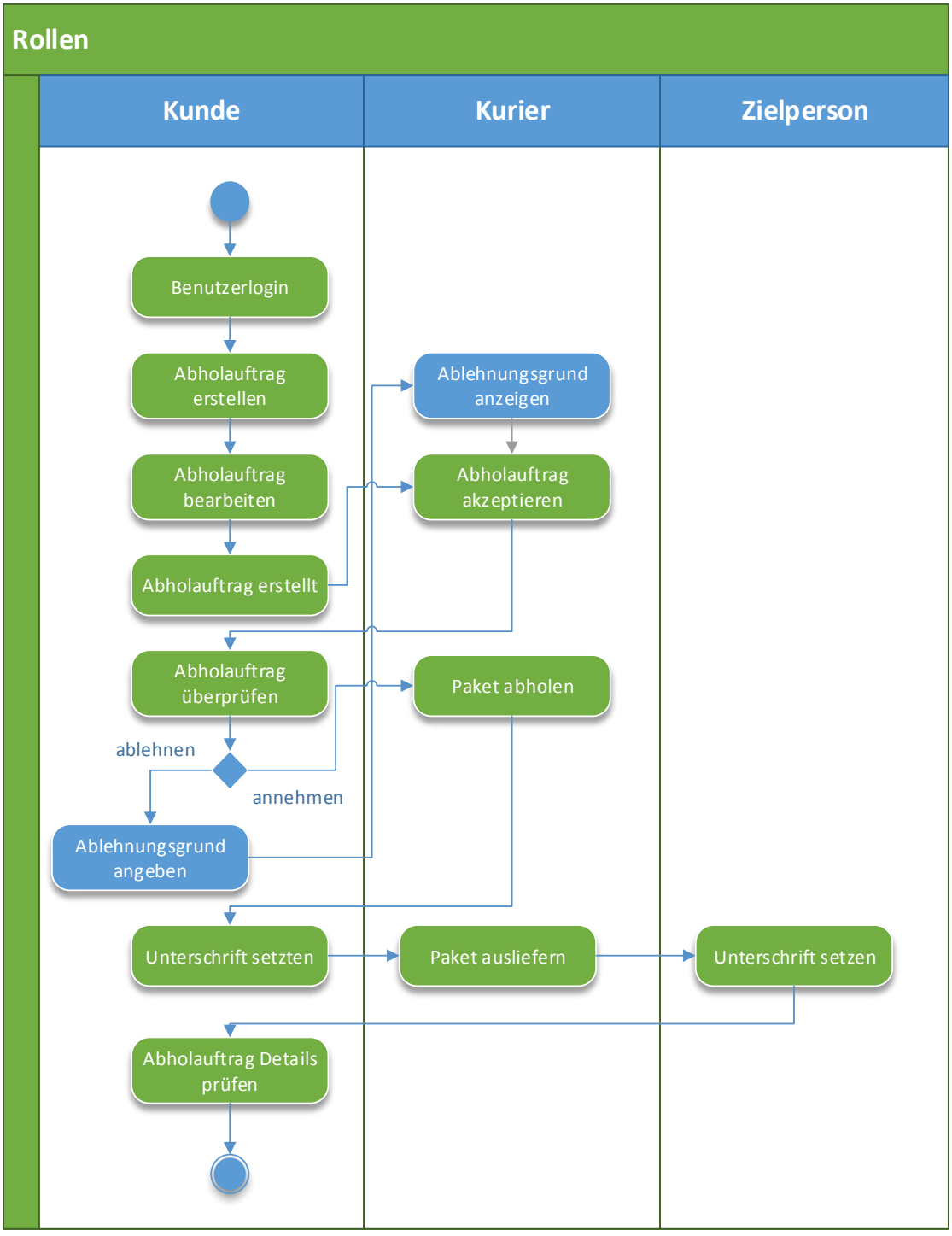


Abbildung 3.5.: Genereller Ablauf in DPARA

3.4. Prozessentwurf

Das im Kapitel 3.2.2 beschriebene Szenario wird im AristaFlow Process Template Editor modelliert. Abbildung 3.8 zeigt den kompletten Prozess in der Übersicht. Der Subprozess „Auftrag überprüfen“ wird in Kapitel 3.4.2 im Detail beschrieben.

3.4.1. Hauptprozess: Abholauftrag erstellen

Der Hauptprozess wird durch einen Kunden initiiert. Dieser muss in der ersten Aktivität Auskunft über das Wunschabholdatum und die Abholadresse geben. Weiterhin kann er angeben, ob es sich um eine Eillieferung handelt. In den ersten Skizzen und Anläufen der Prozesserstellung wurde die Adresse mit vier `STRING` Datenelementen umgesetzt. Der Nutzer musste Straße, Stadt, Ort und Postleitzahl übermitteln. Das Problem bei dieser Herangehensweise ist, dass bei mehreren Adressen in einer Prozessvorlage die Übersicht schnell verloren geht. Zudem müssen bei Operationen immer alle Felder gelesen oder geschrieben werden. In DPARA wurde deswegen ein `SelectLocation` Widget und ein Identifizierungszeichen für Parameter eingeführt (siehe Abbildung 3.6). Ein Datenelement vom Typ `STRING` kann somit als Adresse markiert werden und wird in DPARA durch das `SelectLocation` Widget (siehe Kapitel 5.4.1.2) verarbeitet.

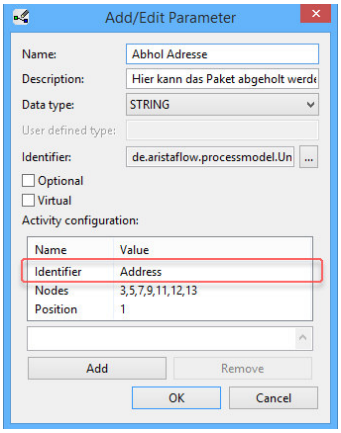


Abbildung 3.6.: Parameter als Adresse markieren

Über die Aktivitätenkonfiguration können weitere Attribute hinzugefügt werden, welche von DPARA dynamisch gelesen und verarbeitet werden. Diese Thematik wird im Detail in Kapitel 5.3.3 behandelt. Die Aktivität „Abholauftrag akzeptieren“ erhält nun die Abholadresse, Zieladresse, ein Wunschabholdatum und die Auskunft, ob es sich um eine Eillieferung handelt als Eingabe und erwartet als Ausgabe den Namen des Kuriers und ein Datumsvorschlag für die Paketabholung. Des Weiteren wird ein Authentifizierungscode (siehe Abbildung 3.7) von DPARA generiert, welcher zur Identifikation des Kuriers dient.

Abbildung 3.7.: Authentifizierungscode

An dieser Stelle wird nun der Subprozess „Auftrag überprüfen“ gestartet. Im Hauptprozess werden im Falle einer Ablehnung dem Kurier der Grund und das vom Kunden neu vorgeschlagene Abholdatum angezeigt. Bei erfolgreicher Auftragsvermittlung wird die Aktivität „Paket abholen“ gestartet, welche den Kurier dazu auffordert das Paket beim Kunden abzuholen. Die Aktivität kann nur mit der Erfassung der Kundenunterschrift vom Kurier positiv beendet werden. Die Unterschrift ist vom Typ `USERDEFINED` und wird mit dem Identifikationsparameter `Signature` gekennzeichnet. Nun muss der Kurier das Paket zur Zielperson transportieren. Die Aktivität „Paket ausliefern“ kann dann beendet werden, wenn die Unterschrift der Zielperson erfasst wurde und der Kurier sich in unmittelbarer Nähe (siehe Kapitel 5.4.3.3) des Zielorts befindet. Zum Schluss kann der Kunde in der Aktivität „Auftrag überprüfen“ die Unterschrift der Zielperson einsehen und den Abholauftrag erfolgreich abschließen. Aktivitäten welche mit der Ein- und Ausgabe von Parametern verbunden sind wurden mit der Aktivitätenvorlage `User Form` umgesetzt.

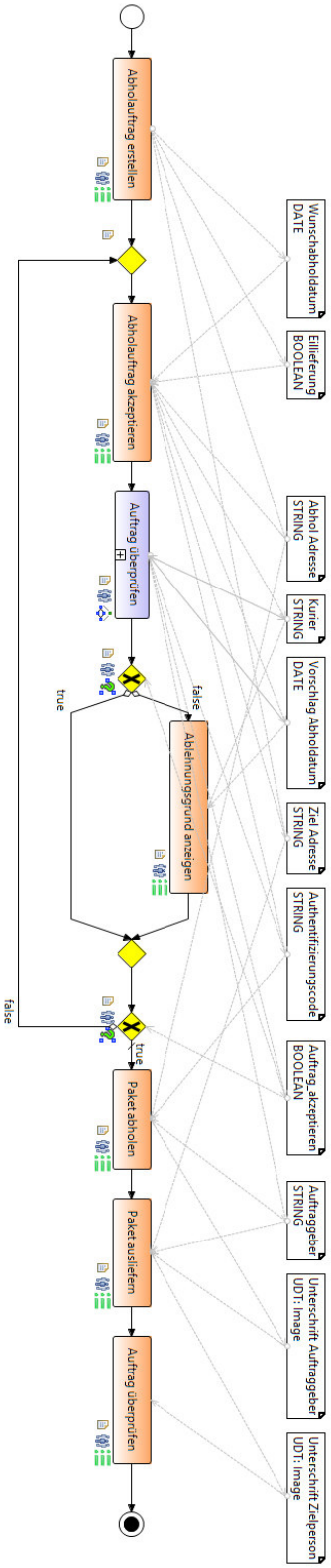


Abbildung 3.8.: Prozesstemplate Abholauftrag erstellen

3.4.2. Subprozess: Abholauftrag überprüfen

Hierfür eignet sich der Vorschlag eines Abholdatums durch den Kunden (Abbildung 3.9). Dieser Vorgang kann solange wiederholt werden bis ein geeignetes Datum dem Kunden vom Kurier präsentiert wurde. Zunächst werden im Subprozess die weitergereichten Daten empfangen. Zu diesen gehören der Name des Kuriers, das vorgeschlagene Abholdatum und der Authentifizierungscode. Der Kunde prüft diese und hat nun die Möglichkeit den Auftrag zu bestätigen oder abzulehnen. Bei einer Ablehnung gibt er ein neues Wunschabholdatum an und kann zusätzlich einen Ablehnungsgrund angeben. Der Kurier wird in beiden Fällen über das Ergebnis benachrichtigt.

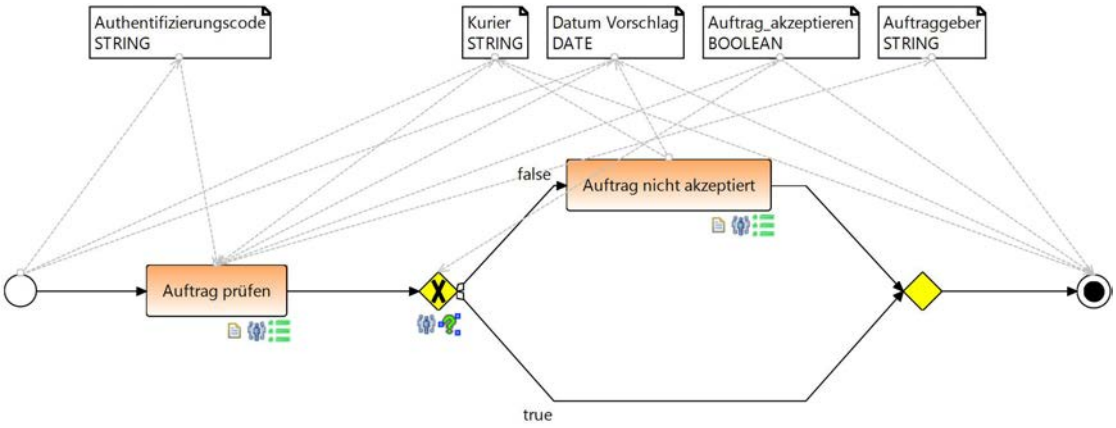


Abbildung 3.9.: Subprozess Abholauftrag überprüfen

Der fertige Subprozess kann nun in den Gesamtprozess (Abbildung 3.8) integriert werden. Mit Hilfe einer XOR-Aktivitätenvorlage wird das Ergebnis aus dem Subprozess verarbeitet und eine Entscheidung gefällt. Bei Ablehnung muss der Abholauftrag neu akzeptiert werden. Bei einer Bestätigung wird die nächste Aktivität „Paket abholen“ aufgerufen.

3.4.3. Organisationsstruktur

Über den AristaFlow OrgModel Editor wird in DPARA das Organisationsmodell erstellt. Dieses beinhaltet die in Abbildung 3.10 gezeigten Nutzer.

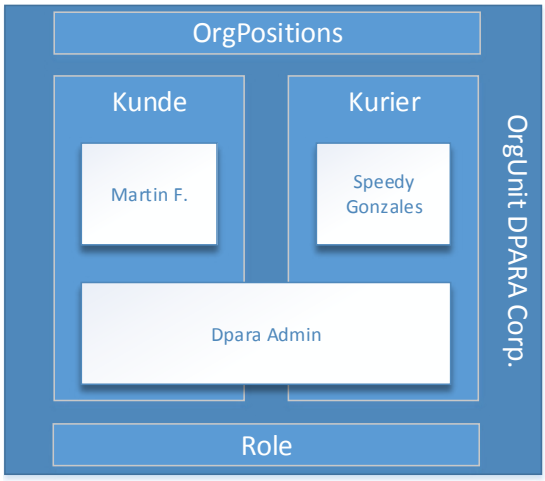


Abbildung 3.10.: Das Organisationsmodell

Jeder Aktivität in der Prozessvorlage wird eine sog. „Staff Assignment Role“ (dt. Mitarbeiterzuordnung) zugewiesen. Der Prozess „Abholauftrag erstellen “ kann in DPARA ausschließlich von einem Kunden gestartet werden. In diesem Fall von Martin F. und dem DPARA Administrator. Da es somit Nutzer gibt, welche beide Rollen erfüllen, wurde im Prozess dafür gesorgt, dass der Prozessersteller selbst seinen eigenen Auftrag nicht akzeptieren kann. In AristaFlow können in so einem Fall komplexe Benutzerzuordnungsregeln für Aktivitäten eingetragen werden (Siehe Listing 3.1).

```
1 OrgPosition(id=20) EXCEPT Agent(id=%i:InstanceInitiator-AgentID%)
```

Listing 3.1: Beispiel: Zuweisung einer Benutzeraufgabe

Sobald ein Kurier einen Auftrag akzeptiert hat, wird dieser für ihn vorgemerkt und kann ab diesem Zeitpunkt nur noch von ihm weiterbearbeitet werden.

4

Architektur

Nachdem in Kapitel 3.1 der Entwurf der Architektur beschrieben wurde, wird in diesem Abschnitt eine detaillierte Übersicht über einzelnen Bestandteile von DPARA gegeben. Es wird zusätzlich auf die komplette Paket- und Klassenstruktur eingegangen.

4.1. Architekturübersicht

Die folgende Abbildung 4.1 zeigt die fertige Architektur. Auf der Serverseite befindet sich die AristaFlow BPM Suite, über diese werden Prozessdaten abgerufen und gespeichert. Das Organisationsmodell ist hier hinterlegt und wird von der Anwendung für sämtliche Identifikationsvorgänge verwendet. Des Weiteren wird hier das Prozesstemplate initiiert und steht nun für den Nutzer zur Ausführung bereit. Über das Internet werden SOAP Nachrichten verschickt und empfangen, welche im DPARA Kommunikationsmodul verar-

beitet und an die Datenmanager weitergereicht werden. Diese sorgen dafür, dass die über die SOAP-Nachrichten erhaltenen Informationen, zur weiteren Bearbeitung optimal gebündelt werden. Die DPARA Komponenten stellen den Kern der Applikation dar und sorgen im Wesentlichen für die Generierung der Listenansicht von Prozessen und deren Bearbeitung. Sie stellen zusätzlich alle Widgets bereit, welche für das Kurierdienstszenario benötigt werden. Dazu gehören das Signatur- und Adress-Widget, welche beide im Kapitel 5.4 im Detail beschrieben werden. Die Hauptkonfigurationsklasse `Config` und weitere Hilfsklassen sind ebenfalls Bestandteil der DPARA Komponenten. Sämtliche Einstellungen bezüglich der Kommunikation und Informationsaufbereitung werden hier zentral verwaltet. Die AREA Klassen stellen die AR-Funktionalität, die `CameraView` und `MapView` (siehe Abbildung 4.2) zur Verfügung. Diese wurden so angepasst, dass ein optimaler Informationsaustausch mit den DPARA Klassen gewährleistet ist. Kapitel 4.3 gibt eine genaue Auskunft über die Dialogstruktur zwischen den genannten Aktivitäten.

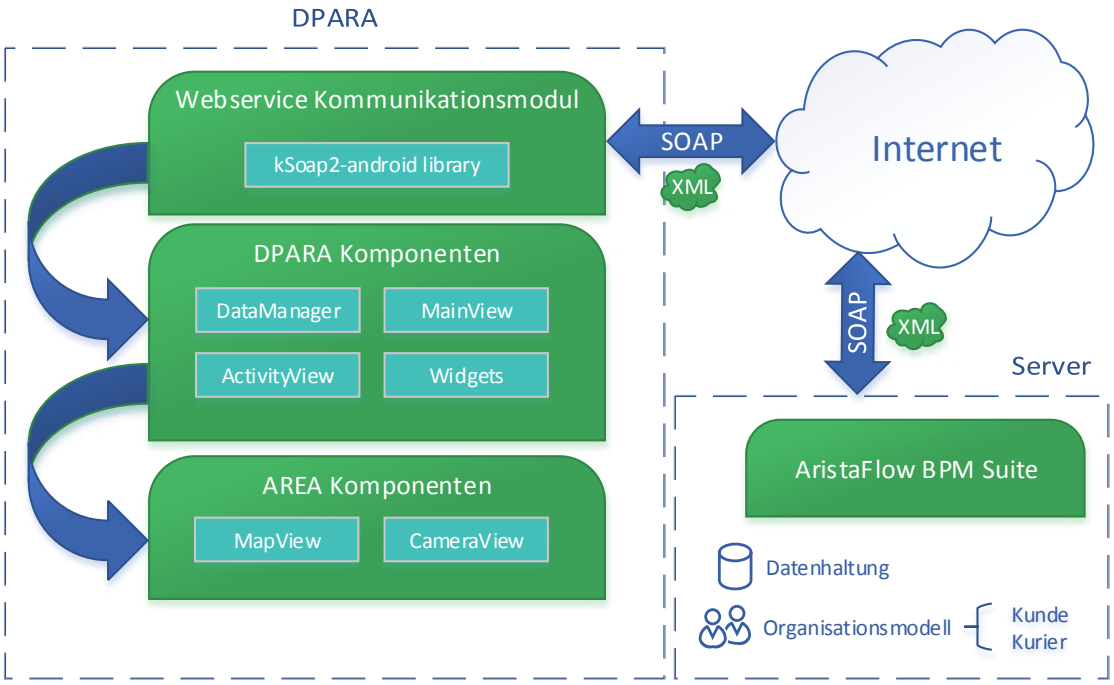
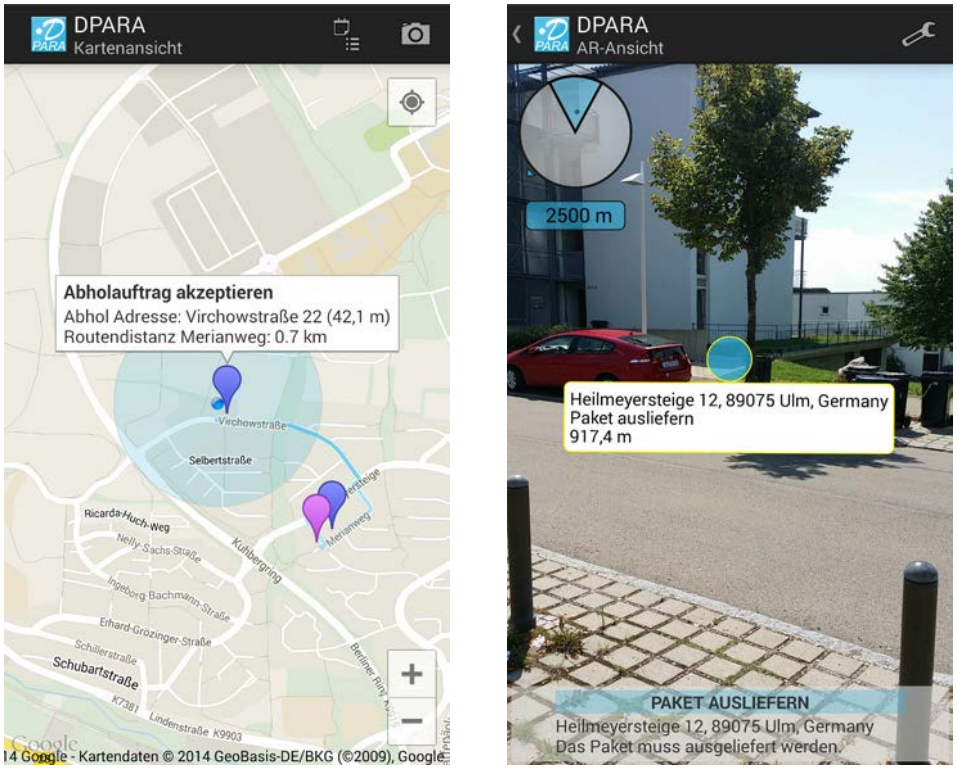


Abbildung 4.1.: Architekturübersicht



(a) Kartenansicht (b) Kameraansicht

Abbildung 4.2.: Karten- und Kameraansicht in DPARA

4.2. Paket- und Klassenstruktur

Abbildung 4.3 zeigt die Paket- und Klassenstruktur von DPARA. Neben AREA besteht DPARA aus sechs wichtigen Paketen. Im Paket `de.ulm.dpara.view` befinden sich alle Klassen, welche zur Anzeige der Listen-, Karten- und Kameraansicht benötigt werden. Das Paket `de.ulm.dpara.model` beinhaltet sämtliche Klassen, die benötigt werden, um die vom Webservice erhaltenen Daten in DPARA geeignet weiterverwenden zu können. Diese Aufgabe übernehmen die Datenmanager im Paket `de.ulm.dpara.data`. Das Paket `de.ulm.dpara.util` beinhaltet Hilfsklassen für Funktionen die öfters verwendet werden (z.B. Methode zur Umwandlung eines Distanzwertes von `DOUBLE` in Meter) und die Kommunikationsklasse `Service`. Des Weiteren können hier über eine

Konfigurationsklasse wichtige Applikationseinstellungen getroffen werden. Das Paket `de.ulm.dpara.adapter` beinhaltet die Adapterklasse für die Listenansicht in DPARA. Die Klassen der AREA-Pakete wurden zugunsten der Übersicht in der Abbildung 4.3 nicht im Detail abgebildet.

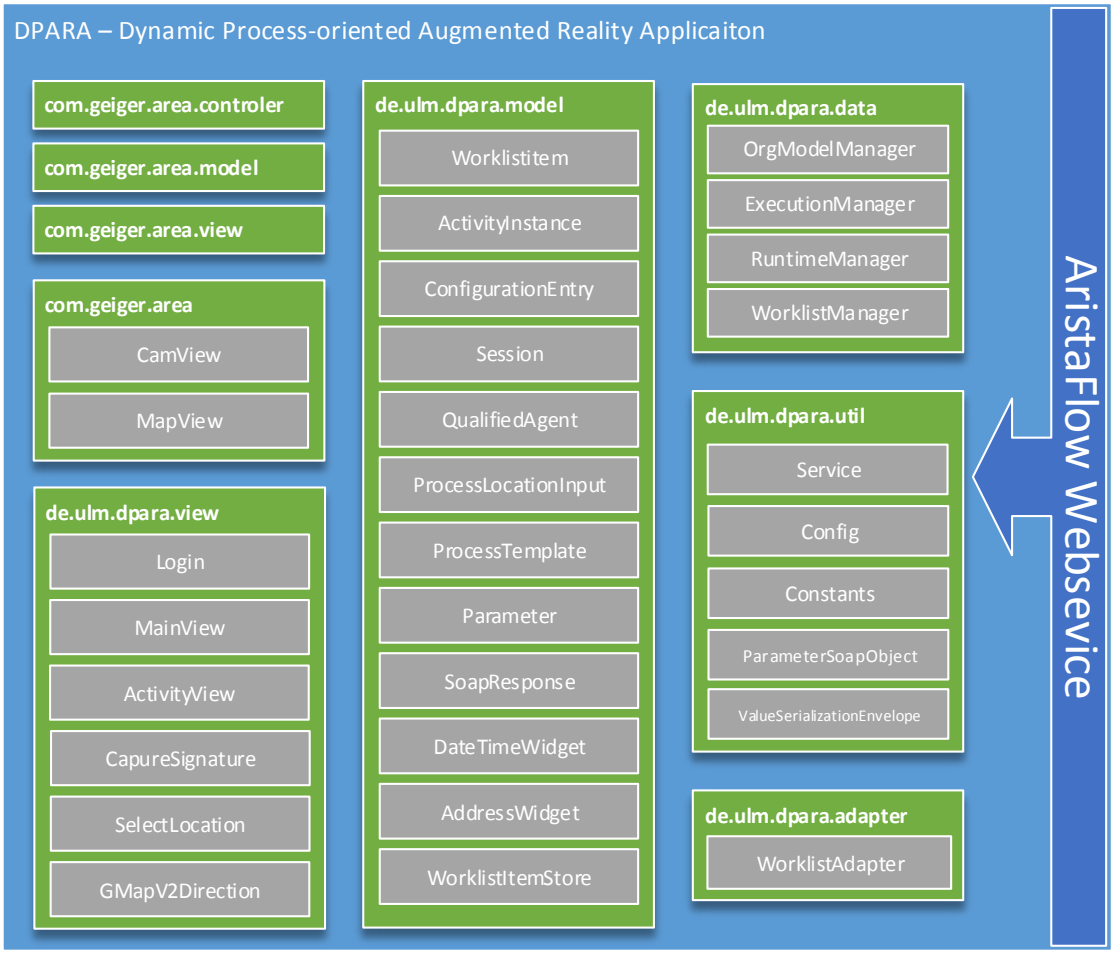


Abbildung 4.3.: Paket- und Klassenstruktur

4.3. Dialogstruktur

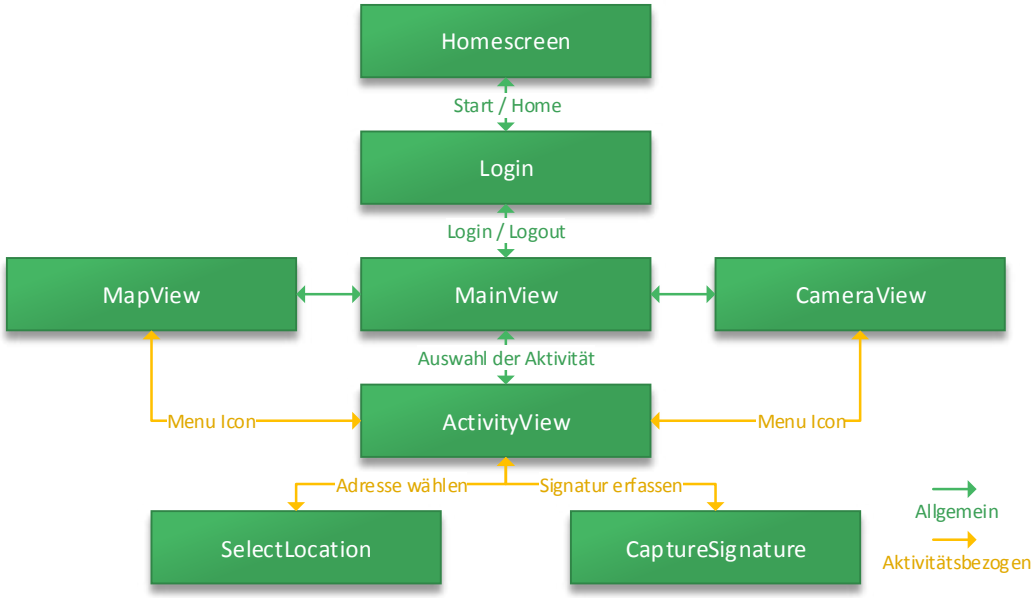


Abbildung 4.4.: Dialogstruktur

Die Dialogstruktur (siehe Abbildung 4.4) gestaltet sich für alle Rollen in DPARA gleich. Zentrum ist die Aktivität `MainView`. In diese gelangt der Nutzer nach Eingabe seiner Login Daten. Über die `Actionbar` kann er nun zwischen der `MainView`, `CameraView` und `MapView` wechseln. Wurde dabei keine spezielle Aktivität vorher ausgewählt, werden hier allgemein alle für den Nutzer ausgewählten Aktivitäten und deren PLIs angezeigt. Wird eine spezielle Aktivität ausgewählt, gelangt der Nutzer in den Bearbeitungsmodus. Auch hier kann er über die `Actionbar` in die `MapView` und `CameraView` gelangen. Die anzuzeigenden Inhalte werden hinsichtlich der ausgewählten Aktivität gefiltert. Die Aktivitäten `SelectLocation` und `CaptureSignature` werden in der `ActivityView` vom Nutzer über die `Buttons` „Adresse wählen“ bzw. „Signature erfassen“ gestartet. Dieser Fall tritt dann auf wenn DPARA Ausgabeparameter als Adresse oder Signatur identifiziert und diese Widgets zur Bearbeitung zuweist.

5

Implementierung

In diesem Kapitel wird auf die wichtigsten Aspekte der Implementierung eingegangen. Als erstes wird die Kommunikationsschnittstelle zu AristaFlow und das Datenmanagement in DPARA betrachtet. Weiterhin werden die wichtigsten Datenstrukturen im Detail erläutert. Am Schluss des Kapitels wird die Implementierung der Präsentationsschicht veranschaulicht. Dies umfasst sämtliche Features der Listen-, Karten- und Kamerasicht. Es folgt in Kapitel 6 die Vorstellung der Anwendung anhand ausgewählter Screenshots.

5.1. Kommunikation

In diesem Abschnitt werden die wesentlichen Aspekte der Kommunikation von DPARA mit dem AristaFlow Webservice erläutert.

5.1.1. Konfiguration

Konfigurationsvariablen, wie die `Server-IP`, `Namespaces`, `Webservice Actions` und `SOAP Fields`, werden durch die `Config` Klasse bezogen. Sollte sich etwas an der Webservice-Definition ändern, kann dies hier zentral angepasst werden. Des Weiteren kann hier der `refresh delay` und `timout` angepasst werden. Diese Variablen geben in Millisekunden an, nach welcher Zeitspanne die Prozessliste automatisch aktualisiert oder eine Fehlermeldung geworfen wird, falls der Webservice nicht erreichbar ist.

5.1.2. Ablauf

Im Paket `de.ulm.dpara.util` befindet sich die Klasse `Service.java`, mit dessen Hilfe Simple Object Access Protocol Nachrichten vom AristaFlow Webservice empfangen und auch zu diesem gesendet werden können. Ein `Complex Request` (dt. Komplexe Anfrage) beschreibt eine Webservice-Operation, der wahlweise mehrere Parameter unterschiedlicher Datentypen angefügt werden können.

```
1 SoapObject r1 = s2.getService(Config.getTermOrgmodelmanager(), Config.
      getActionSecmgrauthenticatename());
```

Listing 5.1: Initialisierung der Webservice Kommunikation

Der Service wird über die Methode `getService()` initialisiert (siehe Listing 5.1). Die notwendigen Parameter werden durch die zentrale Konfigurationsklasse `Config` bereitgestellt (siehe Listing 5.2).

```
1 PropertyInfo pi = new PropertyInfo();
  pi.setNamespace(Config.getNamespaceOrgmodelmanager());
3 pi.setType(PropertyInfo.STRING_CLASS);
  pi.setName(Config.getFieldAgentname());
5 pi.setValue(username);
  r1.addProperty(pi);
```

Listing 5.2: Einem Webservicerequest wird ein String-Parameter hinzugefügt

Die Serviceinitialisierung und Parameterübergabe geschieht in DPARA über die Datenmanager (siehe Kapitel 5.2). Abbildung 5.1 erläutert den Login Vorgang in DPARA mit Hilfe des OrgModel Managers, anhand eines UML-Sequenzdiagramms.

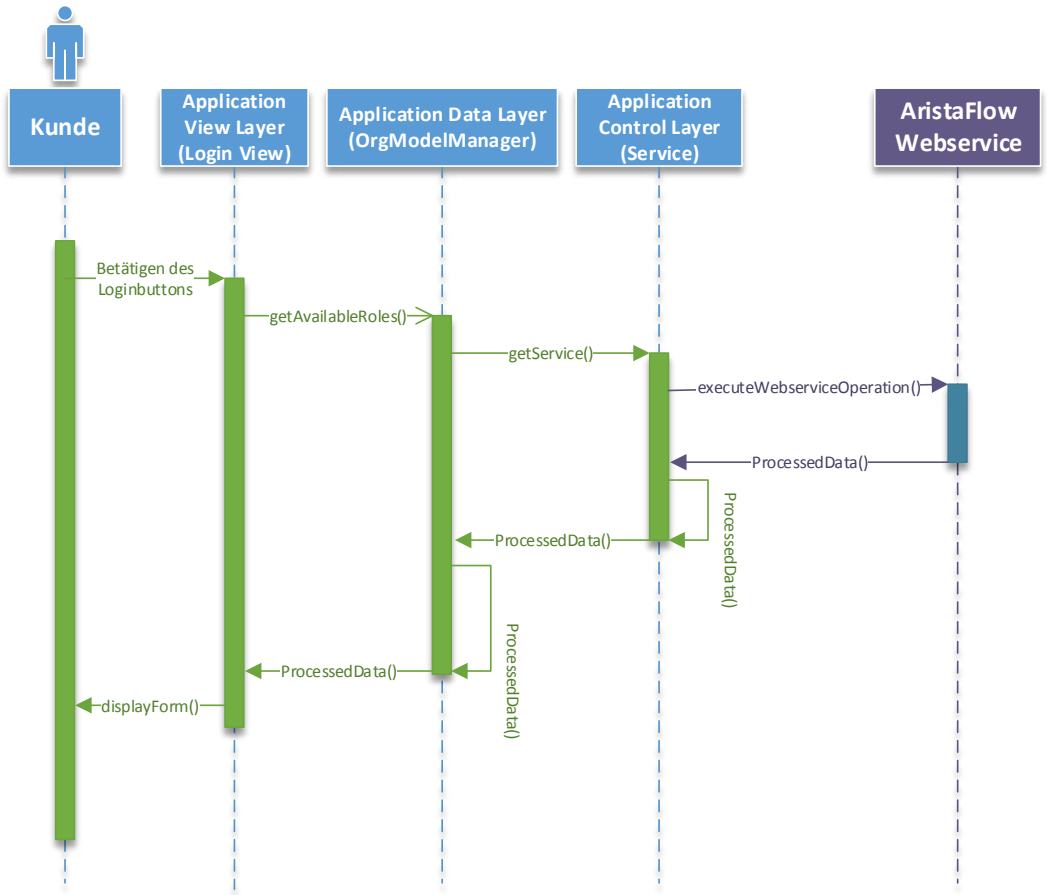


Abbildung 5.1.: Login Vorgang in DPARA

Ein Webservieaufruf kann über den Datenmanager ausgeführt werden. Dies geschieht über die Methode `setComplexRequest()`. In dieser wird der Container für die SOAP Nachricht, der sog. `SoapSerializationEnvelope`, konfiguriert und das im vorherigen Schritt erstellte SOAP-Objekt hinzugefügt. Listing 5.3 zeigt die Konfiguration des `SoapSerializationEnvelope`.

```
SoapSerializationEnvelope soapEnvelope =
    new SoapSerializationEnvelope(SoapEnvelope.VER11);
soapEnvelope.setOutputSoapObject(request);
soapEnvelope.setAddAdornments(false);
soapEnvelope.implicitTypes = true;
soapEnvelope.dotNet = false;
soapEnvelope.env = SoapSerializationEnvelope.ENV;
soapEnvelope.xsd = SoapSerializationEnvelope.XSD;
soapEnvelope.enc = SoapSerializationEnvelope.ENC;
soapEnvelope.xsi = SoapSerializationEnvelope.XSI;
soapEnvelope.encodingStyle = SoapSerializationEnvelope.ENC;
```

Listing 5.3: Konfiguration eines SoapSerializationEnvelope

Anschließend wird über einen HTTP-Aufruf das geschachtelte Objekt an den Webservice versendet. Dem HTTP-Aufruf wird ein `Timeout` (dt. Auszeit) hinzugefügt (siehe Listing 5.4), dieser regelt nach welcher Zeit eine Fehlermeldung geworfen wird, falls der Webservice nicht erreichbar ist.

```
private final HttpTransportSE getHttpTransportSE(String URL) {
HttpTransportSE ht = new HttpTransportSE(Proxy.NO_PROXY,URL,Timeout);
ht.debug = true;
ht.setXmlVersionTag("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
return ht;
}
```

Listing 5.4: Dem HTTP Aufruf wird ein Timeout hinzugefügt

Das Ergebnis des Aufrufs wird in einem `SoapResponse`-Objekt (Kapitel 5.1.4) gespeichert. Sämtliche Webservice-Aufrufe werden nach diesem Schema abgehandelt. Wird ein `SoapFault` zurückgegeben, kann dieser ebenfalls im `SoapResponse`-Objekt gespeichert und bei Bedarf ausgelesen werden. Ein `SoapFault` beinhaltet eine spezifische Webservice-Fehlermeldung.

5.1.3. Session

Für die Kommunikation mit dem AristaFlow Webservice wird zunächst ein Benutzername und Passwort benötigt. Der Webservice antwortet mit den für den Nutzer zulässigen Organisationspositionen und Rollen. Nachdem eine Auswahl getroffen wurde, werden vom Webservice weitere Benutzerkontendetails und ein `Token` übermittelt. Dieser `Token` und eine von DPARA erstellte `SessionID` werden bei allen weiteren Anfragen benötigt. Die Anmeldung ist nun vollständig und der Webservice übermittelt die `WorklistID` des Nutzers. Über diese kann nun die Liste mit ausführbaren Prozessen (bestehend aus `WorklistItems`) angefragt werden. In DPARA werden `Token`, `SessionID` und `WorklistID` in einem `Session`-Objekt abgelegt. Dieses wird von den Datenmanagern, wie z.B. dem `OrgModelManager`, für die weitere Kommunikation genutzt. Abbildung 5.2 veranschaulicht diesen Vorgang in einem UML-Sequenzdiagramm.

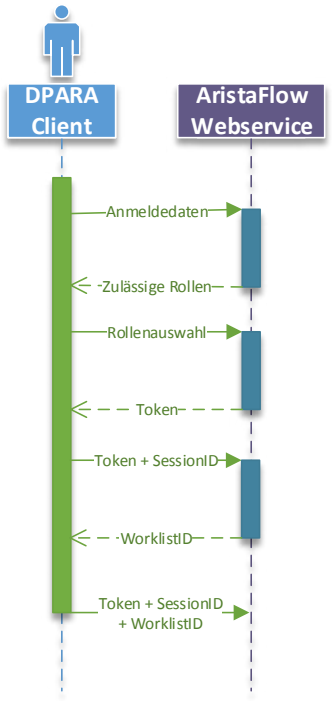


Abbildung 5.2.: Anmeldung am AristaFlow Webservice

5.1.4. SoapResponse

Um das Ergebnis einer Webservice-Anfrage geeignet weiterverarbeiten zu können, existiert die Klasse `SoapResponse`. Mit Hilfe dieser können nicht nur `SoapObjects` gespeichert und weitergereicht werden, sondern auch `SoapFaults` und `SoapPrimitives`. Ein `SoapFault` ist eine Fehlermeldung vom Webservice. Aus diesem Objekt können Informationen wie `FaultCode` und `FaultActor` gelesen werden. Auf diese Weise kann dem Nutzer beim Auftreten eines Fehlers Feedback gegeben werden. Die Managerklassen von DPARA erhalten bei Webservice-Anfragen diesen Objekttyp als Ergebnis und haben für dessen Inhalte die entsprechenden Abhandlungsroutinen, um neue Informationen abzulegen oder eine Fehlermeldungen zu setzen.

5.2. Datenmanagement

Über die Datenmanagerklassen kann DPARA auf Inhalte der Webservice-Antworten von AristaFlow zugreifen. Jede Klasse verwaltet die für ihren Service spezifizierten Methoden, wobei DPARA sich hier an der Webservice-Definition von AristaFlow orientiert. Die Managerklassen des `OrgModelManager` und `ExecutionManager` benötigen für viele Methoden einen `Token` und eine `Session`. Diese werden in Form eines `Session`-Objekt hinterlegt. Antwortet der Webservice mit einer Fehlermeldung wird diese in dem `SoapResponse` Objekt als `SoapFault` gespeichert. Die Service initiierende Aktivität wird über den Fehler benachrichtigt und kann nun bei Bedarf die spezifische Fehlermeldung dem Nutzer anzeigen. Dies geschieht in DPARA über das `Toast` Widget.

5.2.1. OrgModelManager

Der `OrgModelManager` wir für den Login Prozess initiiert. Er benötigt einen Benutzernamen und das zugehörige Passwort. Mit diesen Informationen kann nun über die Methode `getAvailableRoles()` die für den Nutzer hinterlegten Rollen über den Webservice angefragt werden. Jener antwortet mit einer Liste von Agenten, welche in DPARA als `QualifiedAgents` weiter behandelt werden. Diese Liste von `QualifiedAgents` wird

nun über einen `AlertDialog` dem Nutzer präsentiert. Dieser entschiedet sich für eine Rolle und kann über die Methode `loginWithRole()` den Login Prozess vervollständigen. Bei erfolgreichem Login, übermittelt der Server einen `Token`, welcher für die weitere Nachrichtenkommunikation von DPARA in einem `Session`-Objekt übermittelt wird. Die gewählte Rolle wird ebenfalls unter dem Variablennamen `activeAgent` der nächsten Aktivität weitergereicht. Diesem sind Informationen, wie die Mailadresse, Vor- und Nachname, hinterlegt.

5.2.2. WorklistManager

Nachdem der Nutzer sich in DPARA mit seinen Benutzerdaten angemeldet hat, wird hier über die Methode `logon()` und der Übergabe des `Tokens` die `WorklistID` erhalten. Über diese kann nun die `Worklist`, eine Liste von ausführbaren Prozessaktivitäten, angefragt werden. Wird die Applikation initial gestartet (bzw. befindet sich nicht im Speicher), ruft DPARA zunächst die **vollständige** Liste der für den Nutzer ausführbaren Prozessaktivitäten ab. Sobald die Liste in DPARA existiert wird diese im `WorklistItemStore` gehalten und darauf folgende Anfragen mit einer Revisionsnummer versehen. Über diese werden nun Prozessaktivitäten, welche seit der aktuellen Revisionsnummer unverändert sind, herausgefiltert und nicht mitgesendet. Dies reduziert die zu übertragende Datenmenge. Der Webservice teilt in seiner Antwort zudem mit, ob ein Prozess geändert, gelöscht oder neu hinzugefügt wurde. Diese Information wird genutzt, um in DPARA `User Notifications` (dt. Nutzerhinweise) anzuzeigen. Über die Methode `updMgrGetAttachedDataContext` kann DPARA die öffentlichen Variablen eines Prozessen anfragen. Auf diese Weise können ausgewählte prozessinterne Informationen in der `ListView` angezeigt werden, ohne tiefer in den Prozess einsteigen zu müssen.

5.2.3. ExecutionManager

Über den `ExecutionManager` können Templates abgefragt, instanziiert und gestartet werden. Die Methode `getInstantiableTemplates()` frägt alle für den Nutzer

ausführbaren Templates vom Server an. Dieser antwortet mit einem `UUID` (Universally Unique Identifier), welcher eine `TemplateID` repräsentiert. Über diesen Identifikator und die Methode `getTemplateReference()` können nun weitere Informationen zum Template erhalten werden. Zu diesen Informationen zählen der Name, eine Beschreibung und prozessspezifische Details, die z.B. darüber Auskunft geben, ob der Prozess einen Subprozess enthält, oder abgeändert werden kann. Weiterhin kann ein Prozess über die `TemplateID` gestartet oder beendet werden. Wird ein Prozess vorzeitig vom Nutzer beendet, kann dem Server ein Abbruchgrund übermittelt werden.

5.2.4. RuntimeManager

Der `RuntimeManager` ist dafür zuständig, dass Aktivitäten gestartet, pausiert, wiederaufgenommen, zurückgesetzt und beendet werden können. Dafür benötigt er zusätzlich neben dem `Session`-Objekt, das für die jeweilige Aktivität verantwortliche `WorklistItem`. Aus diesem werden wichtige Informationen wie z.B. der `EbpType` gelesen und der Anfrage hinzugefügt. Nachdem eine Aktivität gestartet wurde, erhält DPARA alle notwendigen Informationen, welche zur Anzeige bzw. Ausführung notwendig sind. Dazu gehören in erster Linie die Ein- und Ausgabeparameter und die jeweils zugehörigen Konfigurationseinträge. Damit diese in geeigneter Form weiterverarbeitet werden können, werden in DPARA entsprechende Datenstrukturen zur Verfügung gestellt (siehe Kapitel 5.3).

5.3. Datenstrukturen

Die vom Webservice empfangenen Daten müssen in DPARA in geeigneter Form weiterverarbeitet werden können. In diesem Kapitel werden die wichtigsten im Detail erläutert.

5.3.1. WorklistItem

Ein `WorklistItem` beinhaltet die wichtigsten Informationen einer ausführbaren Prozessaktivität. Neben einer `ID`, dem Namen und einer Beschreibung, können hier Ein-

stellungen der Aktivität abgefragt werden. Über die boolschen Einträge `closable` und `suspensible` wird geregelt, ob die Aktivität geschlossen oder pausiert werden darf. In DPARA wird dementsprechend das Funktionsspektrum angepasst und die damit zusammenhängenden GUI-Elemente generiert und angezeigt.

5.3.2. Parameter

Ein `WorklistItem` besitzt eine Liste von Ein- und Ausgabeparameter. Die wichtigsten Eigenschaften eines Parameters werden in diesem Abschnitt behandelt. Jeder dieser Parameter hat einen Namen und eine Beschreibung. Er wird über eine `identifierID` eindeutig gekennzeichnet und besitzt einen zugewiesenen Wert. Diese Werte werden durch den AristaFlow Webservice geliefert und dann in DPARA so aufbereitet, dass sie in jeder Aktivität genutzt werden können. Im AristaFlow Process Template Editor kann zusätzlich bestimmt werden, ob ein Parameter z.B. optional ist. Solche Einstellungen können auch in DPARA am Objekt abgefragt werden. Einem Parameter kann eine Liste von `Nodes` zugewiesen werden. Diese Liste beschreibt, an welchen `Nodes` im Prozess der Parameter und seine zugehörigen PLIs sichtbar sind. Zusätzlich besitzt er eine Liste von `ConfigurationEntrys` (dt. Konfigurationseinträge). Über diese können dem Parameter spezielle Eigenschaften zugewiesen werden. Diese werden näher im Kapitel 5.3.3 beschrieben.

5.3.3. ConfigurationEntry

Konfigurationseinträge werden benötigt, um einem Ein- und Ausgabeparameter weitere Eigenschaften wie z.B. ein `Identifier` (dt. Identifikator), oder ein `predefinedValue` (dt. vordefinierter Wert) zuordnen zu können. Über den `Identifier` kann geregelt werden, ob es sich bei dem Parameter um eine Adresse oder z.B. um eine Signatur bzw. Unterschrift handelt. DPARA kann dann über diese Information entscheiden, welches Widget benötigt wird, um den Parameter anzuzeigen oder bearbeiten zu können.

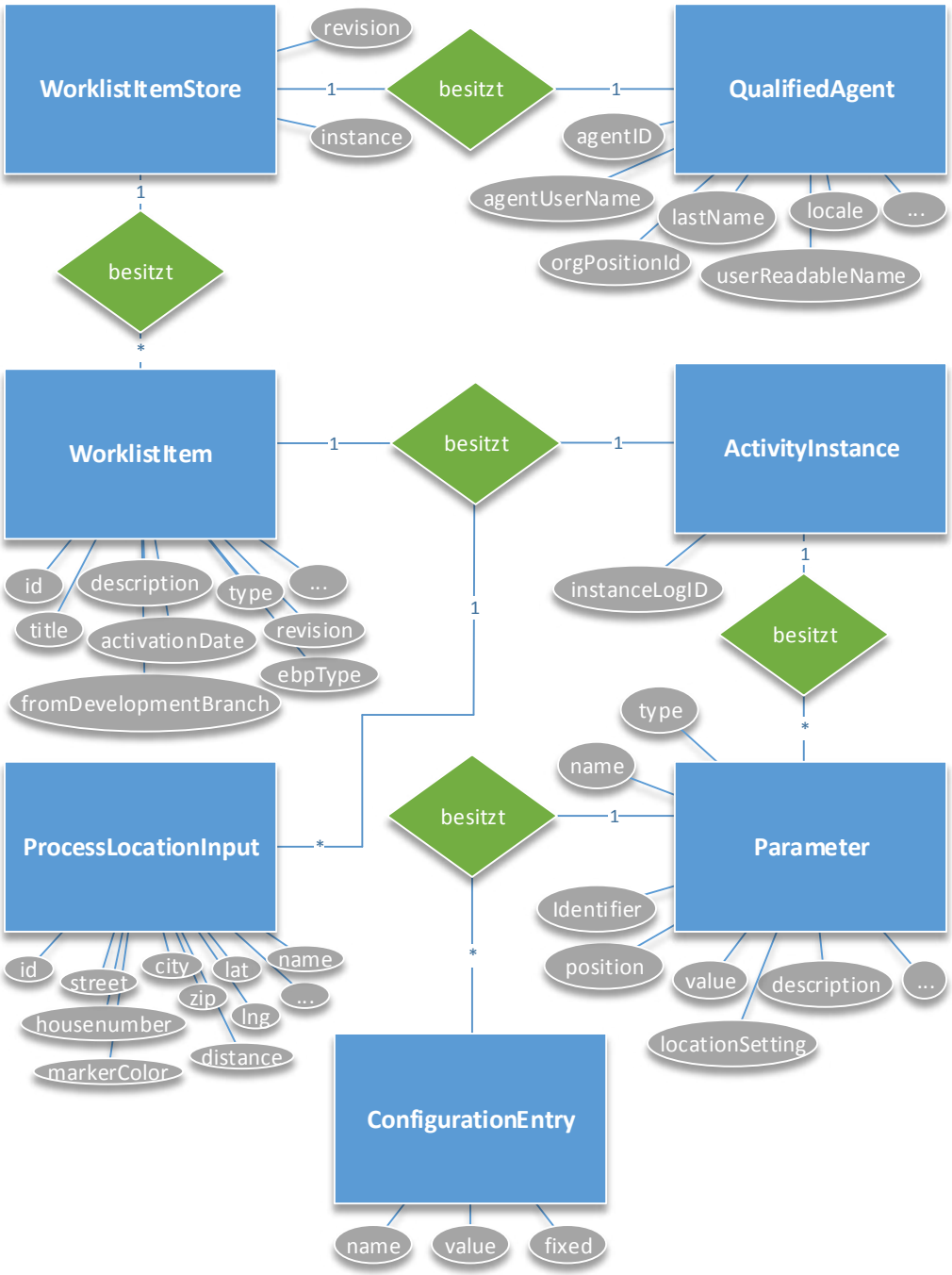


Abbildung 5.3.: WorklistItemStore ER-Diagramm

5.3.4. WorklistItemStore

Im `WorklistitemStore` werden alle für den eingeloggten Nutzer ausführbaren Prozessaktivitäten mit der zugehörigen Revisionsnummer gespeichert. Der Nutzer ist in Form eines `QualifiedAgents` hinterlegt. Der Store bleibt solange die Applikation aktiv im Speicher gehalten wird bestehen. Die Aktivitäten werden nicht lokal auf dem Gerät in Form einer Datenbank abgelegt. Beim Nutzerwechsel wir automatisch die Arbeitsliste ausgetauscht. Abbildung 5.3 zeigt ein komplettes ER-Diagramm des `WorklistitemStores`.

5.3.5. ProcessLocationInput

In DPARA repräsentiert ein `ProcessLocationInput` eine Positionsangabe in einer Aktivität. Dazu muss im Prozess Template ein Parameter mit dem `Identifier Address` definiert sein. Dieser wird vom Nutzer in der `ActivityView` über das `SelectLocation` Widget (siehe Kapitel 5.4.1.2) bearbeitet. Über dieses kann er die benötigten Adressdaten händisch eingeben oder diese über eine Kartenansicht bestimmen. Zur vollständigen Adresserfassung wird eine asynchrone Anfrage [6] an die Google Geocoding API [8] mit den vom Nutzer erfassten Daten gestartet. Das Resultat ist ein `JSON`-Dokument mit detaillierten Informationen, wie Höhe, genauen Geokoordinaten und einer formatierten Adressausgabe.

```
{
"geometry" : {
"location" : {
"lat" : 48.423353,
"lng" : 9.953213999999999
}
},
"icon" : "http://maps.gstatic.com/mapfiles/place_api/icons/university71.png",
"id" : "d64a243fe115d818bdbab2a34fb8905f721f7b01",
"name" : "University Ulm",
"photos" : [
```

```
{
"height" : 480,
"html_attributions" : [ "Von einem Google-Nutzer" ],
"photo_reference" : "CnRoAAAAO7hRmOKNHm8AKZMNwHuH_CZ4PTT2 ...",
"width" : 640
}
],
"place_id" : "ChIJyZn2er9lmUcRTj9Y8ZWk2qY",
"rating" : 4.5,
"reference" : "CoQBcwAAAEpd22j6trWAVP5fCoEMY4mlw07t03RWO ...",
"scope" : "GOOGLE",
"types" : [ "university", "establishment" ],
"vicinity" : "Helmholtzstrasse 18, Ulm"
}
```

Listing 5.5: JSON-Ausgabeformate der Geocodierungsantwort

Bei mehreren Parametern werden die PLIs untereinander in der vom Nutzer angegebenen Reihenfolge verknüpft. Auf diese Weise kann DPARA über eine `Direction And Places Library` [1] eine Route berechnen und bei Bedarf anzeigen . In der Kartenansicht kann der Nutzer über ein Kontextmenü zwischen den verknüpften Adressen wechseln (siehe Abbildung 5.4). Des Weiteren besitzen Positionsangaben in der Kartenansicht eine spezifische Farbe für ihre Marker, welche innerhalb der Gruppe gleich ist und für den gesamten Ablauf des Prozesses gespeichert wird. Der Nutzer kann sich dadurch besser orientieren, was gerade bei einer größeren Anzahl von PLIs nützlich ist.

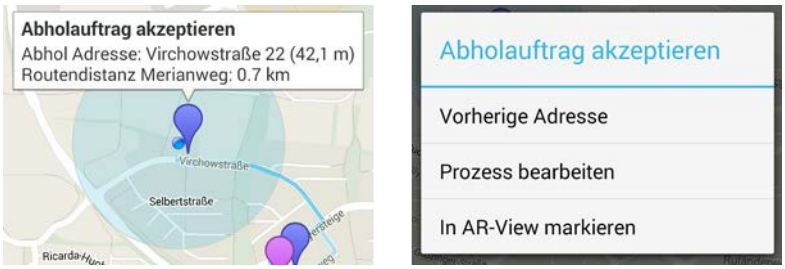


Abbildung 5.4.: Markerdarstellung und Kontextmenü der Kartenansicht

5.4. Präsentationsschicht

Dieses Kapitel beschreibt die Präsentationsschicht der Applikation und deren Funktionen. Insgesamt besteht DPARA aus sieben `Activities` und 20 XML-Dokumenten. Tabelle 5.1 zeigt die wichtigsten in der Übersicht und erläutert ihre Funktionalität und Aufgabe.

Name	**Funktionsbeschreibung**	**Layout**
Login	Systemanmeldung über Benutzername und Passwort.	Activity
MainView	Anzeige der ausführbaren Aktivitäten eines Prozesses in Listenform.	ListActivity
ActivityView	Dient zur dynamischen Detailansicht von Aktivitäten eines gestarteten Prozesses und deren Ein- und Ausgabeparameter.	Activity
AREAMainActivity	Anzeige der PLIs in der Kameraansicht und Anzeige der damit verbundenen Prozessdetails.	Activity
MapView	Kartenansicht ausgewählter Aktivitäten eines Prozesses.	Activity
SelectLocation	Auswahl einer Adresse auf einer Karte. Wird von DPARA als Widget für die Eingabe eines PLIs in der `ActivityView` verwendet.	Activity
CaptureSignature	Widget zur Erfassung einer Unterschrift.	Activity

Tabelle 5.1.: Activities in DPARA

5.4.1. Aktivitätenansicht

Nach der Auswahl und Bestätigung eines Listenelements in der `ListView` gelangt der Nutzer in die `ActivityView`. In dieser werden zu Beginn alle UI-Elemente, welche zur vollständigen Anzeige der Aktivität benötigt werden, dynamisch anhand der Ein- und Ausgabeparameter erstellt und konfiguriert. Besitzt die Aktivität als Eingabeparameter

einen PLI, wird in der ersten Zeile die Distanz zu diesem angezeigt. In der Methode `createUI()` prüft DPARA jeweils Ein- und Ausgabeparameter auf ihren Typ und erstellt dementsprechend geeignete UI-Widgets. Der Datentyp eines Parameters wird über den AristaFlow Process Template Editor eingestellt. Tabelle 5.2 zeigt alle Parametertypen, die in DPARA behandelt werden können, in der Übersicht.

Typ	**Beschreibung**	**UI-Element**
STRING	Für die Erfassung eines `STRING` Parameters generiert DPARA ein `EditText` Widget, mit der Überschrift und Beschreibung des Parameters.	EditText
DATE	Für die Erfassung eines Parameters vom Typ `DATE`, werden in DPARA Zeit und Datum in einem eigens konfigurierten `DateTime` Widget erfasst.	DateTime Widget
BOOLEAN	Parameter vom Typ `BOOLEAN` werden über ein `CheckBox` Widget erfasst. Der Name und die Beschreibung des Parameters werden als Titel und Hinweis gesetzt.	CheckBox
ADDRESS	Wird in DPARA ein Parameter vom Typ `STRING` mit dem Identifier `Address` gekennzeichnet, behandelt diesen DPARA mit dem `SelectLocationWidget`.	Select Location Widget
SIGNATURE	In DPARA können Unterschriften per Widget erfasst werden. Parameter müssen dazu mit dem Identifier `Signature` versehen sein.	Capture Signature Widget

Tabelle 5.2.: Erfassung und Darstellung von Ein- und Ausgabeparameter

Beim Abschluss einer Aktivität wird über die Methode `checkLocationSetting()` geprüft, ob die Aktivität ein PLI besitzt und ruft dessen Konfigurationsparameter ab. In DPARA kann angegeben werden, dass ein PLI zwingend für eine Aktivität ist. Dies bedeutet, dass der Nutzer die Aktivität nur dann abschließen kann, wenn er sich in unmittelbarer Nähe zu diesem Punkt befindet. In der Applikation wird dieser Abstand

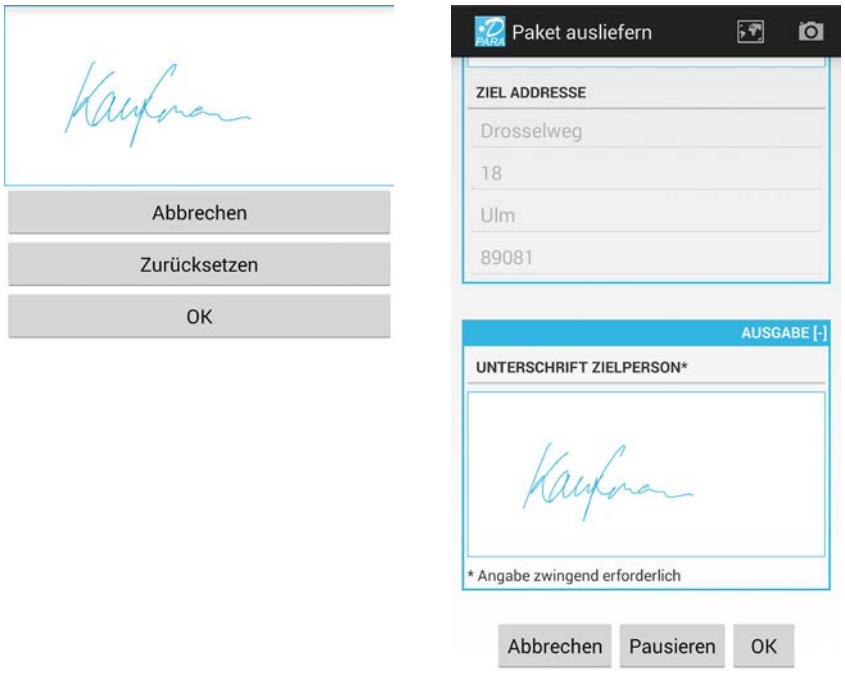
durch den persönlichen Radius angegeben. Dieser Parameter kann in der `Config` eingestellt werden. Sämtliche Parameter, welche als Eingabe zwingend sind, werden hinter dem Titel mit einem Stern markiert. Vergisst der Nutzer diese Eingabe zu tätigen, wird er vom System über einen `Toast` darauf hingewiesen. Sind alle benötigten Eingaben vorhanden, kann der Nutzer je nach Einstellung die Aktivität abschließen, pausieren oder zurücksetzen. DPARA liest die Einstellungen einer Aktivität automatisch und stellt entsprechend `Buttons` für die angegebene Funktion bereit. Wird eine Aktivität pausiert, werden die bisher eingegebenen Informationen zwischengespeichert und beim nächsten Start wieder angezeigt. Des Weiteren wird die Aktivität in der Listenansicht grau hinterlegt, um den Zustand visuell dem Nutzer zu vermitteln. Wird die Aktivität zurückgesetzt, werden alle bisherigen Eingaben gelöscht und müssen vom Nutzer neu eingegeben werden.
Ein- und Ausgabeparameter können zusätzlich in der `ActivityView` über eine Variable in der Darstellungsreihenfolge sortiert werden. Diese wird in AristaFlow über die `position`-Variable angegeben. Eingabeparameter werden vor den Ausgabeparametern angezeigt. Beide Bereiche können bei Bedarf vom Nutzer minimiert werden, um eine bessere Übersicht zu erhalten. In Abbildung 5.5 wurde ein Parameter vom Typ `DATE` für die Ausgabe erkannt. Listing A.1 (Anhang) zeigt wie DPARA in der `ActivityView` diesen behandelt.

Abbildung 5.5.: Eingabe eines Parameters vom Typ `DATE`

5.4.1.1. Signature-Widget

Um einen Abhol- bzw. Annahmevorgang sicher bestätigten und später nachweisen zu können, wird in DPARA die Unterschrift des Kunden bzw. der Zielperson erfasst. Dies ist bei großen Paketdiensten, wie DHL und UPS, die gängige Methode. Bei einem Smartphone besteht die Möglichkeit auf dem Touchscreen eine Unterschrift zu zeichnen. Viele Geräte besitzen einen sogenannten Stylus, welcher Schrifteingaben wesentlich erleichtert. In DPARA wurde dieses wichtige Feature im `SignatureWidget` umgesetzt [17]. Besitzt die Aktivität als Ausgabe einen Parameter mit dem Datentyp `USERDEFINED` und dem Identifikator `Signature`, behandelt DPARA diesen mit dem Unterschriften-Widget (siehe Abbildung 5.6). Es wird ein `Button` mit der Beschriftung „Unterschrift erfassen“ eingeblendet. Im Widget selbst kann nun auf einer begrenzten weißen Fläche die Unterschrift vom Kunden erfasst werden. Die Eingabe kann beliebig oft wiederholt werden. Wird die Eingabe final bestätigt, generiert DPARA ein `Bitmap` und gibt dieses als `ByteArray` der Aktivität zurück. Im Nächsten Schritt wird der `ByteArray` Base64 [7] kodiert und kann nun in Form eines `STRING`-Datentyps dem Webservice übermittelt

werden. Bei Aktivitäten, die Signaturen als Eingabeparameter enthalten, kann DPARA diese dekodieren und über ein `ImageView`-Widget anzeigen.



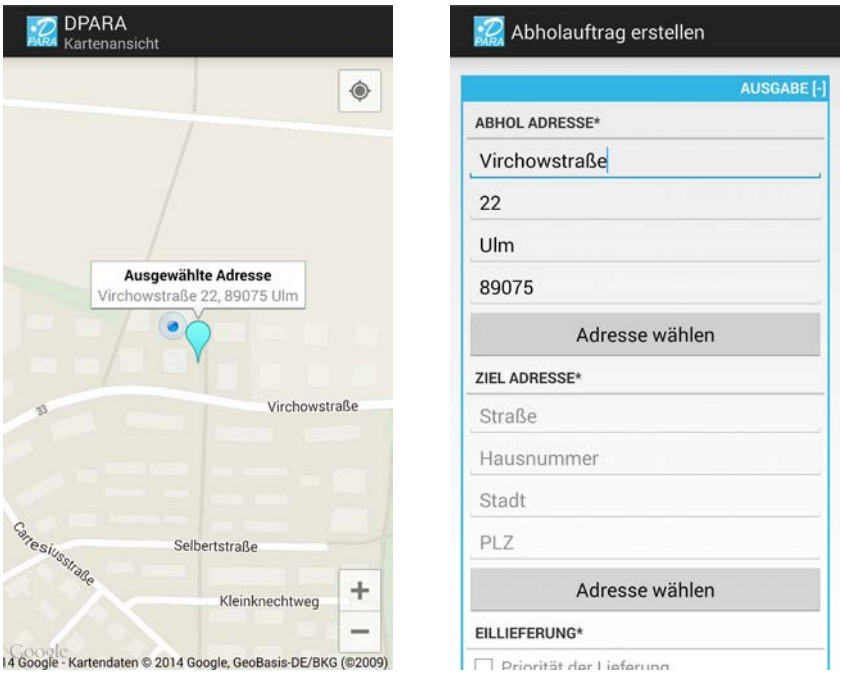
(a) Erfassung einer Unterschrift (b) Erfasste Kundenunterschrift

Abbildung 5.6.: Erfassung und Anzeigen einer Unterschrift

5.4.1.2. SelectLocation-Widget

Sind in einer Aktivität PLIs als Eingabeparameter vorhanden, kann der Nutzer zu jedem einzeln die Adressdaten eingeben, oder er nutzt das von DPARA automatisch angebotene `SelectLocation`-Widget. Unter dem letzten `EditText`-Feld der Adresseingabe wird ein `Button` positioniert, welcher das Widget startet. Der Nutzer gelangt in eine Kartenansicht und bekommt den Hinweis, dass er durch „Klicken“ auf die Karte eine Position bzw. Adresse markieren kann. Nachdem ein Punkt ausgewählt wurde, wird dieser mit einem `Marker` auf der Karte, unter Angabe der vollen Adresse, angezeigt. Diese wird über einen `Geocoder` angefragt. Wird diese Auswahl vom Nutzer durch erneutes

„Klicken“ bestätigt, springt DPARA automatisch zurück in die `ActivityView` und füllt die `EditText` - Felder für die Eingabe des PLIs, mit den im vorherigen Schritt erfassten Daten aus (siehe Abbildung 5.7). Die Aktivität kann nun normal weiter bearbeitet werden. Dieser Vorgang kann beliebig oft wiederholt werden.



(a) Adresseingabe über die Kartenansicht

(b) Manuelle Adresseingabe

Abbildung 5.7.: Auswahl einer Adresse über das SelectLocation Widget

5.4.2. Listenansicht

Nach erfolgreichem Login befindet der Nutzer sich in der Listenansicht. Hier werden alle ausführbaren Aktivitäten eines Prozesses untereinander aufgelistet. In einem Listenelement werden zu einer Aktivität das Erstellungsdatum, die Erstellungszeit, der Titel, eine Beschreibung und die Distanz zum aktuellen PLI angezeigt. Wird das Listenelement direkt ausgewählt, gelangt der Nutzer in die `ActivityView` und kann dort die Aktivität im Detail bearbeiten. Zusätzlich beinhaltet das Listenelement auf der rechten Seite ein

Icon, welches das Kontextmenü der Aktivität öffnet (siehe Abbildung 5.8). Über dieses Menü kann der Nutzer die Priorität der Aktivität anpassen oder diese mit einer optionalen Angabe beenden. Der Prozess ist dann vom Nutzer abgebrochen worden. Über die Menütaste kann der automatische Update-Modus gestartet werden. Dieser sorgt dafür, dass in allen Ansichten die Liste ausführbarer Aktivitäten aktuell gehalten wird. Die Zeitspanne der Aktualisierung kann in der `Config` eingestellt werden. Der Nutzer kann weiterhin über dieses Menü Prozesse filtern, neue Prozesse über die Templateauswahl starten, oder sich von DPARA abmelden.

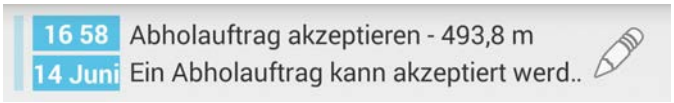


Abbildung 5.8.: Listenelement

5.4.2.1. Subprozesse

In der Arbeit wurde als Anforderung definiert, Subprozesse in DPARA ausführen zu können. In Aristaflow können ganze Prozessabläufe als sogenannter Subprozess ausgelagert werden. Subprozesse haben den `EbpType` LWP und werden in DPARA vom `ExecutionManager` mit der Methode `actStartStartActivity()` behandelt. Abbildung 5.9 zeigt den aktivierten Subprozess „Auftrag überprüfen“ in der `ListView`.

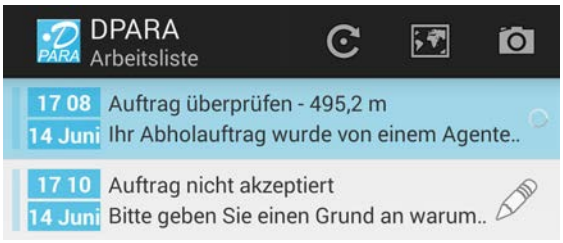


Abbildung 5.9.: Darstellung eines Subprozesses

Dieser ist während der Ausführung blau hinterlegt und nicht anwählbar. Um ihn weiter bearbeiten zu können, werden die Ergebnisse des Subprozesses „Auftrag nicht akzeptiert“ benötigt. In diesem Fall sind das der Ablehnungsgrund und ein Wunschabholdatum vom Kunden. Erst nachdem diese Informationen vorliegen kann der Kurier den Auftrag weiter bearbeiten. Ein Ladezeichen rechts neben der Beschreibung suggeriert dem Nutzer, dass der Prozess sich momentan in Warteposition befindet. Wurde der Subprozess vom Nutzer erfolgreich bearbeitet, aktualisiert DPARA automatisch die Arbeitsliste und zeigt dem Nutzer nun die anstehende Aktivität des vorherigen Prozesses an.

```
1 if (w.getEbpType().equals("LWP")) {
      if (w.getState().equals("AVAILABLE")) {
3         worklistItem = w;
          if (worklistItem != null)
5             new BackgroundAsyncTask(5).execute();
      }
7 }
```

Listing 5.6: Behandlung eines Subprozesses in DPARA

Listing 5.6 zeigt wie DPARA in der `ListView` den `EbpType` einer Aktivität überprüft und dementsprechend die Abhandlungsroutine auswählt. Nachdem ein Subprozess über die AristaFlow Engine erfolgreich ausgeführt wurde, muss DPARA über den `WorklistManager` und die Methode `getFromDevelopmentBranch()` prüfen, ob sich alte Aktivitäten des übergeordneten Prozesses in der Arbeitsliste befinden und löscht diese dann aus dem `WorklistItemStore`.

5.4.2.2. Automatic Update / Broadcast

In DPARA wurde ein Funktion implementiert, welche im Hintergrund automatisch die Prozessliste aktualisiert. Das Intervall kann in der `Config` eingestellt werden. Damit nicht bei jedem Update die komplette Prozessliste angefragt wird, wurde eine revisionsbasierende Aktualisierungsanfrage gewählt (siehe Kapitel 5.2.2). Damit der Nutzer

auch in der Karten- und Kamerasicht davon profitiert wirft DPARA einen `Broadcast`, welcher von den genannten Klassen über einen `Broadcast Listener` (siehe Listing 5.7) empfangen wird. Auf diese Weise werden alle Ansichten aktuell gehalten. Die automatische Aktualisierung wird in der Listenansicht über das Optionsmenü aktiviert. Die Einstellungen werden über die `SharedPreferences` (siehe Kapitel 5.4.2.5) gespeichert und beim nächsten Programmstart wiederhergestellt.

```
receiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
    String rev = "";
    if (intent.getExtras().containsKey("revision"))
        rev = intent.getExtras().get("revision").toString();
    if (!revision.equalsIgnoreCase(rev)) {
        map.clear();
        updateMap();
        Toast.makeText(context, getString(R.string.updateMap),
                Toast.LENGTH_SHORT).show();
        revision = rev;
    }
}
};
```

Listing 5.7: Broadcast Receiver in der MapView

5.4.2.3. Filtereinstellungen

Die in der `ListView` angezeigten Prozessaktivitäten können nach Aktivierungsdatum, Priorität und Distanz, jeweils auf- und absteigend sortiert werden. Bei einer größeren Anzahl an Aktivitäten kann der Nutzer sich mittels dieser Funktion besser orientieren. Die gewählte Einstellung wird in den `SharedPreferences` gespeichert und ist somit beim nächsten Start der Applikation wieder aktiv. Für die Filterfunktion wurde eine Comparator Klasse angelegt, welche um beliebige Eigenschaften eines `WorklistItems` erweitert

werden kann. Somit sind weitere Filterfunktionen leicht umzusetzen. Listing 5.8 zeigt ein Codebeispiel für eine Sortierung nach dem Aktivierungsdatum.

```
1 public static Comparator<WorklistItem> ACTIVATIONDATE_ASC = new Comparator<
      WorklistItem>() {
  @Override
3 public int compare(WorklistItem o1, WorklistItem o2) {
    return o1.activationDate.compareTo(o2.activationDate);
5 }
  };
```

Listing 5.8: Filterung nach dem Aktivierungsdatum

5.4.2.4. Prioritätseinstellungen

Die Prioritätseinstellungen in DPARA werden über das Kontextmenü (siehe Abbildung 5.11) einer Aktivität erreicht. Hier stehen über ein `Spinner`-Widget alle acht Prioritätsstufen zur Auswahl. Die Prioritätswerte wurden von AristaFlow übernommen. Der Nutzer kann die individuelle Priorität einer Aktivität ändern. Visuell wird die Priorität in DPARA über einen Farbbalken im Listenelement dargestellt. Abbildung 5.10 zeigt die verschiedenen Abstufungen des Farbbalkens, welche in direkter Abhängigkeit zum Prioritätswert stehen.

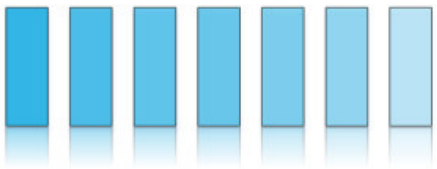


Abbildung 5.10.: Darstellung der Priorität einer Aktivität

Mit steigender Priorität erhöht sich der Farbkontrast und damit die visuelle Präsenz. Das Listenelement wird hervorgehoben. Da die Prioritätseinstellung über ein `AlertDialog` im `ArrayAdapter` der Listenansicht geschieht, muss diese nun auf geeignete Weise

weitergereicht werden. Zu diesem Zweck wurde ein `MainView Interface` erstellt, welches über die Methode `updateProcessSettings()` (siehe Listing 5.9) die geänderten Einstellungen abfragen kann. Die Datenmanager können nun über ihre Methoden die Einstellungen an den AristaFlow Webservice weitergeben.

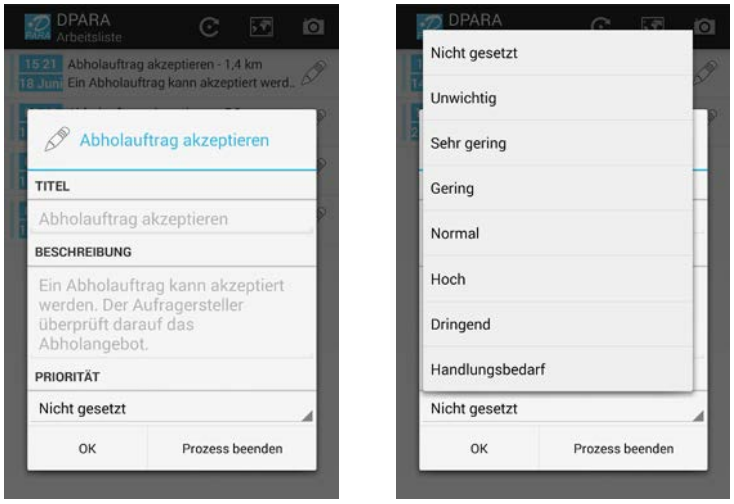
```
worklistItems = new ArrayList<WorklistItem>();
this.worklistAdapter = new WorklistAdapter(this, R.layout.row, worklistItems,
     new MainViewInterface() {
@Override
public void abortProcess(String errorMsg, String InstanceID) {
    MainView.this.instanceID = InstanceID;
    new BackgroundAsyncTask(4, errorMsg).execute();
}

@Override
public void updateProcessSettings(WorklistItem w, String title, String
    description, String individualPriority) {
    new BackgroundAsyncTask(7, null, w, title, description,
        individualPriority).execute();
}
});
```

Listing 5.9: Aktualisierung der Prioritätseinstellungen über ein Interface

5.4.2.5. Shared Preferences

In den `Shared Preferences` kann die Applikation Parameter ablegen, welche die allgemeine Bedienung erleichtern. Ein Nutzer der sich in DPARA erstmalig anmeldet, muss Benutzernamen und Passwort angeben. Beim zweiten Anmeldevorgang wird ihm der zuletzt benutzte Nutzername vorgeschlagen. Gleichermaßen werden vom Nutzer gewählte Einstellungen, wie Filteroptionen, hier abgelegt. Beim Neustart können diese wiederhergestellt werden.



(a) Aktivitäten Kontextmenü (b) Prioritätsauswahl

Abbildung 5.11.: Prioritätsbestimmung in DPARA

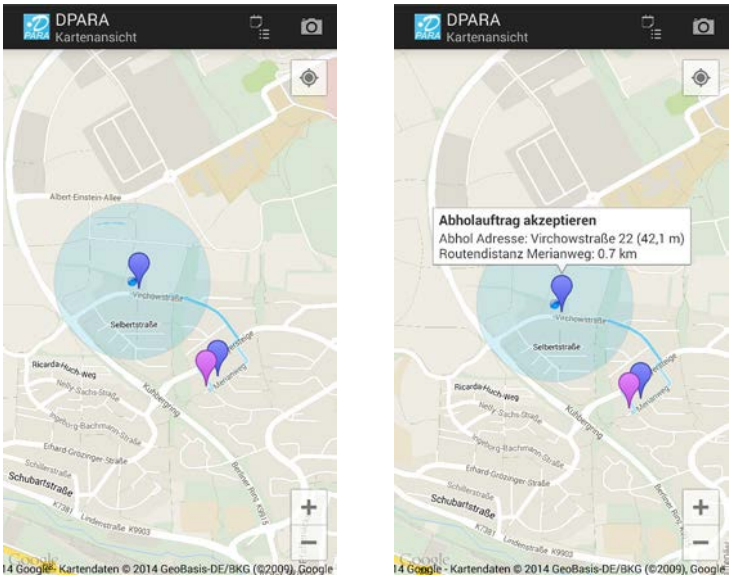
5.4.3. Kartenansicht

Die Kartenansicht spielt eine wichtige Rolle in DPARA. Sie bietet dem Nutzer zusätzlich zu der AR-Sicht, eine übersichtlichere Darstellung der PLIs eines Prozesses (siehe Abbildung 5.12). Die von AREA mitgelieferte Kartenansicht wurde für DPARA um einige Features erweitert, damit der Nutzer im Kurierdienst Anwendungsszenario optimale Unterstützung erhält. Die implementierten Features werden hier im Detail erläutert.

5.4.3.1. Positionsmarkierungen und Infobox

Auf der Karte werden Positionsangaben klassisch durch einen `Marker` gekennzeichnet (siehe Abbildung 5.12(a)). „Tippt“ der Nutzer auf einen Marker erscheint eine Infobox (siehe Abbildung 5.12(b)), in welcher er detaillierte Angaben zum PLI entnehmen kann. Zu diesen zählen der zugeordneten Prozesstitel, der Parametername der Adresse und die formatierte Adressausgabe mit der Distanz zum aktuellen Nutzerstandort. Handelt es sich zusätzlich um einen verknüpften PLI, kann DPARA eine Routendistanz berechnen.

Diese wird unter der ausgewählten Adresse angezeigt. „Tippt“ der Nutzer auf die Infobox öffnet sich ein `Alert Dialog`, welcher dem Nutzer diverse Navigationsoptionen anbietet. Über jenes Menü kann er direkt in die `ActivityView` gelangen, um dort die Aktivität direkt bearbeiten zu können. Des Weiteren kann er zwischen verknüpften PLIs wechseln. DPARA springt dann automatisch zum ausgewählten Kartenausschnitt. Im Anwendungsszenario kann der Nutzer somit bequem zwischen Abhol- und Zieladresse wechseln.



(a) Positionsmarkierungen (b) Infobox einer Markierung

Abbildung 5.12.: Positionsmarkierungen und Infobox

5.4.3.2. Routeneinzeichnung

Wurde ein Abholauftrag erstellt und damit die Abhol- und Zieladresse definiert, kann DPARA über die `DirectionAndPlacesLibrary` [1] eine Route berechnen und diese in der Kartenansicht anzeigen. Die Routenberechnung orientiert sich dabei an den Straßen, kann aber ohne Aufwand auf Fußwege umgestellt werden. Wurde die Kartenansicht direkt über die Listenansicht gestartet, werden alle Routen, der in der Liste enthaltenen

PLIs, angezeigt. Bearbeitet der Nutzer einen Prozess in der `ActivityView` besitzt er die Möglichkeit die Kartenansicht aufzurufen. In dieser werden nur PLIs zum aktuellen Prozess angezeigt. Besitzt eine Aktivität nur eine Adresse, oder weitere Adressen sind im aktuellen Prozessschritt ausgeblendet, wird keine Route berechnet. Hinsichtlich des Anwendungsszenarios wurde noch ein weiteres Feature bezüglich der Routeneinzeichnung implementiert. Befindet sich der Kurier in der Aktivität „Paket abholen“ oder „Paket ausliefern“ , wird die Route vom Kurier zum jeweiligen Ziel rot auf der Karte eingezeichnet (siehe Abbildung 5.13).

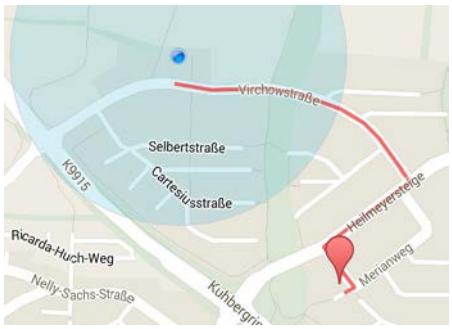


Abbildung 5.13.: Routendarstellung bei der Paketauslieferung

5.4.3.3. Persönlicher Radius

Der persönliche Radius dient in DPARA dem Nutzer als visueller Hinweis (siehe Abbildung 5.13). Alle Aktivitäten, welche sich in diesem Radius befinden und deren PLIs als `mandatory` gekennzeichnet wurden, können vom Nutzer positiv abgeschlossen werden. Befindet sich eine solche Aktivität außerhalb des eingestellten Radius, bekommt der Nutzer einen entsprechenden Warnhinweis, dass die Aktivität von seiner aktuellen Position aus nicht abgeschlossen werden kann. Der persönliche Radius wird in der `Config` eingestellt. für das Anwendungsszenario wurde ein Radius von 200 Metern gewählt. Der Radius kann nach Belieben geändert werden. Auf diese Weise kann in DPARA eine Aktivität in direkte Abhängigkeit zu einem PLI gebracht werden. Es besteht die Möglichkeit einzustellen, dass ein Nutzer eine Aktivität erst öffnen bzw. bearbeiten

kann, wenn diese sich seinem persönlichen Radius befindet. Diese Variante wurde in der Kurierdienst Applikation nicht verwendet.

5.4.4. Kameraansicht

In diesem Kapitel wird die AR-Komponente von DPARA näher erläutert. Der Nutzer hat die Möglichkeit PLIs einer Aktivität in der Kameraansicht zu betrachten. Wie auch schon in der Kartenansicht kann der Nutzer frei wählen, ob alle in der Arbeitsliste enthaltenen PLIs oder nur die einer bestimmten Aktivität angezeigt werden sollen. Zusätzlich kann über den durch AREA bereitgestellten Distanzfilter die Auswahl weiter eingeschränkt werden. Ein PLI wird in der Kameraansicht als blauer Kreis mit Label dargestellt.

Abbildung 5.14.: Ansicht eines PLI in der Kameraansicht

In diesem befinden sich der Titel der Aktivität, eine formatierte Adressausgabe und die Distanz zum aktuellen Ziel. Der aktive PLI ist dabei immer gelb umrandet (siehe Abbildung 5.14) und das Label in seiner Transparenz gegenüber anderen sichtbar hervorgehoben (siehe Abbildung 5.15(b)). Dies garantiert, dass die zu bearbeitende Aktivität schnell gefunden wird und der Text des Labels gut lesbar ist. Befindet sich die Auswahl

nicht im Sichtbereich des Nutzers, wird er durch Navigationspfeile (siehe Kapitel 5.4.4.3) am Rand des Bildschirms auf die Richtung hingewiesen in die er das Smartphone bewegen muss, um den aktiven PLI in den Sichtbereich zu bringen. Im unteren Bereich der Kameraansicht befindet sich zusätzlich eine Infobox, welche die gleichen Informationen wie das Label, um eine zusätzliche Beschreibung der Aktivität ergänzt, enthält. Dies ist notwendig, da der Nutzer nicht ständig das Smartphone auf den PLI richten kann. Auf diese Weise hat er trotzdem die für ihn wichtigen Informationen immer im Sichtbereich.

(a) Große Entfernung (b) Geringe Entfernung

Abbildung 5.15.: Entfernungsdarstellung in DPARA

Über eine Touchgeste im Infobox Bereich des PLIs kann die Aktivität direkt bearbeitet werden. Wurden mehre PLIs für die Anzeige geladen, werden in der Infobox zusätzlich zwei Pfeile eingeblendet, über welche er diese sequentiell durchblättern kann. Über die Menütaste besteht zusätzlich die Option diese nach AR-Distanz, Nutzerentfernung, Aktivierungszeit und Priorität zu sortieren. AR-Distanz bedeutet, dass beim sequentiellen durchblättern immer der im Sichtfeld befindliche PLI ausgewählt wird, der zum aktuellen Zeitpunkt die geringste visuelle Entfernung aufweist. Benutzt der Nutzer Berührungsgesten um über die Kreissymbole PLIs auszuwählen, wird dieser gelb umrandet und die Infobox aktualisiert sich automatisch. Durch diese Features kann der Nutzer, bei eng zusammenliegenden PLIs, mit der Infobox navigieren und erzielt dadurch eine höhere Treffsicherheit.

5.4.4.1. Visuelle Darstellung der Entfernung

Um die Entfernung eines PLIs visuell zu veranschaulichen, werden die blauen Kreise in der Kameraansicht mit zunehmender Entfernung kleiner (siehe Abbildung 5.15). Auf diese Weise kann der Nutzer abschätzen welcher PLI näher gelegen ist, ohne die Distanzwerte vergleichen zu müssen. Die Größe des Kreises steht in Relation zur Distanz, darf dabei aber festgelegte Schwellenwerte nicht über- und unterschreiten. Die Schriftgröße der Infobox ist trotz unterschiedlicher Distanzen immer identisch.

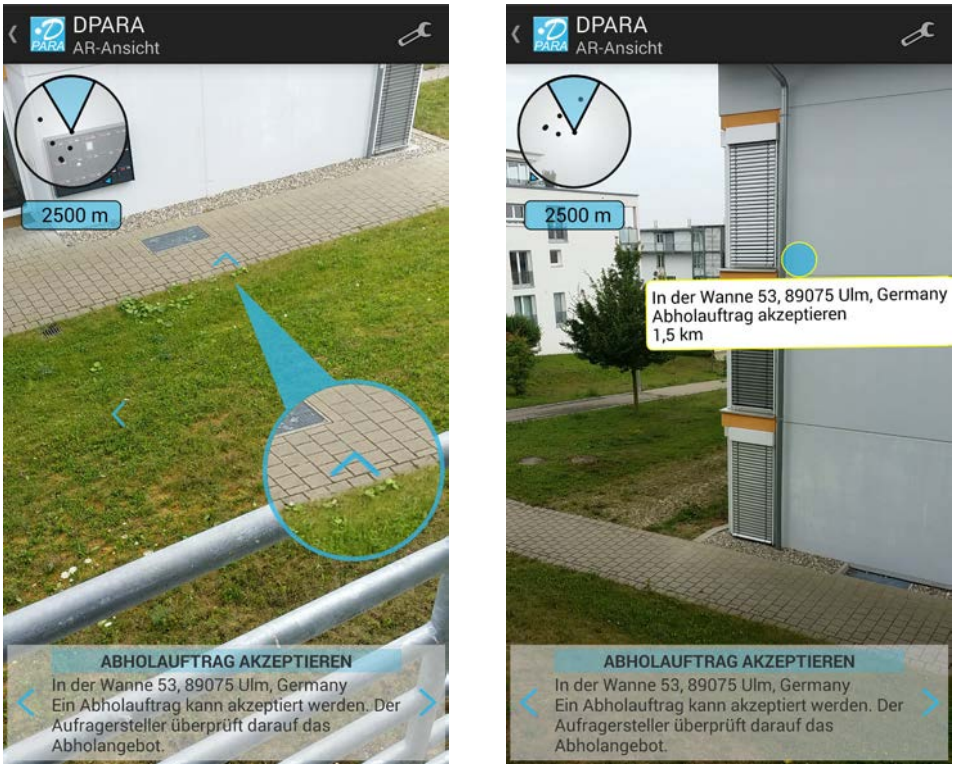
5.4.4.2. Direkte Prozessbearbeitung

Wird die Infobox in der Kameraansicht vom Nutzer per Berührungsgeste ausgewählt, öffnet sich ein Fenster mit näheren Details zur Aktivität (siehe Abbildung 5.16). Zusätzlich zur Entfernung wird hier noch die Höhe des PLIs angegeben. Wird nun die Option „Prozess bearbeiten“ gewählt, gelangt der Nutzer direkt in die `ActivityView` der zugehörigen Aktivität des Prozesses. Die direkte Eingabe von Parameterwerten in der Kameraansicht wurde in DPARA nicht umgesetzt, da dies bei einer größeren Anzahl schnell unübersichtlich werden kann.

Abbildung 5.16.: Direkte Prozessbearbeitung in der Kameraansicht

5.4.4.3. Navigationspfeile

Bei der Navigation im Kameramodus kann es vorkommen, dass man die gewünschte Aktivität aus den Augen verliert. Für diesen Fall wurde in DPARA ein Feature eingebaut, welches mittels Navigationspfeilen (siehe Abbildung 5.17(a)) den Nutzer in der Orientierung unterstützt.



(a) Anzeige der Navigationspfeile (b) Navigation zum PLI abgeschlossen

Abbildung 5.17.: Orientierung mittels der Navigationspfeile in der Kameraansicht

Um das zu erreichen wurde der `LocationController` von AREA in der Methode `onSensorsChanged()` erweitert. Zusätzlich wurden die `AREAGeoLocation` Klasse um vier boolesche Werte erweitert. Mit dessen Hilfe kann eine Aussage getroffen werden, ob sich der PLI rechts, links, oben oder unten vom Sichtfeld des Nutzers befindet (siehe Listing 5.10). Diese Information wird dann genutzt, um die jeweiligen Pfeile einzublen-

den, welche dem Nutzer die Lage des PLIs weisen. Jener soll das Smartphone in die angegebene Richtung bewegen. Sobald der PLI sich teilweise im Sichtfeld befindet verschwinden die Navigationspfeile. Des Weiteren werden diese immer nur zum aktuell markierten PLI angezeigt.

```
1  }else{
   if (locHorizontalHeading <= leftAzimuth)loc.setLeftFromView(true);
3  else loc.setLeftFromView(false);
   if (locHorizontalHeading >= rightAzimuth)loc.setRightFromView(true);
5  else loc.setRightFromView(false);
   if (locVerticalHeading >= topAzimuth)loc.setTopFromView(true);
7  else loc.setTopFromView(false);
   if (locVerticalHeading <= bottomAzimuth)loc.setBottomFromView(true);
9  else loc.setBottomFromView(false);
   locations1.add(loc);
11 }}
```

Listing 5.10: Ermitteln der Lage eines PLIs abseits des Sichtfeldes

6

Vorstellung der Applikation

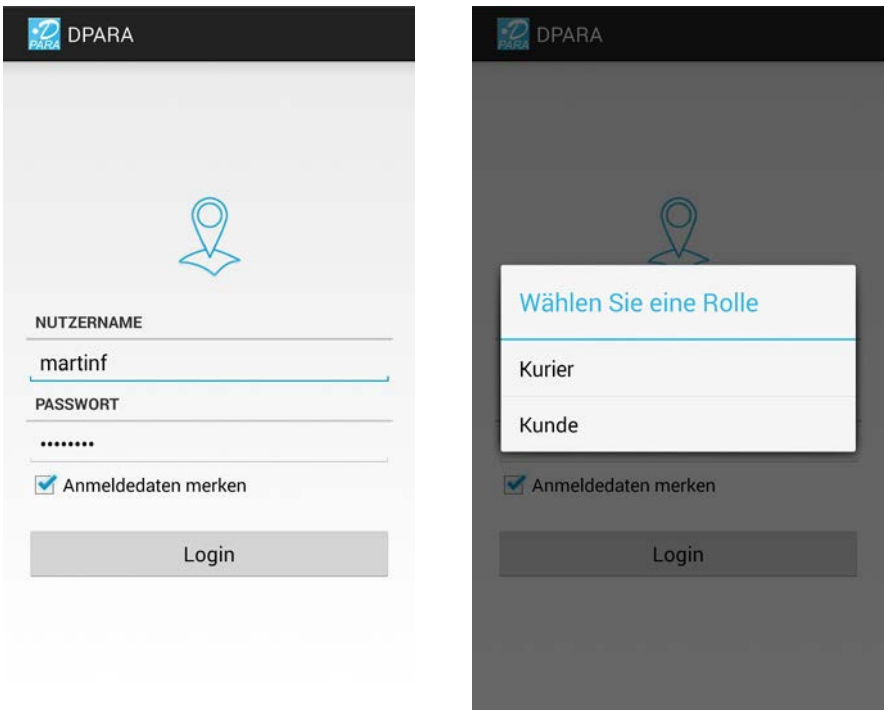
In diesem Kapitel wird DPARA aus Sicht des **Kunden** und des **Kuriers** anhand eines Paketabholvorganges vorgestellt. Der Prozessablauf wird mit Hilfe von Screenshots und Fotos dokumentiert und die Aktionen des Kunden bzw. Kuriers im Detail erläutert. Dies umfasst folgende Punkte:

1. Login und Auftragserstellung
2. Akzeptieren und Überprüfung des Auftrages
3. Navigation zur Abholadresse
4. Erfassung der Kundenunterschrift und Auslieferung

Um das Zusammenspiel mit AristaFlow zu veranschaulichen, wird zusätzlich an den wichtigsten Punkten eine Abbildung aus dem Prozessmonitoring präsentiert. Das Ausliefern des Paketes wurde nicht weiter beschrieben, da es analog verläuft. Funktionen, die

in diesem Beispiel nicht vorkommen, können im Kapitel 5 (Implementierung) eingesehen werden.

6.1. Login und Auftragserstellung



(a) DPARA Login (b) Auswahl der Rolle

Abbildung 6.1.: Login des Kunden

Zu Beginn gibt der Kunde Nutzernamen und Passwort in der Login-Aktivität an (siehe Abbildung. 6.1(a)). Über eine `CheckBox` kann er festlegen, dass beim nächsten Anmeldevorgang der Nutzername automatisch vom System eingetragen wird. DPARA prüft nun die Benutzerdaten und präsentiert die ihm zugeordneten Rollen zur Auswahl (siehe Abbildung 6.1(b)). In der Listenansicht angekommen, startet der Kunde den Prozess „Abholauftrag erstellen“ und bestimmt nun die Abhol- und Zieladresse, sowie das Da-

tum und die Zeit. Abbildung 6.2 zeigt diesen Vorgang im laufenden Programm und im Prozessmonitor von AristaFlow. Die gelb markierte Aktivität befindet sich in Ausführung und benötigt die abgebildeten Datenelemente als Eingabe. Die Eingabe kann entweder über die `EditText`-Felder geschehen oder der Kunde ruft über den `Button` „Adresse wählen“ das `LocationSelection`-Widget auf. Hier kann er über eine Kartenansicht die Abhohl- und Zieladresse bestimmen (siehe Abbildung. 6.2).

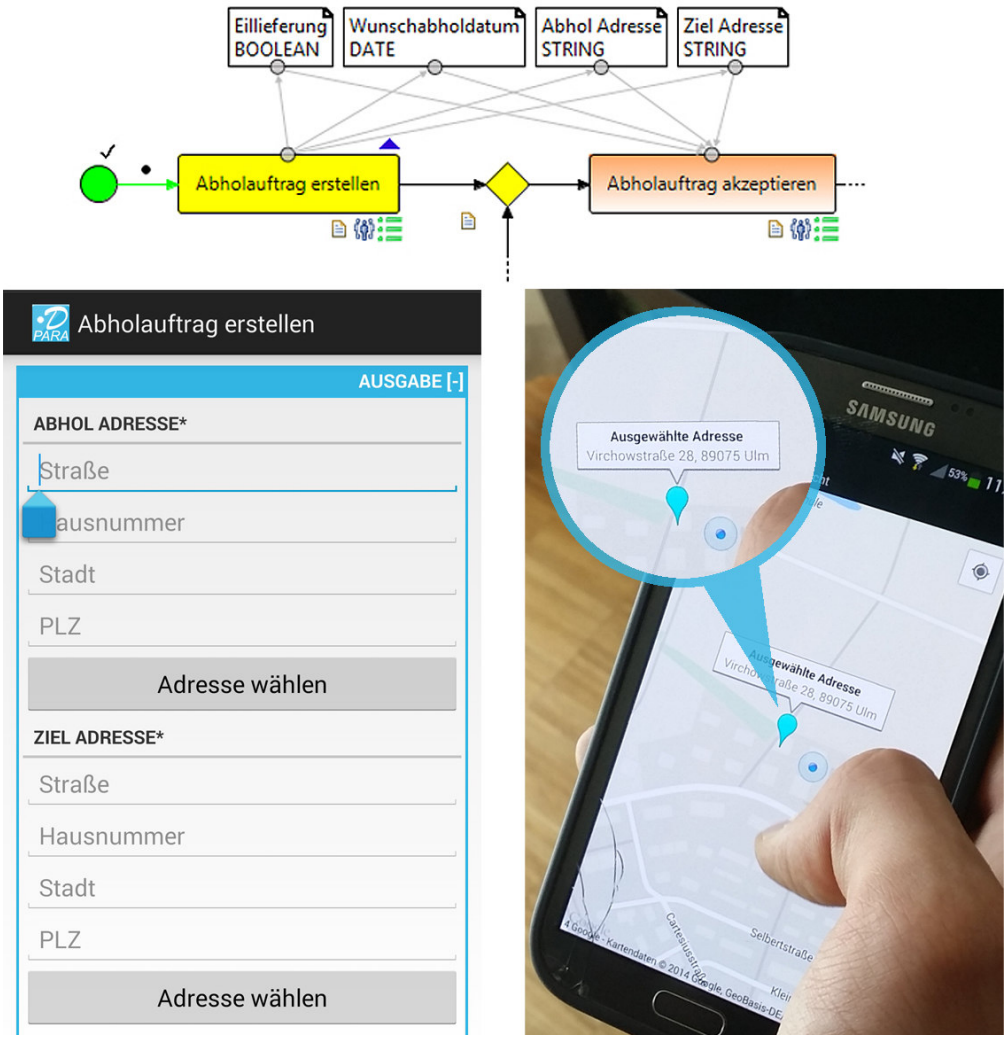
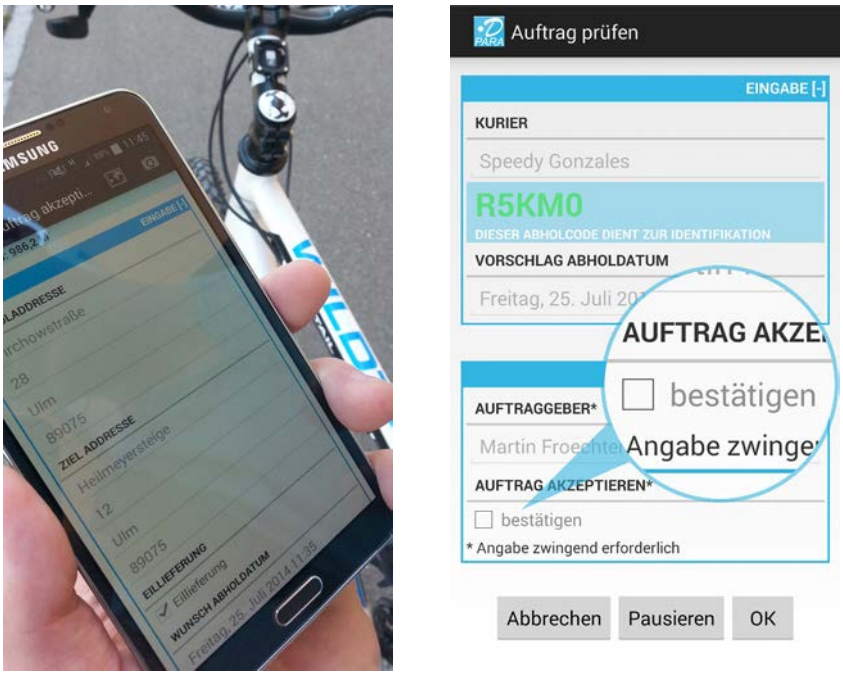


Abbildung 6.2.: Ausfüllen des Abholauftrages

6.2. Akzeptieren und Überprüfung des Auftrages

Nachdem der Kurier den Auftrag entgegengenommen und ein Abholdatum festgelegt hat (siehe Abbildung 6.3(a)), kann der Kunde die Auftragsdetails erneut prüfen. Abbildung 6.4 zeigt diesen Vorgang im Prozessmonitoring. In diesem Fall hat der Kunde den Auftrag erst beim zweiten Anlauf bestätigt (siehe Abbildung 6.3(b)). Der erste Abholauftrag wird vom Kunden, unter Angabe eines Ablehnungsgrundes und einem expliziten Wunschabholdatums, abgelehnt.



(a) Kurier akzeptiert den Abholauftrag (b) Kunde prüft die Auftragsdetails

Abbildung 6.3.: Login des Kunden

Der Kurier bekommt nun zunächst den Ablehnungsgrund und das Wunschabholdatum angezeigt und kann nun wiederholt sein Abholangebot, auf den Wunsch des Kunden abgestimmt, aufgeben. Der Kunde bestätigt nun dieses und der Kurier kann die Navigation zur Abholadresse starten.

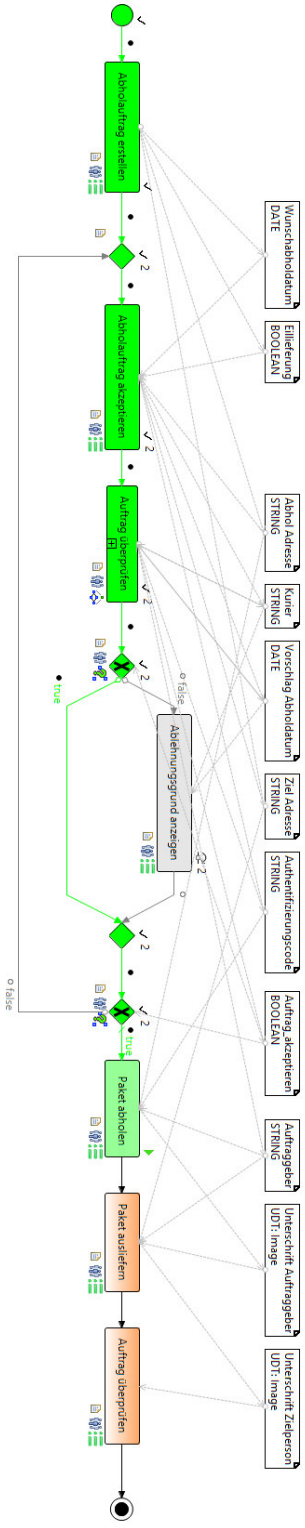
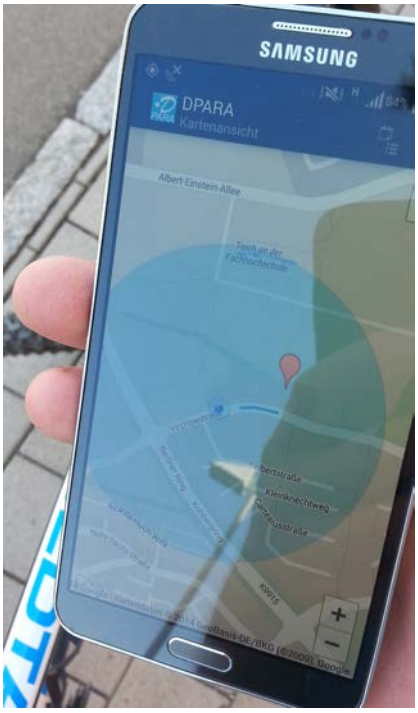


Abbildung 6.4.: Kunde hat den Abholauftrag erst beim zweiten Anlauf bestätigt

6.3. Navigation zur Abholadresse

Um einen Überblick zu bekommen, betrachtet der Kurier den Abholauftrag in der Kartenansicht. Hier sieht er die Abholadresse als roten Marker und die eingezeichnete Route. Die Adresse befindet sich bereits in seinem persönlichen Radius (siehe Abbildung 6.5(a)). Verliert er sein Ziel aus den Augen, bekommt er durch Navigationspfeile auf dem Bildschirm einen Richtungshinweis (siehe Abbildung 6.5(b)).



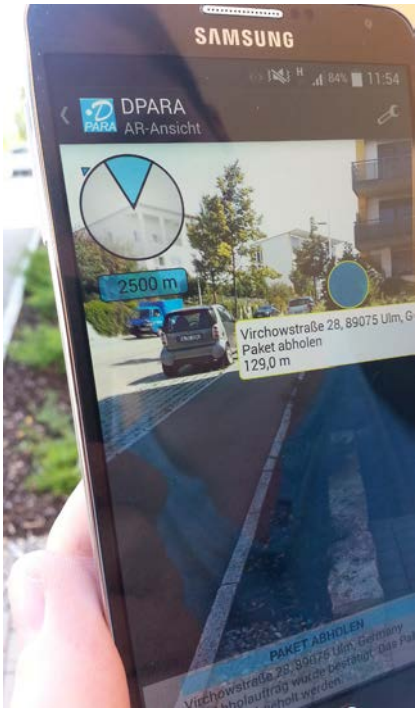
(a) Kartenansicht mit Abholadresse und Routeneinzeichnung

(b) Navigationspfeil hilft bei der Orientierung, da die Abholadresse sich außerhalb des Sichtfeldes befindet

Abbildung 6.5.: Kurier akzeptiert Abholauftrag und startet die Navigation

Abbildung 6.6 zeigt wie der Kurier sich mittels der AR-Sicht in DPARA orientieren kann. Der blaue Punkt markiert die Abholadresse und das dazugehörige Label gibt Auskunft über die verbleibende Distanz.

(a) Kurier navigiert zum Ziel



(b) AR-Sicht der Abholadresse und Anzeige der Infobox

(c) Kurier befindet sich wenige Meter vor der Abholadresse

(d) Abholadresse wird durch blauen Kreis markiert

Abbildung 6.6.: Kurier fährt mit Hilfe der AR-Sicht zur Abholadresse

6.4. Erfassung der Kundenunterschrift und Auslieferung

Bei der Paketannahme erfasst der Kurier die Unterschrift des Kunden (siehe Abbildung 6.7). Das Paket kann nun ausgeliefert werden (siehe Abbildung 6.8). Der Kurier orientiert sich mit Hilfe der Karten- und AR-Sicht und navigiert dann zur Zieladresse. Für die Paketabgabe wird eine Unterschrift der Zielperson benötigt. Der Kunde kann diese in der Aktivität „Auftrag überprüfen“ einsehen, der Prozess ist damit beendet.

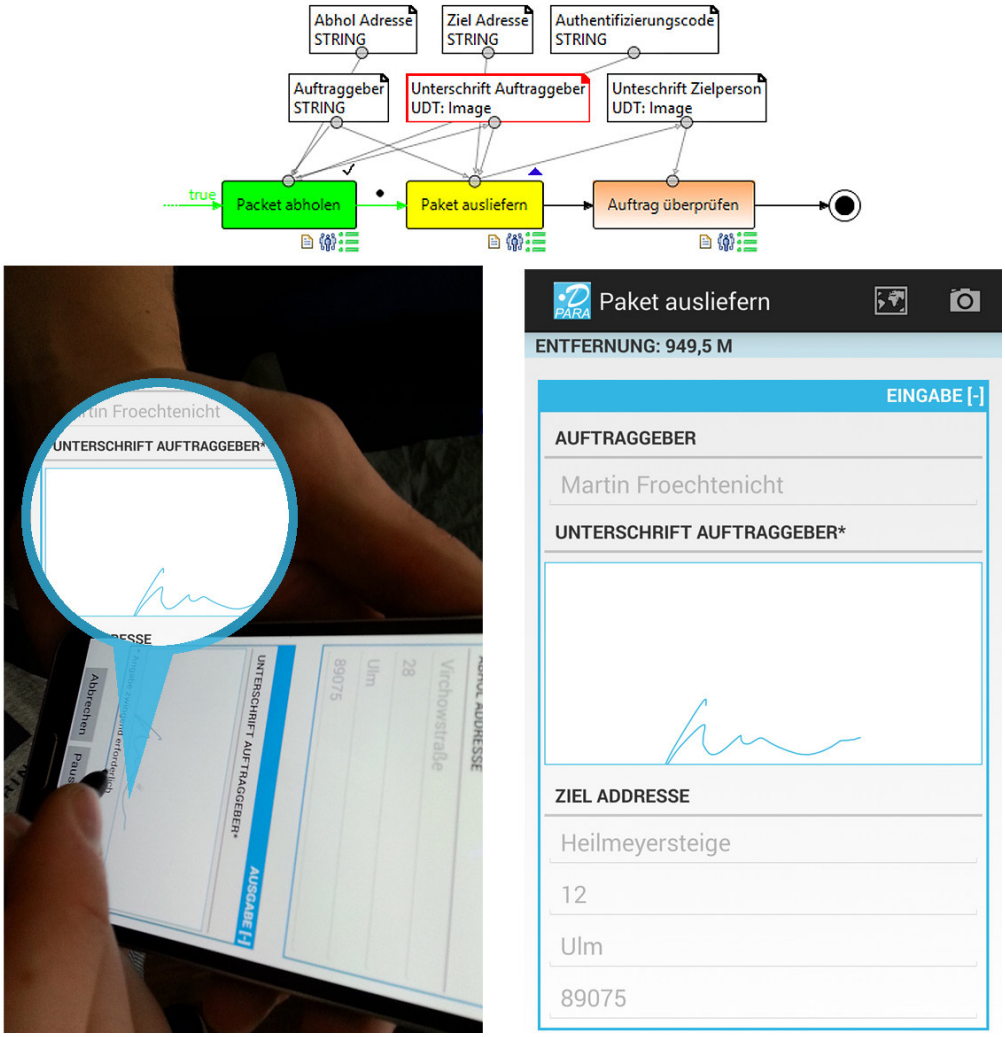
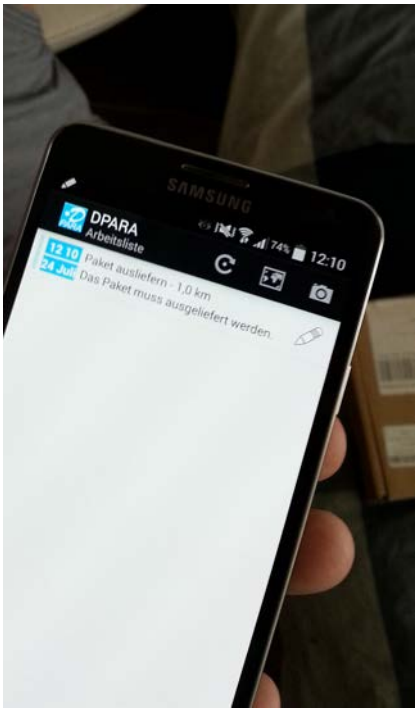
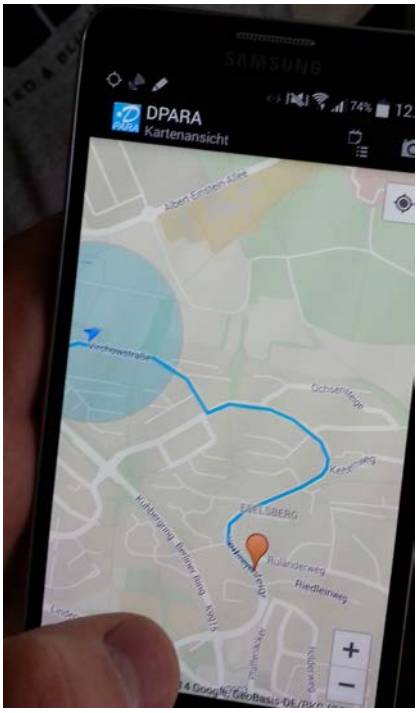


Abbildung 6.7.: Erfassung der Kundenunterschrift

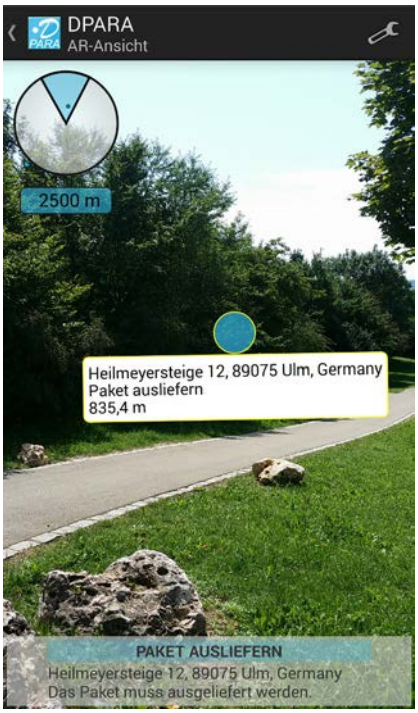


(a) Das Paket kann nun ausgeliefert werden



(b) Kartenansicht der Lieferadresse

(c) Kurier orientiert sich in der AR-Sicht



(d) Kurier navigiert zum Zielort um dort das Paket abzuliefern

Abbildung 6.8.: Auslieferung des Paketes

7

Anforderungsabgleich

In diesem Kapitel werden die Anforderungen, die in Kapitel 2 definiert wurden, hinsichtlich der Implementierung und der Funktionalität abgeglichen. Neben der Beschreibung der Anforderung wird angegeben, ob diese erfüllt werden konnten und ein Verweis zur Umsetzung angegeben. Die folgende Tabelle 7.1 zeigt, dass alle Anforderungen an DPARA als mobiler Workflow-Client erfüllt wurden.

Nr.	**Beschreibung**	**Analyse**
1	Ansicht und Modifizierung von Prozessen im Listen-, Karten und AR-Modus **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4)
2	AR-Sicht nicht mit Informationen überfluten **(nichtfunktional)**	Anforderung erfüllt (siehe Kapitel 5.4)

3	Schema zur Kennzeichnung bzw. Zuordnung von Positionsdaten in einer Aktivität im AristaFlow Process Template Editors **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.3.5)
4	Kommunikation zwischen Smartphone und dem AristaFlow Webservice **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.1)
5	Anzeige der Distanz zu einem PLI **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.1)
6	Filterung der Prozessinstanzen nach Distanz, Aktivierungszeit und Priorität **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.2.3)
7	Stabilität und Eingabevalidierung **(nichtfunktional)**	Anforderung erfüllt (siehe Kapitel 5)
8	Anpassbarkeit **(nichtfunktional)**	Anforderung erfüllt (siehe Kapitel 5)
9	Gute Bedienung, Verständlichkeit, Erlernbarkeit **(nichtfunktional)**	Anforderung erfüllt (siehe Kapitel 5.4)
10	Orientierungsunterstützung in der AR-View **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.4.3)
11	Unterstützung von Subprozessen **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.2.1)
12	Möglichkeit in AR-View zwischen PLIs zu wechseln **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.3.1)
13	Minimalistisches Design **(nichtfunktional)**	Anforderung erfüllt (siehe Kapitel 6)

Tabelle 7.1.: Anforderungsabgleich hinsichtlich der Rolle als mobiler Workflow-Client

Tabelle 7.2 zeigt, dass die in Kapitel 2 gestellten Anforderungen an DPARA hinsichtlich des ausgewählten Kurierdienstszenarios alle erfüllt wurden.

Nr.	**Beschreibung**	**Analyse**
1	Auswahl der Adressen über eine Kartenansicht **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.1.2)
2	Erfassen einer Kundenunterschrift **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.1.1)
3	Authentifizierung des Kuriers **(funktional)**	Anforderung erfüllt (siehe Kapitel 3.4.1)
4	Neuverhandeln des Abholdatums **(funktional)**	Anforderung erfüllt (siehe Kapitel 3.4.2)
5	Anzeige der Route zwischen PLIs **(funktional)**	Anforderung erfüllt (siehe Kapitel 5.4.3.2)

Tabelle 7.2.: Anforderungsabgleich hinsichtlich des Kurierdienstszenarios

8

Herausforderungen und Problemstellungen

Im Verlauf der Entwurf- und Implementierungsphase gab es viele Herausforderungen und Problemstellungen. Die wichtigsten werden hier in diesem Kapitel aufgelistet und zu jedem Problem der gefundene Lösungsansatz angegeben.

8.1. Galaxy Note 3 Sensorproblematik

Beim Testen von AREA auf einem Galaxy Note 3 wurde festgestellt, dass es zu größeren Abweichungen bei den Interessensmarkierungen in der Kameraansicht kommt (siehe Abbildung 3.2(a)). Die Problematik ist auf ein falsch kalibrierten Kompass zurückzuführen.

Beim Neustart des Geräts wird dieser zurückgesetzt und muss neu eingestellt werden. Das Konfigurationsmenü kann über die Telefon-App und die Tastenkombination *#0*# unter dem Menüpunkt Sensor, aufgerufen werden. Das Telefon muss nun in allen drei Achsen bewegt werden. Bei erfolgreicher Kalibrierung zeigt das Menü die Zahl drei im Kompass an.

8.2. GPS-Signal in Räumen

Das GPS-Signal ist in abgeschlossenen Räumen oft nicht ausreichend. Dies kann bei der Darstellung von PLIs in der Kameraansicht zu größeren Ungenauigkeiten führen. Es gibt bereits Lösungen für diese Problematik (siehe Kapitel 3.2.1). Die sogenannten iBeacons senden über Bluetooth dem Nutzer Informationen über seine Position. Eine exakte Positionsangabe ist aber nicht möglich. Die Genauigkeit kann durch die Anzahl der eingesetzten iBeacons erhöht werden, ist aber für DPARA im momentanen Entwicklungstand nicht ausreichend.

8.3. AristaFlow Webservice Kommunikation

Für die Kommunikation mit dem AristaFlow Webservice wird auf Android Seite eine Bibliothek benötigt, um Simple Object Access Protocol Nachrichten verarbeiten und verschicken zu können. Da Android keine eigene SOAP-Bibliothek anbietet, wurde auf die leichtgewichtige und effiziente Alternative kSoap2 zurückgegriffen. Die Gespräche mit Fabian Maier und die Einsicht in seine Bachelorarbeit [13] zu dieser Thematik waren sehr hilfreich.

8.4. Subprozess Ausführungsproblematik

Subprozesse können zur Ausführung dem `Automatic Client` zugewiesen werden. Dieser muss neben dem AristaFlow Server im Hintergrund laufen.

8.5. Einzeichnung von Navigationspfeilen

Der `LocationController` von AREA gibt alle Interessenspunkte, welche sich im Sichtfeld des Geräts befinden, als `Array` zurück. Für die Einzeichnung von Navigationspfeilen musste dieser so abgeändert werden, dass zusätzlich Informationen über jene außerhalb des Sichtfeldes abrufbar sind.

8.6. Rückgabe der ausgewählten Prioritätswerte

Das Menü zur Einstellung von Prioritätswerten befindet sich im `ListAdapter`. Damit die vom Nutzer ausgewählten Werte durch die `ListActivity` angezeigt und von den Datenmanagern weitergereicht werden können, wurde ein gemeinsames Interface eingerichtet.

8.7. Erfassung von Unterschriften

Im Kurierdienst Szenario soll der Kurier Unterschriften vom Kunden erfassen können. Es wurde ein `Signature`-Widget erstellt, welches Parameter vom Typ `Signature` behandelt.

8.8. Übergabe eines Wertes in einem SOAP-Primitive

Damit einem SOAP-Primitive-Objekt ein Wert übergeben werden kann, musste ein eigener `ValueSerializationEnvelope` erstellt werden. Dieser erkennt jenes Objekt und schreibt den Wert in die zu verschickende SOAP-Nachricht.

9

Zusammenfassung und Ausblick

In diesem Kapitel wird eine Zusammenfassung über das gesamte Projekt und ein Ausblick hinsichtlich der Zukunft von Anwendungen, welche AR zur Anzeige und zur Bearbeitung von Prozessdaten verwenden, gegeben. Am Schluss wird noch auf mögliche Features von DPARA eingegangen, welche während der Konzept- und Implementierungsphase zum Anwendungsszenario entstanden sind, aber im zeitlichen Rahmen der Arbeit nicht umgesetzt werden konnten.

9.1. Zusammenfassung und Ausblick

In dieser Arbeit wurde gezeigt, dass durch die Kombination von **A**ugmented **R**eality (AR) mit prozessorientierten mobilen Anwendungen neue und innovative Konzepte entstehen können, Informationen anzuzeigen und zu bearbeiten. Über das Smartphone oder eine

Datenbrille können Prozessaktivitäten mit Positionsdaten (z.B. Adressdaten oder Koordinaten) in der realen Welt abgebildet werden. Auf diese Weise können Informationen vom Nutzer in ihrem ortsspezifischen Kontext betrachtet werden und stehen für ihn immer zum richtigen Zeitpunkt zur Verfügung. Informationen werden über die Positionsangabe gefiltert und können durch die Interaktion mit der Umwelt abgerufen werden. Eingaben werden von einer Prozess-Engine ausgewertet und die dazu passenden Prozesszweige bzw. Abhandlungsroutinen ausgewählt. Auf diese Weise sind dargestellte Inhalte in der AR dynamisch auf die Bedürfnisse des Nutzers und seine Situation angepasst. Für Anwendungsfelder bei denen Mobilität eine wichtige Rolle spielt, erweist sich diese Art und Weise Prozesse abzuarbeiten als sehr vorteilhaft. Ein Nutzer ist nicht mehr an einen einen Desktop-PC gebunden und kann die Möglichkeiten seines mobilen Devices voll ausschöpfen.

Die in der Arbeit entstandene Anwendung **D**ynamic **P**rocess-oriented **A**ugmented **R**eality **A**pplication (DPARA) zeigt anhand des Kurierdienstszenarios wie das Grundgerüst einer solchen Applikation aussehen könnte. Es wurde versucht, das Potential beider Bereiche, AR und Prozessorientierung, aufzunehmen und die optimale Schnittmenge zu finden, welche einerseits die wichtigsten Funktionalitäten abdeckt und dabei immer übersichtlich und einfach zu bedienen bleibt. Während der Konzeption wurde verstärkt auf jene Funktionen geachtet, die dieses Ziel unterstützen. Eine BPM-Suite wie AristaFlow eignet sich optimal um die Inhalte von DPARA dynamisch bereitzustellen und stellt dabei zusätzlich wichtige Funktionen wie die Organisationsstruktur und eine umfassende Prozessmodellierung. Die **A**ugmented **R**eality **E**ngine **A**pplication (AREA) eignet sich hervorragend um Informationen mit Positionsdaten in einer AR abbilden zu können. Kurier und Kunde können dadurch in DPARA eigenständig agieren und sind nicht abhängig von übergeordneten Organisationspositionen. Navigation und Prozessbearbeitung sind in der Anwendung fließend vereint. Dies spart dem Kurier wertvolle Zeit bei der Auftragsbearbeitung und hilft zusätzlich bei der Planung seiner Aktivitäten. Über das auf der Serverseite initiierte Prozesstemplate kann das Szenario jederzeit erweitert und verbessert werden. Diese Änderungen sind dann in der Anwendung sofort verfügbar, da DPARA dynamisch die vom AristaFlow Webservice erhaltenen Daten verarbeitet und anzeigt. Während den Praxistests von DPARA wurde festgestellt, dass sich die Handha-

bung der AR-Funktion beim Fahrradfahren als kompliziert erweist. Diese Problematik wäre bei einer Datenbrille nicht gegeben, da diese über Eye-Tracking gesteuert wird und der Kurier somit die Hände frei hat. Auch die sogenannten Wearable Devices, wie z.B. eine Smartwatch, wären zu diesem Zweck denkbar. Fast jeder zweite deutsche besitzt ein Smartphone [14], diese Zahl unterstreicht den Trend der Mobilität und die damit zusammenhängende Notwendigkeit, Technologien wie AR einzusetzen, um mobiles Arbeiten auf eine neues Level zu bringen. Anwendungen die sich diesen Ansatz verfolgen, werden in der Zukunft eine wichtige Rolle spielen, den Einsatzgebieten ist dabei keine Grenze gesetzt.

9.2. Mögliche Features in DPARA

Während der Konzept und Implementierungsphase sind viele Ideen zu möglichen Features hinsichtlich des Kurierdienstszenarios entstanden, welche im zeitlichen Rahmen der Masterarbeit nicht implementiert werden konnten. Die wichtigsten werden hier in diesem Kapitel unter der Überschrift Ausblick vorgestellt.

9.2.1. Mehrsprachige Unterstützung

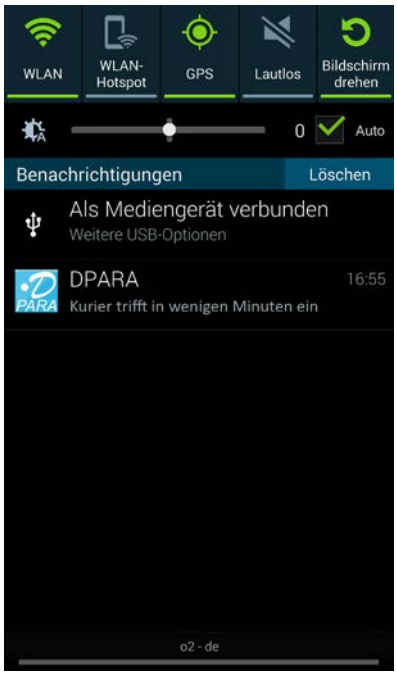
DPARA kann mehrere Sprachen unterstützen. Zu diesem Zweck muss im AristaFlow OrgModel Editor die `locale`-Variable des jeweiligen Nutzerprofils an die bevorzugte Sprache angepasst werden. DPARA fragt diese beim Login über den Webservice an und kann dementsprechend auf die gewünschte Sprache umstellen, falls die Übersetzungen zu allen GUI-Elementen in dieser Sprache vorliegen.

9.2.2. Personen als Zieladresse

Bei einem Abholauftrag kann eine Person als Zieladresse angegeben werden. Diese wird über das Organisationsmodell von AristaFlow identifiziert. Die Position des Nutzers wird bei ausgewählten Aktivitäten im Prozess (z.B. Auslieferung) dem Kurier übermittelt.

Für dieses Feature muss der Nutzer in DPARA der Nutzung seiner Position zustimmen. Auf diese Weise ist er bei einem Abholauftrag nicht mehr an einen festen Ort gebunden, sondern kann seinen täglichen Erledigungen nachkommen.

(a) Kunde sieht Position des Kuriers in der AR-Ansicht



(b) Benachrichtigung über die Ankunft des Kuriers

Abbildung 9.1.: Kunde bereitet sich auf Ankunft des Kuriers vor

Befindet sich der Kurier in direkter Nähe, kontaktiert er ihn über die Anwendung. Durch die Standortfreigabe sieht auch der Kunde die aktuelle Position des Kuriers (siehe Abbildung 9.1). Mit Hilfe dieser Information kann er sich auf die exakte Ankunft des Kuriers vorbereiten.

9.2.3. Eingabe von Parametern in der AR-View

Ein wichtiger Faktor in der AR-Ansicht ist die Übersichtlichkeit. Werden in einer Aktivität nur wenige Parameter zur Bearbeitung benötigt, kann man diese über die Infobox

eingeben. Bei vielen Parametern wird der Nutzer wie gewohnt in die `ActivityView` weitergeleitet. Der Nutzer muss die AR-View nicht verlassen, um Aktivitäten zu bearbeiten.

9.2.4. Zwischenziele

In DPARA wurde für das ausgewählte Szenario eine Abhol- und Zieladresse als Eingabe für einen Abholauftrag definiert. Das Hinzufügen von weiteren Adresszielen könnte als Subprozess ausgelagert werden. Auf diese Weise sind Zwischenziele (z.B. Blumenladen) auf der Route möglich.

9.2.5. Routenanzeige in der Kameraansicht

Bisher ist es in DPARA nur möglich über die Kartenansicht eine Route im Detail zu betrachten. In der Kameraansicht wird zum aktuellen Zeitpunkt noch keine Route in die reale Welt projiziert. Mit diesem Feature wäre man nicht mehr zwingend auf die Kartenansicht angewiesen. Es wurde die AR GPS Drive/Walk Navigation Anwendung [9] getestet und festgestellt, dass die Einzeichnung der Route öfters nicht zum Kamerabild passt. Die Genauigkeit ist hier stark von den Sensoren und einem guten GPS-Signal abhängig.

Abbildung 9.2.: AR GPS Drive/Walk Navigation

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[1] AKEXORCIST. Directionandplaceslibrary. `https://github.com/akexorcist/Android-GoogleDirectionAndPlaceLibrary` - [Online; Zuletzt besucht am 18/7/2014] .

[2] DADAM, P., REICHERT, M., RINDERLE-MA, S., LANZ, A., PRYSS, R., PREDESCHLY, M., KOLB, J., LY, L. T., JURISCH, M., KREHER, U., AND GOESER, K. *From ADEPT to AristaFlow BPM Suite: A Research Vision has become Reality*. In Proceedings Business Process Management (BPM'09) Workshops, 1st Int'l. Workshop on Empirical Research in Business Process Management (ER-BPM '09), Ulm, Germany, September 2009, LNBIP 43, Springer, pp. 529-531, 2009.

[3] FRÖCHTENICHT, M. *Social Application „Wazzup". Hält dich freizeittechnisch auf dem Laufenden!* Bachelorarbeit, Universität Ulm, 2011.

[4] GEIGER, P., PRYSS, R., SCHICKLER, M., AND REICHERT, M. *Engineering an Advanced Location-Based Augmented Reality Engine for Smart Mobile Devices*. Technical Report UIB-2013-09, University of Ulm, 2013.

[5] GEIGER, P., SCHICKLER, M., PRYSS, R., SCHOBEL, J., AND REICHERT, M. *Location-based Mobile Augmented Reality Applications: Challenges, Examples, Lessons Learned*. In 10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps, Barcelona, Spain, April 3-5, 2014, pp. 383-394, 2014.

[6] GOOGLE INC. Asynctask. `http://developer.android.com/reference/android/os/AsyncTask.html` - [Online; Zuletzt besucht am 12/5/2014] .

[7] GOOGLE INC. Base64 android api. http://developer.android.com/reference/android/util/Base64.html - [Online; Zuletzt besucht am 23/7/2014] .

[8] GOOGLE INC. Google geocoding api. https://developers.google.com/maps/documentation/geocoding/?hl=de - [Online; Zuletzt besucht am 18/7/2014] .

[9] HSIEH, WALTER. Ar gps drive/walk navigation, Juni 2014. https://play.google.com/store/apps/details?id=com.w.argps&hl=de [Online; Zuletzt besucht am 17/7/2014] .

[10] JULIER, S., BAILLOT, Y., BROWN, D., AND LANZAGORTA, M. *Information Filtering for Mobile Augmented Reality*. Journal IEEE Computer Graphics and Applications Volume 22 Issue 5, September 2002 Page 12-15, 2002.

[11] KAUFMANN, TIM. Genau, aber nicht perfekt. http://www.golem.de/news/bluetooth-low-energy-ibeacon-ist-mehr-als-ein-leuchtfeuer-1403-105331-3.html - [Online; Zuletzt besucht am 01/8/2014] .

[12] KNEIDINGER, JUERGEN. Insidear 2012, augmented reality for the process industry, Oktober 2012. https://www.youtube.com/watch?v=UvrRgpYtj2w - [Online; Zuletzt besucht am 15/7/2014] .

[13] MAIER, F. *Entwicklung eines mobilen und Service getriebenen Workflow-Clients zur Unterstützung von evaluierten Studien der klinischen Psychologie am Beispiel der AristaFlow BPM Suite und Android*. Bachelorarbeit, Universität Ulm, 2012.

[14] MARC SAUTER. Fast jeder zweite deutsche nutzt ein smartphone. http://www.golem.de/news/google-fast-jeder-zweite-deutsche-nutzt-ein-smartphone-1308-101251.html Reality.aspx - [Online; Zuletzt besucht am 10/8/2014] .

[15] MICROSOFT ÖSTERREICH. Microsoft und augmensys kooperieren bei augmented reality. http://www.microsoft.com/de/news/Press/2014/Jan14/Microsoft-und-Augmensys-kooperieren-bei-Augmented-Reality.aspx - [Online; Zuletzt besucht am 04/8/2014] .

[16] MOSABUA. ksoap2-android. `https://code.google.com/p/ksoap2-android/` - [Online; Zuletzt besucht am 18/7/2014] .

[17] MYSAMPLECODE. Android capture signature using canvas. `http://www.mysamplecode.com/2011/11/android-capture-signature-using-canvas.html` - [Online; Zuletzt besucht am 23/7/2014] .

[18] OLSSON, T., AND SALO, M. *Online User Survey on Current Mobile Augmented Reality Applications*. Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium, 2011.

[19] PH.D. GONZÁLEZ MENDÍVIL, E., NARANJO SOLÍS, R. E., AND RÍOS, H. *Innovative Augmented Reality System for Automotive Assembling Processes and Maintenance: An Entrepreneurial Case at Tec de Monterrey*. Instituto Tecnológico y de Estudios Superiores de Monterrey Av. Eugenio Garza Sada 2501 Sur, Z.C. 64849 Monterrey, Nuevo León, 2013.

[20] PRYSS, R., LANGER, D., REICHERT, M., AND HALLERBACH, A. *Mobile Task Management for Medical Ward Rounds - the MEDo Approach*. In 1st Int'l Workshop on Adaptive Case Management (ACM'12), BPM'12 Workshops, Tallinn, Estonia, 2 September 2012, LNBIP 132, Springer, pp. 43-54, 2012.

[21] PRYSS, R., MUSIOL, S., AND REICHERT, M. *Collaboration Support Through Mobile Processes and Entailment Constraints*. In 9th IEEE Int'l Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'13), Austin, Texas, United States, IEEE Computer Society Press, 2013.

[22] PRYSS, R., TIEDEKEN, J., AND REICHERT, M. *Managing Processes on Mobile Devices: The MARPLE Approach*. In CAiSE'10 Demos, Hammamet, Tunisia, June 2010, 2010.

[23] SCHOBEL, J., SCHICKLER, M., PRYSS, R., NIENHAUS, H., AND REICHERT, M. *Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned*. In 9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps, International

Conference on Web Information Systems and Technologies, Aachen, Germany , 8 - 10 May 2013, pp. 509-518, 2013.

[24] SMARTBEAR SOFTWARE. Soapui. `http://www.soapui.org/` - [Online; Zuletzt besucht am 22/7/2014] .

[25] TUCKER, RICHARD AND MACDONALD, JAMES. Augmented reality, mobile, bpm, sap integration demo, September 2011. `https://www.youtube.com/watch?v=mTflf_PqUYs` - [Online; Zuletzt besucht am 15/7/2014] .

Glossar

AristaFlow BPM Suite

Die AristaFlow BPM Suite ist modular aufgebaut und unterstützt jede Phase eines Prozess-Lebenszyklus, von der Prozessmodellierung bis hin zur Prozessausführung und dem Prozess-Monitoring. Die BPM Suite wurde in dieser Arbeit in ihrem vollem Umfang verwendet. Für die Kommunikation zwischen DPARA und dem AristaFlow Server wurde der AristaFlow Webservice genutzt.

Augmented Reality

Unter Augmented Reality (dt. erweiterte Realität, kurz: AR) versteht man die Erweiterung der menschlichen Realitätswahrnehmung, durch computergestützt ausgewählte Informationen. Darunter fallen alle menschlichen Sinnesmodalitäten. Meistens bezieht man sich aber auf die visuelle Darstellung von Informationen in Form von Einblendungen oder Überlagerungen. In dieser Arbeit wird der Begriff Augmented Reality im visuellen Kontext benutzt.

Augmented Reality Engine Applicatiation

Augmented Reality Engine Applicatiation (kurz: AREA) ist eine AR-Engine, welche Interessenspunkte relativ zur momentanen Position des Benutzers auf dem Display in einer Kameraansicht anzeigen kann. Dazu muss die GPS-Koordinate des Smartphones und des Interessenspunktes bekannt sein. DPARA benutzt AREA um Prozesse, welche mit einer Koordinate verknüpft sind, in der Kameraansicht darstellen zu können.

Dynamic Process-oriented Augmented Reality Application

Dynamic Process-oriented Augmented Reality Application (kurz: DPARA) ist der Name der Applikation, welche wesentliche Konzepte und Ideen umsetzt um Prozessinformationen mittels Augmented Reality auf einem mobilen Endgerät unter Android darstellen zu können.

Head-up-Display

Ein Head-up-Display ist ein Anzeigesystem das Informationen in das Sichtfeld des Nutzers projiziert. Dieser kann über seine Kopfhaltung bzw. Blickrichtung die Anzeige steuern.

kSoap2

Das Android Projekt kSoap2 bietet eine leichtgewichtige und effiziente SOAP Bibliothek für die Android Plattform. DPARA nutzt diese Bibliothek zur Kommunikation mit dem AristaFlow Webservice.

Model View Controller

Das Model View Controller Muster (kurz: MVC) wird zur Strukturierung von Software-Entwicklung in die drei Einheiten Datenmodell, Präsentation und Programmsteuerung verwendet. Ziel ist ein flexibler Programmentwurf, der später Änderungen und Erweiterungen erleichtert.

Process Location Input

Bei einem Process Location Input (dt. Prozess Position Eingabe, kurz: PLI) handelt es sich um eine geographische Positionsangabe, welche einer Aktivität in einem Prozess zugewiesen werden kann.

Simple Object Access Protocol

Beim Simple Object Access Protocol (kurz: SOAP) handelt es sich um ein Netzwerkprotokoll, mit dessen Hilfe zwischen Systemen Nachrichten ausgetauscht werden können.

Smartphone

Ein Smartphone ist ein Mobiltelefon das viele Computer-Funktionalitäten zur Verfügung stellt. Sie werden über einen berührungsempfindlichen Bildschirm gesteuert und verfügen über eine Internetanbindung. Weitere wichtige Funktionalitäten sind die GPS-Navigation und die Kamera.

Stylus

Ein Eingabestift, der zur Bedienung von Touchscreens an Computer und Mobiltelefonen verwendet wird. Durch diesen ist eine präzisere Bedienung als mit den Fingern möglich, da nur die dünne Spitze den Bildschirm berührt.

Webservice

Ein Webservice unterstützt den Informationsaustausch zwischen verschiedenen Anwendungen auf unterschiedlichen Plattformen. Der Zugriff auf den Webservice wird durch das Middleware-Protokoll SOAP gesteuert. Die Definition der Schnittstelle eines Webservice erfolgt in einem XML-Dokument - dabei wird die Gültigkeit des Dokuments durch eine WSDL vorgegeben.

Akronymverzeichnis

AR

Augmented **R**eality (siehe Glossar).

AREA

Augmented **R**eality **E**ngine **A**pplication (siehe Glossar).

DPARA

Dynamic **P**rocess-oriented **A**ugmented **R**eality **A**pplication (siehe Glossar).

PLI

Process **L**ocation **I**nput (siehe Glossar).

SOAP

Simple **O**bject **A**ccess **P**rotocol (siehe Glossar).

A

Quelltext

```
1  if (outputParameters.get(i).getType().equals("DATE")) {
   TextView tv =
3      (TextView) getLayoutInflater().inflate(R.layout.template, null);
   tv.setLayoutParams(lparams);
5  String parameterName = outputParameters.get(i).getName();

7  if (!outputParameters.get(i).isOptional())
       parameterName += "*";
9
   tv.setText(parameterName);
11 output.addView(tv);
   boolean dateAvailable = false;
13 String dateValue = "";

15 if (outputParameters.get(i).getPredefinedValue() != null) {
```

```
        for (int j = 0; j < worklistItem.getActivityInstance().getInputParameter
            ().size(); j++) {
17          if (worklistItem.getActivityInstance().getInputParameter().get(j).
                getName().equalsIgnoreCase(outputParameters.get(i).
                getPredefinedValue())) {
                dateAvailable = true;
19              dateValue = worklistItem.getActivityInstance().getInputParameter
                    ().get(j).getValue();
            }
21      }
    } else {
23      if (outputParameters.get(i).getValue() != null) {
            dateValue = outputParameters.get(i).getValue();
25          dateAvailable = true;
        }
27  }

29  DatePicker dp = new DatePicker(this);
    dp.setLayoutParams(lparams);
31  dp.setCalendarViewShown(false);

33  if (!dateAvailable) {
        Calendar now = Calendar.getInstance();
35      dp.setMinDate(now.getTimeInMillis() - 1000);
        now.add(Calendar.MONTH, 3);
37      dp.setMaxDate(now.getTimeInMillis() - 1000);
    }
39
    TimePicker tp = new TimePicker(this);
41  tp.setIs24HourView(true);
    tp.setLayoutParams(lparams);
43
    DateTimeWidget dtw = new DateTimeWidget(this);
45  dtw.setDp(dp);
    dtw.setTp(tp);
47  dtw.setId(i);

49  // Date patter: 2014-03-19T19:30:28.800+01:00
```

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss", Locale.
    getDefault());

if (dateAvailable) {
    Date formatedDate = null;
    try {
        formatedDate = sdf.parse(dateValue);
    } catch (ParseException e) {
        e.printStackTrace();
    }

    Calendar calendar = Calendar.getInstance();
    calendar.setTime(formatedDate);
    if (formatedDate != null) {
        dp.setMinDate(System.currentTimeMillis() - 1000);
        dp.init(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
            calendar.get(Calendar.DAY_OF_MONTH), null);
        calendar.add(Calendar.DAY_OF_WEEK, 3);
        tp.setCurrentHour(calendar.get(Calendar.HOUR_OF_DAY));
        tp.setCurrentMinute(calendar.get(Calendar.MINUTE));
    }
} else {
    Calendar calendar = new GregorianCalendar(TimeZone.getTimeZone("GMT+2"));
    tp.setCurrentHour(calendar.get(Calendar.HOUR_OF_DAY));
    tp.setCurrentMinute(calendar.get(Calendar.MINUTE));
}

output.addView(dp);
output.addView(tp);
uiElements.add(dtw);
continue;
}
```

Listing A.1: Erfassung eines Parameters vom Typ `DATE`

Name: Martin Fröchtenicht Matrikelnummer: 650564

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 15. September 2014 .....

Martin Fröchtenicht