Evaluierung plattformübergreifender Entwicklungstechnologien für „TeamPlay“

Diplomarbeit

verfasst von:

Paul Lindenberger

Paul Schmutz

07. Mai 2015

Höhere technische Lehranstalt Perg

Fachrichtung Höhere Informatik

Auftraggeber:

Auris IT Consult GmbH

Betreuungslehrer:

DI Michael Stumpfl

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Diplomarbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer andern Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Perg, Mai 2015

_____	_____
Paul Lindenberger	Paul Schmutz

Kurzfassung

Die vorliegende Diplomarbeit befasst sich mit einer Forschung zum Thema plattformübergreifende Softwareentwicklung und wurde im Auftrag des Unternehmens Auris IT Consult GmbH durchgeführt.

Die Auris IT Consult GmbH erstellt Software-Lösungen, um betriebliche Abläufe in anderen Unternehmen zu erleichtern und zu optimieren. Darunter befindet sich das Aktivitätsverwaltungssystem bzw. Zeiterfassungssystem „TeamPlay“. Nachdem dieses Programm nur auf wenigen Desktop-Plattformen lauffähig ist, entstand der Gedanke für eine plattformübergreifende Lösung, damit die Software für Kunden auf möglichst vielen Geräten und Betriebssystemen lauffähig ist.

Das Projektteam mit den Mitgliedern Paul Lindenberger und Paul Schmutz hat die Aufgabe, mit verschiedenen plattformübergreifenden Entwicklungstechnologien jeweils einen Prototyp für „TeamPlay“ zu konzipieren und umzusetzen. Die einzusetzenden Entwicklungstechnologien werden dabei von der Auris IT Consult GmbH festgelegt. Im Zuge dieser Forschungsarbeit sind die unterschiedlichen Entwicklungstechnologien hinsichtlich verschiedener Parameter wie z. B. Entwicklungskosten, Performance, usw. zu bewerten, um die am besten geeignete Technologie zu ermitteln.

Das geplante Ergebnis am Projektende sind lauffähige, mit den gegebenen plattformübergreifenden Entwicklungstechnologien erstellte Prototypen und eine ausführliche Dokumentation über die Vor- und Nachteile der jeweiligen Entwicklungstechnologie. Diese Resultate dienen dem Unternehmen Auris IT Consult GmbH schließlich als Entscheidungsgrundlage für eine der plattformübergreifenden Technologien, mit der das Unternehmen zukünftig Software entwickeln wird.

Abstract

The topic of this project consists of a research work for a company named Auris IT Consult GmbH on the subject of cross-platform development.

Auris IT Consult develops software solutions to facilitate and optimise business processes in other companies. One of these is their time-tracking system and activity management system called “TeamPlay”. Due to the fact that the system supports desktop operating systems only, the idea of introducing a cross-platform solution emerged. So customers can use one software on multiple devices and operating systems.

The project team composed of Paul Lindenberger and Paul Schmutz have the task to design and develop prototypes based on the software “TeamPlay” for different cross-platform development technologies. These technologies are assigned by Auris IT Consult GmbH. In the course of the project work the project team has to rate the quality of these technologies regarding different parameters such as development costs, performance, etc. to identify the most suitable one.

The expected results at the end of the project are functioning prototypes created with the given cross-platform development technologies and a detailed documentation concerning advantages and disadvantages of the technologies. Therefore these outcomes will serve as a basis of decision-making for one of the given cross-platform technologies, which the company will develop software solutions with in the future.

Inhaltsverzeichnis

1 Projektumgebung

1.1 Diplomanden

Abbildung 1: Paul Lindenberger

Paul Lindenberger

Geburtsdaten	7. März 1996
Adresse	Unteres Brunnenfeld 19 4312 Ried in der Riedmark
Email	paul.lindenberger@outlook.com
Schulausbildung	2002 – 2006 Volksschule Ried / Rdm. 2006 – 2010 AHS Linz 2010 – 2015 HTL für Informatik in Perg
Berufserfahrung	2012: Cyberhouse Linz, 4 Wochen 2014: Auris IT Consult GmbH, 3 Wochen

Abbildung 2: Paul Schmutz

Paul Schmutz

Geburtsdaten	17. Februar 1996
Adresse	Schweinberg 15 3313 Wallsee
Email	paul.schmutz@aon.at
Schulausbildung	2002 – 2006 Volksschule Wallsee 2006 – 2010 Donauhauptschule Wallsee 2010 – 2015 HTL für Informatik in Perg
Berufserfahrung	2012: SecureGUARD GmbH, 5 Wochen 2013: SecureGUARD GmbH, 5 Wochen 2014: Auris IT Consult GmbH, 3 Wochen

1.2 Auftraggeber

Unternehmensname	Auris IT Consult GmbH
Email	office@auris-consult.at
Tel.	+43 720 720 585
Unternehmenssitz	Berggasse 2 4400 Steyr

Abbildung 3: Firmenlogo

Das Unternehmen

Die Auris IT Consult GmbH wurde 2003 von Ing. Wolfgang Bräu gegründet.

Mittlerweile ist das Unternehmen an mehreren Standorten und gleichzeitig mehreren Ländern vertreten. Der Hauptsitz liegt in Steyr. Daneben befinden sich zwei weitere Niederlassungen in Baar (Schweiz) und Eggstätt (Deutschland).

Das Unternehmen entwickelt verschiedenste Software-Lösungen, mit denen betriebswirtschaftliche Prozesse der Kunden unterstützt und beschleunigt werden. Der Schwerpunkt in der Softwareproduktion liegt derzeit im .NET-Bereich.

Neben dem professionellen, kundenspezifischen Maßschneidern von Softwareanwendungen wird langfristig Kontakt zu Kunden durch Serviceleistungen und Support gepflegt und aufrechterhalten.

Das Team besteht aus einer jungen Expertengruppe. Durch die überschaubare Größe des Teams wird auf die Anliegen der einzelnen Mitarbeiter Rücksicht genommen, was den Arbeitsfluss stärkt.

Die Produkte der Auris IT Consult GmbH werden für die Kunden individuell angepasst. Ein Beispiel für eine Softwarelösung ist „CashBox“, mit welcher das Kassenmanagement für Gastronomiebetriebe und für Handelsunternehmen realisiert wird. Weiters wird mit der Businessanwendung „TeamPlay“ die Aktivitäten- und Zeiterfassung für Projekte und Arbeitsabläufe eines Unternehmens umgesetzt.

Abbildung 4: Produktlogos einiger der Softwarelösungen der Auris IT Consult GmbH [1] [2] [3]

1.3 Projektumfeld

1.3.1 Schule

Die Diplomarbeit wurde im Rahmen der Matura an der HTL in Perg umgesetzt und stellt einen Bestandteil der Matura dar.

Abbildung 5: HTL Perg Logo [4]

Machlandstraße 48
4320 Perg
Tel. 0 72 62 / 53 9 26

1.3.2 Betreuungslehrer

Für technische und organisatorische Fragen stand während der Erstellung der Diplomarbeit Herr Professor Dip.-Ing. Michael Stumpfl mit seinen Fachkenntnissen für das Projektteam zur Verfügung. Besonders für technische Fragen sowie auch organisatorische Überlegungen setzte sich Herr Stumpfl ein.

Abbildung 6: Betreuungslehrer DI Stumpfl [5]

Prof. Dipl.-Ing. Michael Stumpfl

m.stumpfl@htl-perg.ac.at

Werdegang:

- 1992 - 1997 HTBLA Leonding: EDV und Organisation
- 1997 - 2003 Diplomstudium Informatik
- 2003 - 2009 Softwareentwicklung ruwido austria GmbH

1.3.3 Ansprechpartner des Auftraggebers

Auftraggeber der Diplomarbeit ist die Auris IT Consult GmbH. Aus diesem Grund arbeitete das Projektteam mit dem dortigen Entwicklungsleiter Herrn Ing. Thomas Steininger zusammen. Er leitete Diplomarbeitsbesprechungen wie Statusberichte oder Planungsgespräche und koordinierte einen geoordneten Ablauf der einzelnen Arbeitsschritte.

Ing. Thomas Steininger

t.steininger@auris-consult.at

Abbildung 7: Auftraggeber Ing. Steininger

1.4 Projektorganigramm

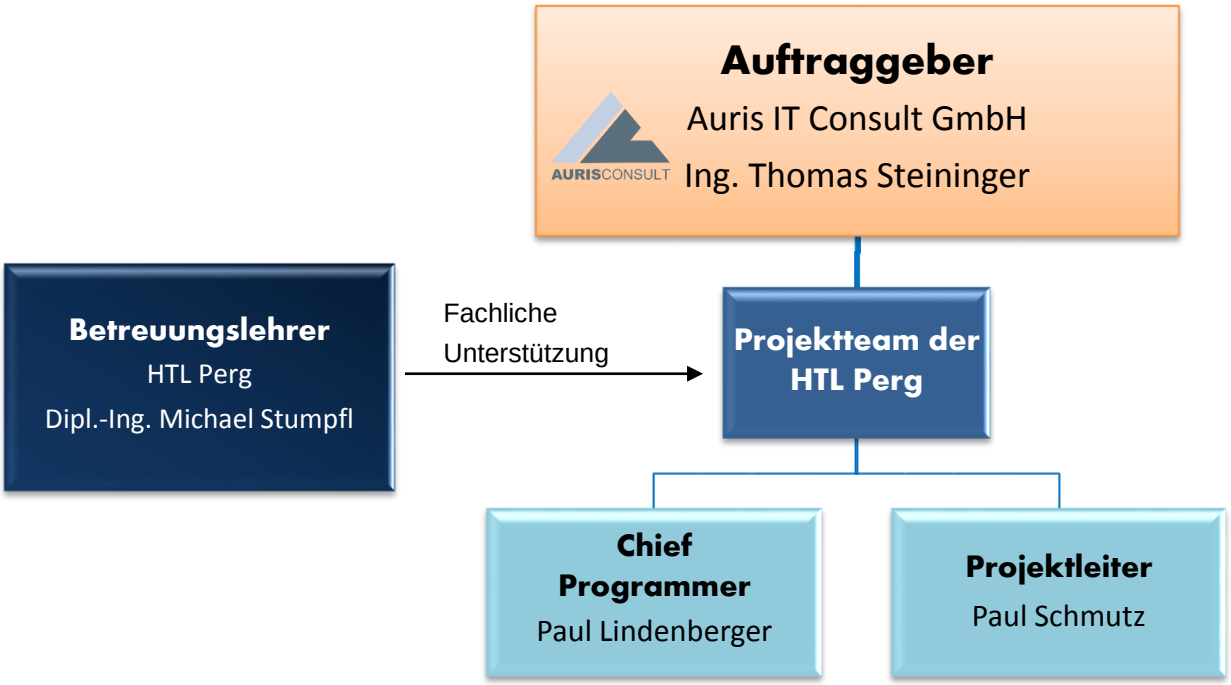


Abbildung 8: Projektorganigramm

1.5 Zuständigkeiten

Zur Veranschaulichung, welche Rolle die einzelnen Projektbeteiligten bei den Tätigkeiten und Funktionen spielten, wird die folgende IMV-Matrix dargestellt.

Tätigkeit/Funktion	Lindenberger	Schmutz	Steininger	Stumpfl
Projektauftrag	MV	MV	IM	I
Projektziele	MV	MV	IM	I
Lastenheft, Pflichtenheft & Projekthandbuch	MV	MV	I	I
Mockups-Konzept	MV	MV	I	I
Prototyp PhoneGap	MV	I	I	I
Prototyp Adobe AIR	I	MV	I	I
Prototyp Eclipse Scout	IM	MV	I	I
Offlinefähigkeitskonzept	I	MV	I	I
Testen	MV	MV	IM	
Dokumentation & Diplomschrift	MV	MV	I	I
Projektabnahme	MV	MV	IM	I

Tabelle 1: IMV-Matrix

I…Informiert M…Mitarbeitend V…Verantwortlich

Die Kommunikation zwischen dem Projektteam (vertreten durch Paul Schmutz) und dem Auftraggeber (vertreten durch Ing. Thomas Steininger) wurde durchwegs über E-Mail abgehalten. In etwa monatlichen Abständen wurden mit dem Firmenpartner Status- und Planungsbesprechungen vereinbart. Diese erfolgten meist persönlich im Haus des Auftraggebers bzw. auch online über eine Skype-Sitzung.

2 Entstehung

2.1 Thema und Aufgabenstellung

Das Thema dieser Diplomarbeit handelt von plattformübergreifender Softwareentwicklung mit dem Fokus auf Mobilgeräte. Wichtig dabei ist, so viele Plattformen wie möglich abzudecken.

Die Aufgabenstellung im ersten Schritt besteht aus der Auswahl geeigneter plattformübergreifender Entwicklungstechnologien. Der Auftraggeber liefert Vorschläge für solche Technologien, über die das Projektteam nähere Informationen recherchiert. Anschließend werden nach gemeinsamer Vereinbarung die zu realisierenden Technologien gewählt.

Das Projektteam hat mit den festgelegten Technologien jeweils einen Prototypen für das Zeiterfassungssystem „TeamPlay“ des Auftraggebers zu entwickeln. Im Zuge des Entwicklungsprozesses wird der Auftraggeber regelmäßig über den Fortschritt und die Erfahrungen bei der Erstellung der Prototypen informiert, damit möglichst rasch die Eignung der Technologien bestimmt werden kann. Während des Projektes muss damit gerechnet werden, dass ein oder mehrere der Technologien unter Umständen ungeeignet sind und die Entwicklung des betreffenden Prototyps abgebrochen wird. In diesem Fall wird je nach zeitlicher Situation eine andere geeignete Technologie gewählt.

Wird aus den evaluierten Prototypen ein Favorit ermittelt, so soll dieser Prototyp noch genauer ins Detail weiterentwickelt werden, um einer realitätsgetreuen Umsetzung von „TeamPlay“ als plattformübergreifende Anwendung zu erzielen.

Da die Offlinefähigkeit und Synchronisation bei mobilen Applikationen ein sehr vorteilhaftes Feature ist, wird das Projektteam für die „TeamPlay“-Anwendung ein Konzept zur Offlinefähigkeit entwerfen.

2.2 Zielsetzung

In Zukunft ist vorgesehen, dass die Auris IT Consult GmbH die Entwicklung von Applikationen mit einer Technologie abwickelt und dennoch für mehrere Plattform verwenden kann. Dadurch können Lösungen basierend auf Kundenwünsche schnell und effizient erreicht werden, was zu einer höheren Kundenzufriedenheit führen soll.

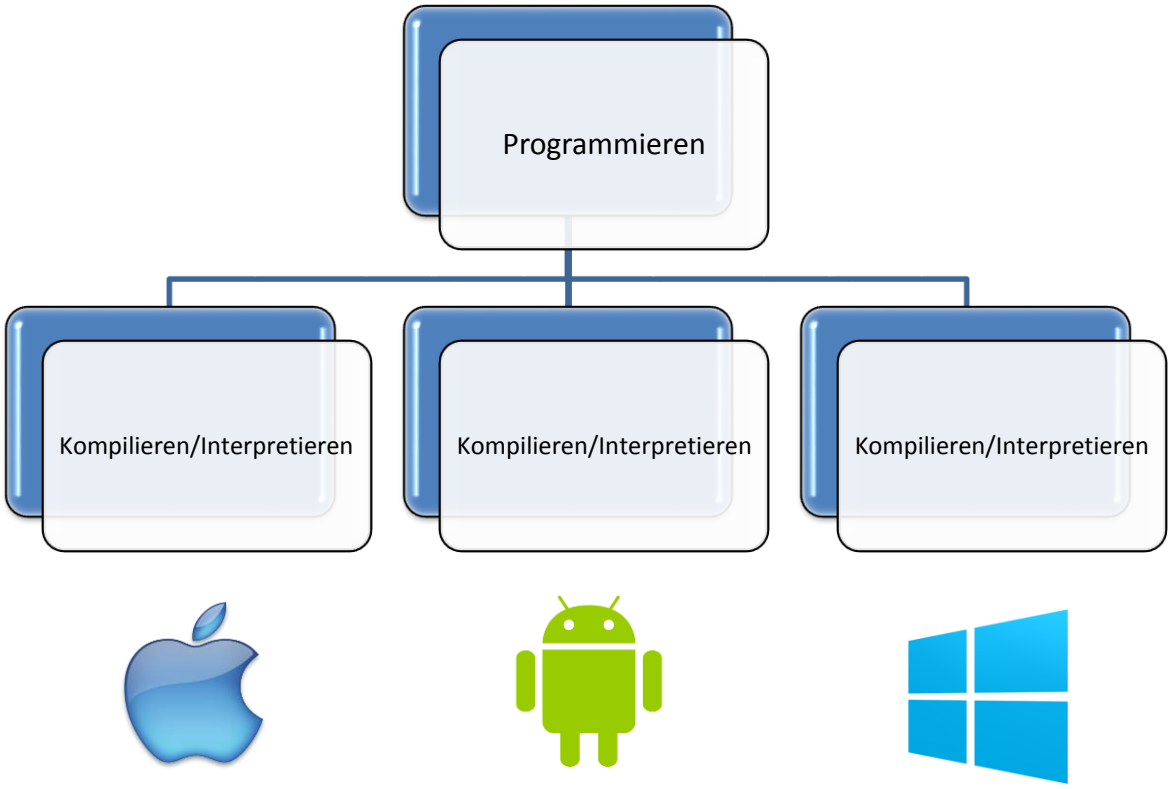


Abbildung 9: Zielsetzung

2.2.1 Geschäftsziele

Das Ziel aus Benutzersicht ist, die grundlegendsten Funktionen des TeamPlay-Systems auch auf mobilen Geräten erledigen zu können.

Dem Benutzer soll ermöglicht werden, alle Projekte und Aktivitäten seines Unternehmens einzusehen und zu suchen sowie Details dazu anzuzeigen. Weiters kann er grundsätzliche Modifikationen an den Aktivitäten vornehmen. Die wichtigste Funktion ist das Management von Zeitbuchungen. Der Benutzer kann eine Buchung starten, stoppen sowie manuell bearbeiten.

Der Vorteil der mobilen Anwendung ist es, einfache und tagtäglich verwendete Funktionen auch bequem mit mobilen Geräten zu erledigen anstatt die aufwändiger zu bedienende Desktop-Anwendung zu verwenden.

Anmerkung: Da es sich nur um Prototypen handelt, gelten eben genannte Geschäftsziele nur für eine abgegrenzte Testumgebung, in der die Mitarbeiter der Auris IT Consult GmbH die Einsatztauglichkeit der Prototypen in der Realität überprüfen.

2.2.2 Produktziele

- Ziele für den Webservice/das Backend
  - lokalen Webservice (Backend) mit beliebiger Technologie (z. B. ASP.NET) aufsetzen
  - repräsentative Menge an Testdaten definieren
  - Stellt Methoden zum Abruf der Testdaten von beliebigen Clients über HTTP bereit
- Ziele für die Prototypen
  - Mitarbeiter-Login
  - Auswahl des Backends (IP-Adresse)
  - Buchungsübersicht anzeigen
  - Buchungsdetails abrufen und aktuelle Buchung stoppen
  - neue Buchung anlegen (nur lokal)
  - Aktivitätsübersicht darstellen
  - Aktivitäten-Suchfunktion funktionstüchtig implementieren
  - Aktivitätsdetails strukturiert anzeigen (Details, Subaktivitäten, Buchungen, Kommentare + jeweils lokales Modifizieren dieser Daten → Details bearbeiten, Buchungen anlegen, Subaktivitäten anlegen, Kommentare anlegen, bearbeiten und löschen)
  - einheitliches, schlichtes, ansprechendes Design
  - ausreichende Performance
  - schnelle Reaktion auf Benutzereingaben
  - selbsterklärende „Smartphone-Bedienung“
  - Unterstützung für Smartphones, Tablets und Desktop

2.3 Ausgangssituation

Das Unternehmen Auris IT Consult steigt in das Thema Cross-Platform-Entwicklung ein, um Apps rasch auf vielen Plattformen zu verbreiten. Bevor jedoch damit begonnen werden kann, wird von vornherein nach der besten Lösung für diese Angelegenheit gesucht, was die Rolle des Projektteams sein soll. Auch Kunden/Partner der Auris IT Consult GmbH suchen nach ähnlichen Ansätzen.

Zur Veranschaulichung des Problems der herkömmlichen App-Entwicklung dient folgende Skizze:

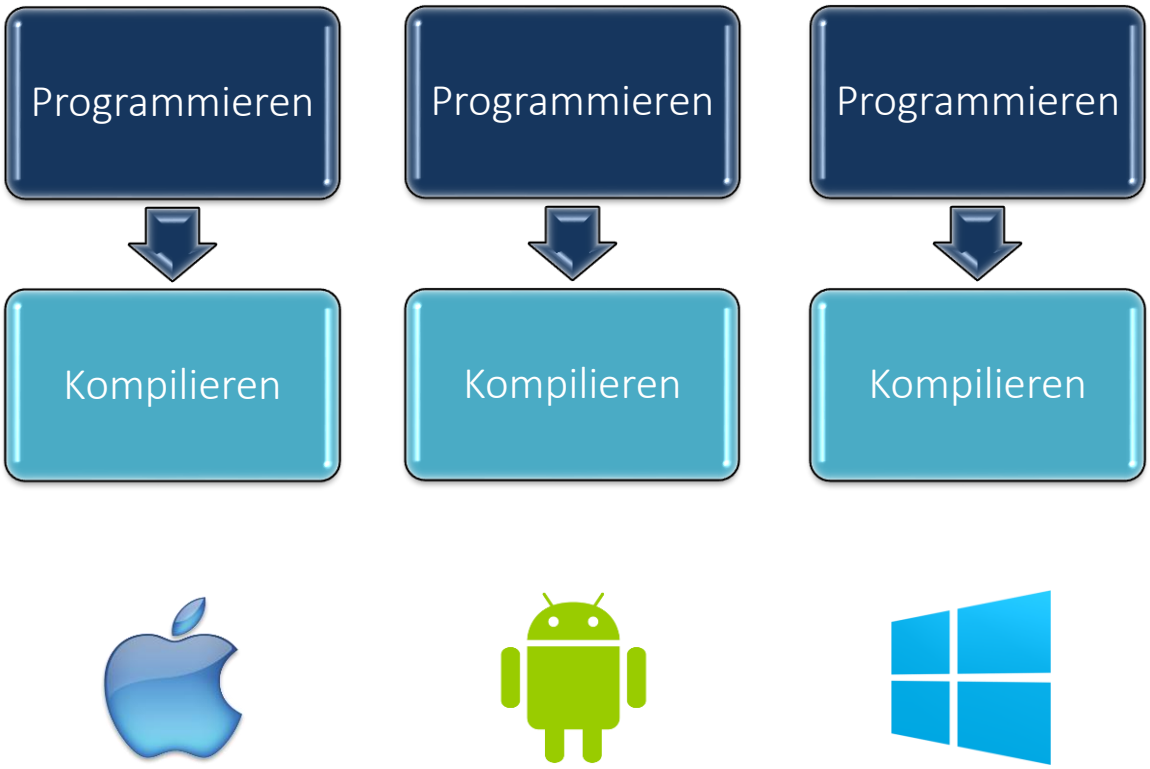


Abbildung 10: Ausgangssituation

Durch diesen mehrfachen Programmier- und Testaufwand werden viele Ressourcen nicht effizient genutzt und der Aufwand liegt bereits für kleinere Applikationen sehr hoch. Ziel dieser Diplomarbeit ist es, für dieses Problem eine hochqualitative Lösung zu finden, durch die der Programmier- und Testaufwand auf ein Minimum begrenzt werden kann.

2.4 Basiskriterien

2.4.1 Auswahl der Technologiekandidaten

Die Hauptaufgabe des Projektteams besteht darin, die plattformunabhängige Entwicklung mithilfe von verschiedenen bestehenden Technologien auszutesten. Dabei stehen einige Technologie-Kandidaten zu Verfügung, die die geforderten Kriterien erfüllen.

Die Technologien haben mindestens folgende Plattformen abzudecken:

- Android
- iOS
- Mac OS
- Windows Desktop
- Windows Phone (optional, aber wünschenswert)

Weiters müssen die Technologien als Resultat für die jeweiligen Plattformen native, installierbare Apps liefern und nicht lose Konstrukte wie z. B. im Web aufrufbare Seiten.

Eine Beschreibungssprache und/oder ein grafisches Entwurfstool sind nicht zwingend notwendig, jedoch wäre diese Unterstützung ein wünschenswertes Kriterium.

Folgende Technologiekandidaten wurden für das Projekt gewählt:

Technologie	Android	iOS	Mac OS	Windows	Windows Phone
Adobe AIR (Action Script)	J	J	J	J	N
PhoneGap mit GWT (Java)	J	J	J	J	J
Eclipse Scout (Java)	J	J	J	J	J
Embarcadero (C++)*	J	J	J	J	N

Tabelle 2: Technologiekandidaten mit Zielplattformen

**Embarcadero ist als Technologie hinzugekommen, da Eclipse Scout frühzeitig ausgeschieden ist. Genaueres befindet sich im Abschnitt 5 Realisierung und Umsetzung.*

2.4.2 Prototypen

Für jeden dieser Technologiekandidaten soll jeweils ein Prototyp entworfen werden, der auf das vom Unternehmen vertriebene Produkt "TeamPlay" aufbaut. Das bedeutet, dass lediglich die Client-Programmierung durchzuführen ist. Primäres Ziel ist eine für alle Plattformen passende Darstellung der „TeamPlay“-Daten.

Für die Aufgabenverteilung der Teammitglieder siehe 3.6.1 Personalressourcen.

3 Planung

3.1 Pflichtenheft

Da zu Beginn des Projektes fundamentale Vereinbarungen mit dem Auftraggeber getroffen wurden, werden in diesem Kapitel einige zentrale Punkte aus dem Pflichtenheft aufgezeigt.

3.1.1 Grenzkriterien

3.1.1.1 Wunschkriterien

Das wichtigste Kriterium der Diplomarbeit ist es, die Entwicklungserfahrungen mit den vereinbarten Technologien ausführlich zu dokumentieren und zu bewerten, damit dem Auftraggeber aussagekräftige Ergebnisse bereitgestellt werden können. In Zukunft möchte der Auftraggeber mit der am besten geeigneten Technologie plattformübergreifende Apps entwickeln.

3.1.1.2 Abgrenzungskriterien

Die folgenden Abgrenzungskriterien schießen über das Ziel hinaus und werden, daher im Projekt nicht berücksichtigt.

- Keine Veröffentlichung im App Store/Play Store
- Kein serverseitiges Programmieren
- Kein Support oder Wartung nach Fertigstellung des Produktes
- Keine Unterstützung für herkömmliche Telefone
- Keine Unterstützung für unbekannte Smartphone-Betriebssysteme

3.1.2 Produkteinsatz

3.1.2.1 Anwendungsbereiche

Die Anwendung wird von den Auftraggebern als Prototyp für die geplante TeamPlay App genutzt. Den Auftraggebern wird es ermöglicht, TeamPlay erstmals am Smartphone zu testen und den Prototyp als Basis für die weitere Entwicklung zu verwenden.

3.1.2.2 Zielgruppen

Die Anwender sollten ein Smartphone mit Internetzugang und einem unterstütztem Betriebssystem besitzen. Weiterhin sind Grundkenntnisse im Umgang mit Smartphone-Applikationen von Vorteil. Als Alternative zum Smartphone kann die Applikation auch unter Windows oder Mac OS benutzt werden.

3.1.3 Produktfunktionen mit GUI-Abbildung

3.1.3.1 Login

Beim Start der Applikation erscheint eine Login Seite auf der sich der Mitarbeiter mit seinem Kürzel und seinem Passwort einloggen kann.

Falls sich der Benutzer bereits einmal eingeloggt hat, kann dieser sich mit einem Klick auf den Login-Button im Hauptmenü als ein anderer Mitarbeiter anmelden.

Die Anmeldeinformationen werden verschlüsselt auf einen permanenten Datenträger gespeichert, um Datenverlust bei unerwartetem Schließen der App zu verhindern.

Abbildung 11: Login

3.1.3.2 Hauptmenü

Das Hauptmenü besteht aus zwei Menüpunkten Zeitbuchungen und Aktivitäten, sowie einem Button zu den Einstellungen und einem zum Login. Im oberen Bereich werden das aktuelle Datum und das Kürzel des aktuell eingeloggten Mitarbeiters angezeigt.

Abbildung 12: Hauptmenü

3.1.3.3 Aktivitäten

Durch einen Klick auf den Menüpunkt Aktivitäten im Hauptmenü gelangt man zur Übersicht aller Aktivitäten in denen man selbst involviert ist oder aktiv ist. Auf dieser Aktivitätenseite gibt es außerdem eine Suchfunktion, die es erlaubt nach Titel und Kunden zu filtern.

Zusätzlich kann durch eine Checkbox eingestellt werden, ob nur eigene Aktivitäten angezeigt werden oder alle.

Abbildung 13: Aktivitätenansicht

Bei einer aktiven Suche wird das Symbol folgendermaßen dargestellt, um anzuzeigen, welche Suche derzeit im Gang ist:

- Lupe: wird immer dann angezeigt, wenn sich kein Suchttext in den Feldern Titel und Kunde befindet (Gegenteil vom Papierzettel) und die Checkbox nicht ausgewählt ist
- Lupe + Männchen: keine Texteingabe bei Filter, aber User will Aktivitäten sehen, wo er beteiligt oder involviert ist (Checkbox ausgewählt)
- Papierzettel + Lupe Symbol: Werte beim Filter (Titel, Kunde) wurden erfasst; dabei kann die Checkbox ausgewählt sein oder nicht (=> es wird kein eigenes Symbol für Papierzettel + Lupe + Männchen erstellt)

3.1.3.4 Aktivitätsübersicht

Die Aktivitäten Übersicht kommt man durch einen Klick auf eine Aktivität wie z. B. auf der vorher beschriebenen Seite. Hier werden alle wichtigen Information angezeigt, wie zum Beispiel Titel, Kunde, Status, Mitarbeiter usw.

Abbildung 14: Aktivitätsübersicht

3.1.3.5 Subaktivitäten

Abbildung 15: Subaktivitäten

Wenn man auf der Aktivitäten Übersicht mit Finger nach links wischt, kommt man zur Subaktivitäten Ansicht. Auf dieser Ansicht werden alle untergeordneten Aktivtäten aufgelistet und durch eine Farbe wird angezeigt, ob man selbst in diesem Projekt aktiv oder involviert ist.

- Grün für Aktiv
- Orange für Involviert
- Grau für Nicht Beteiligt

Mit einem Klick auf das Plus Symbol wird eine neue Subaktivität erstellt und in der Übersicht angezeigt. Dabei werden die übergeordnete Aktivität und der Kunde automatisch eingetragen.

3.1.3.6 Buchungen

Abbildung 16: Buchungen

Mit einem weiteren Wisch nach links kommt man zu den Buchungen, wo sich wiederum durch eine Checkbox einstellen lässt ob untergeordnete Buchungen angezeigt werden. Die Farbe der Buchungen ist abhängig davon, ob die zur Buchung gehörende Aktivität eine untergeordnete oder eine gleichrangige Aktivität ist. Bei jeder Buchung werden jeweils eine Zeile mit den aktiven Mitarbeitern und eine mit den involvierten Mitarbeitern angezeigt. Wenn dabei der aktuell angemeldeten Benutzer dabei ist, wird der Name in Blau ausgegeben.

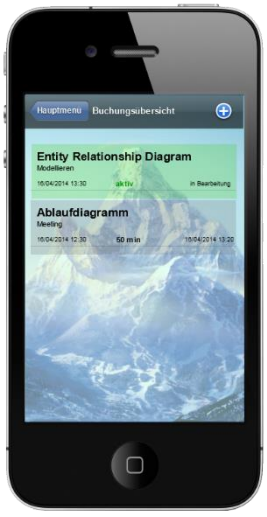
Durch das Plus Symbol wird die aktuelle Buchung gestoppt und eine neue mit aktueller Uhrzeit und Datum erstellt. Danach wird die Seite neu geladen und die neu erstellte Buchung ist somit auch sichtbar. Für genauere Informationen zur Erstellung einer neuen Buchung siehe 3.1.3.9 Buchung.

3.1.3.7 Kommentare

Auf dieser Kommentarseite werden die Einträge abhängig von der Sichtbarkeit in verschiedenen Farben angezeigt. Eigene Kommentare können für 5min nach der Erstellung noch bearbeitet und gelöscht werden.

Abbildung 17: Kommentare

3.1.3.8 Buchungenansicht



Auf diese Ansicht kommt man durch einen Klick auf den Menüpunkt „Zeitbuchungen“ im Hauptmenü. Hier werden alle Buchungen des aktuellen Benutzers angezeigt und die aktive Buchung grün und die abgeschlossenen Buchungen grau markiert.

Abbildung 18: Buchungenansicht

3.1.3.9 Buchung

Abbildung 19: Buchung

Durch einen Klick auf eine Buchung in der Buchungsübersicht kommt man zur Detailansicht, auf welcher Name, Start- und Enddatum/zeit, Kategorie und Buchungstext angezeigt werden und die Buchung gelöscht werden kann. Diese Eigenschaften kann man auch hier mit einem Klick darauf bearbeiten.

Falls diese Buchung gerade läuft und kein Enddatum eingetragen ist, wird ein Stopp-Button angezeigt, welcher bei einem Klick die aktuelle Zeit einträgt und den Benutzer auf die Buchungsübersicht zurückführt.

Bei der Erstellung einer Buchung wird normalerweise die aktuelle Zeit als Startzeit eingetragen, außer es wurde in den letzten 10min eine Buchung beendet, dann wird die Zeit direkt an die zuletzt beendete angeschlossen.

3.1.3.10 Einstellungen

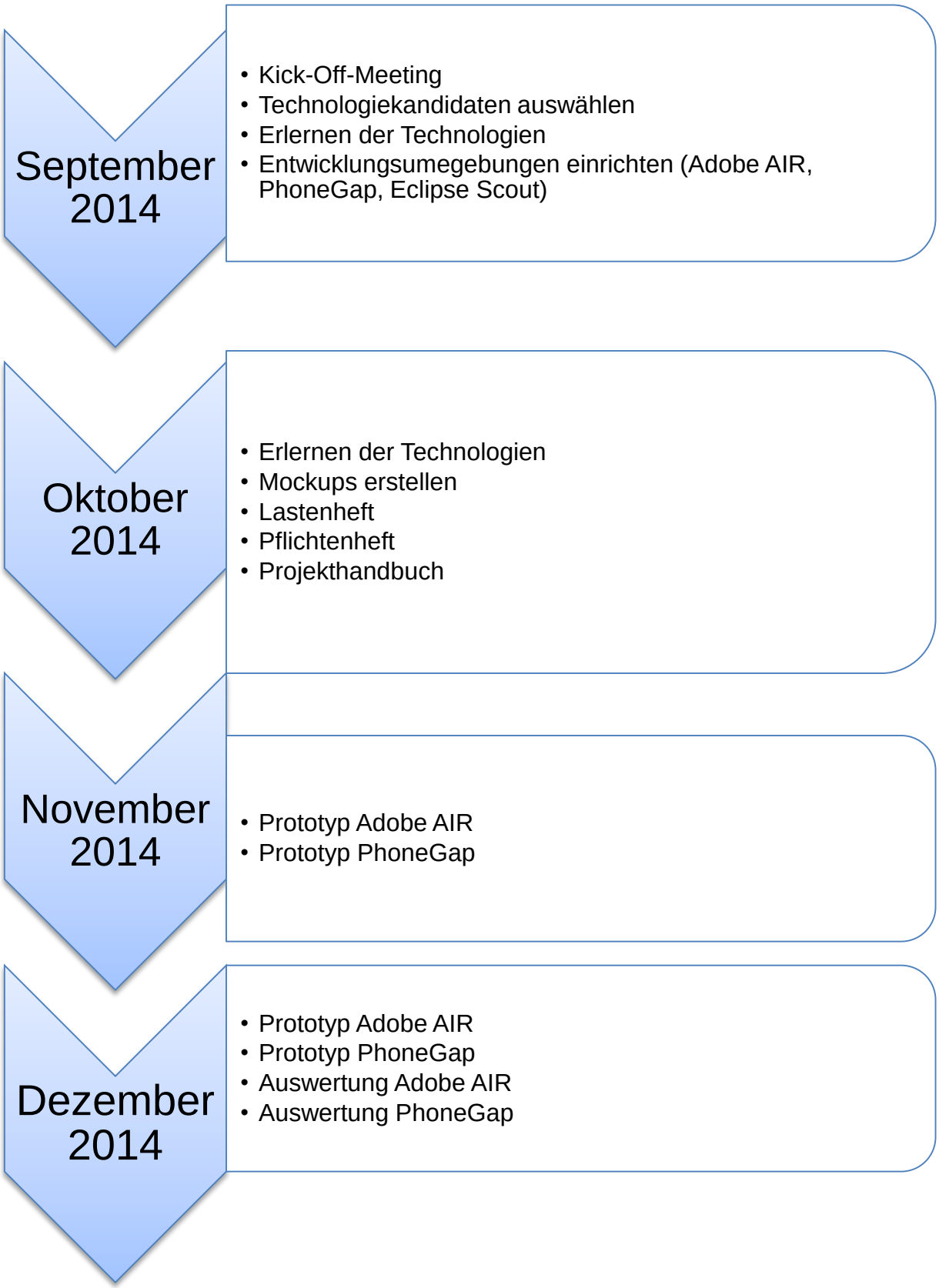
Abbildung 20: Einstellungen

In den Einstellungen kann der Benutzer aus einer vordefinierten Liste von Hintergrundbildern auswählen und er kann entscheiden ob untergeordnete Buchungen standardmäßig angezeigt werden oder nicht.

Eine weitere Einstellung ist ob sich der Benutzer jedes Mal anmelden muss oder der Login gespeichert wird (Standard).

3.2 Projektablaufplan

Der grobe Ablauf des Projekts lässt sich in folgende Schritte einteilen:



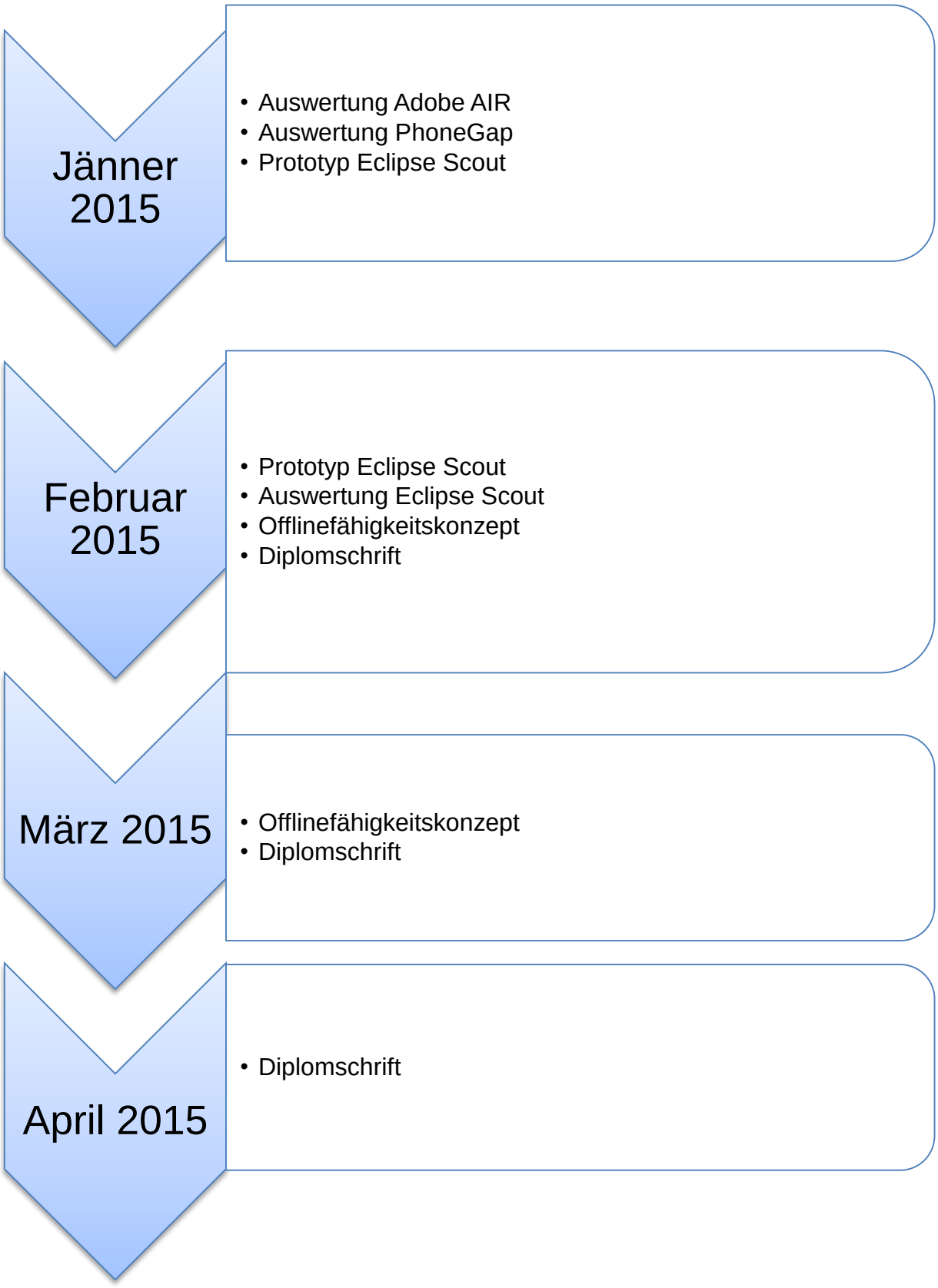


Abbildung 21: Projektablaufplan

3.3 Projektstrukturplan

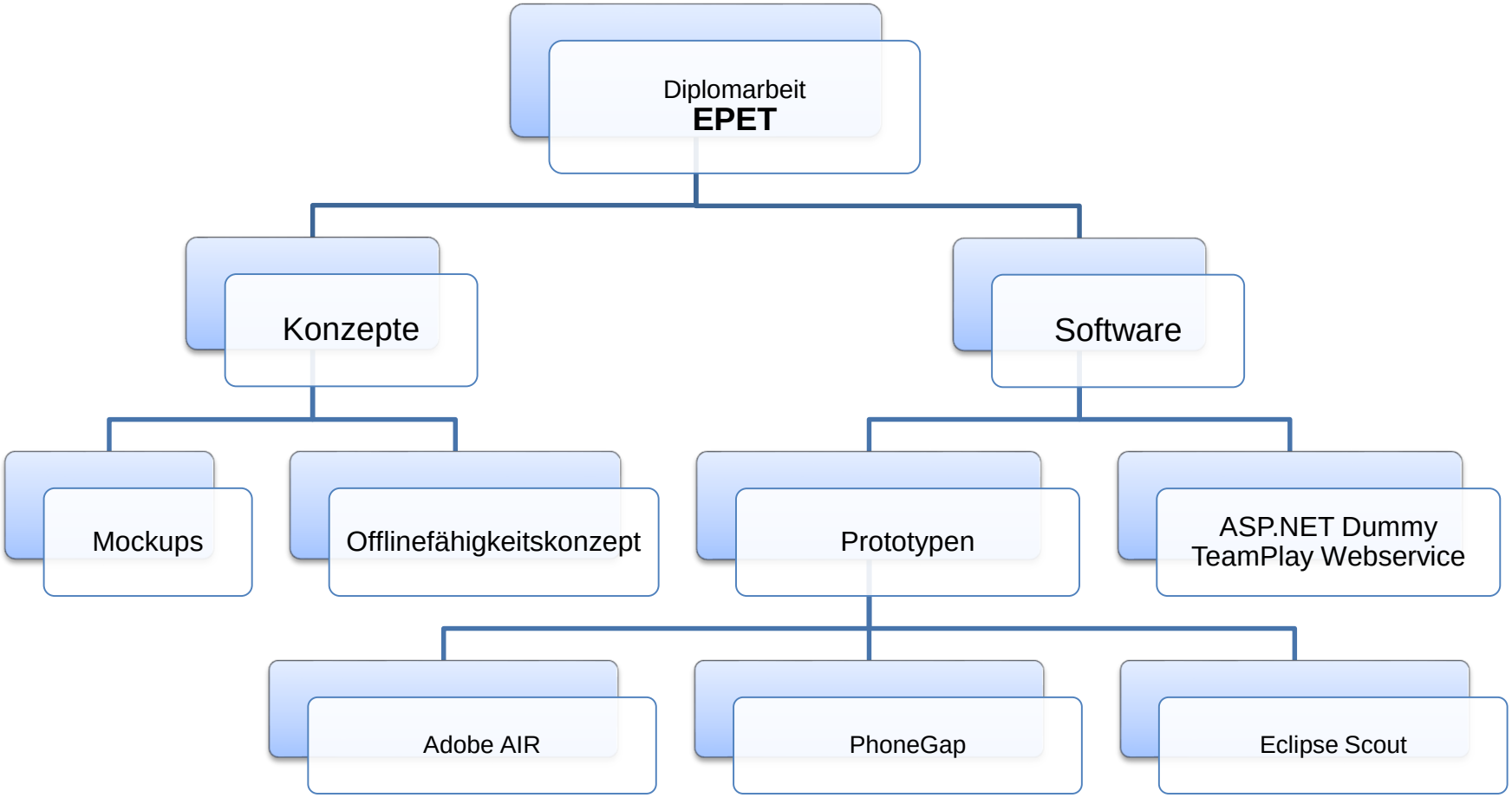


Abbildung 22: Projektstrukturplan

3.4 Steuernde Maßnahmen

3.4.1 Soll-Ablauf

Zur laufenden Koordination und Aufrechterhaltung des Arbeitsprozesses wird jeden Monat ein Meeting mit dem Auftraggeber vereinbart. Dort wird der Projektfortschritt kontrolliert und gemeinsam neue Ziele auf Grundlage des groben Zeitplans gesteckt. Wichtig ist dabei, dass sowohl für den Auftraggeber als auch für die Projektmitglieder die (bis zum nächsten Meeting) zu erledigenden Aufgaben schriftlich (elektronisch) vorliegen.

Weiters erkundigen sich die beiden Projektmitglieder wöchentlich gegenseitig über den Arbeitsfortschritt der intern aufgeteilten Aufgaben. Damit erhält das Team einen Überblick über den Gesamtfortschritt und kann wichtige Maßnahmen schneller einleiten sowie den Projektfluss besser vorhersehen. Eine weitere Folge ist, dass die Projektmitglieder stets an ihre Arbeit "erinnert" werden, was die Produktivität und gegenseitige Motivation erhöht.

3.4.2 Steuernde Maßnahmen bei Problemen

Das Einleiten von steuernden Maßnahmen erfolgt, dann wenn der vorliegende Ablauf nicht der Projektplanung entspricht. Dazu ist es notwendig, laufend (wöchentlich) die Projektüberwachung in Form des im Soll-Ablauf beschriebenen Treffens des Projektteams durchzuführen.

Steuernde Maßnahmen sind:

3.4.2.1 Verfehlung des Sachziels

Es ist stets zu beachten, dass im Vordergrund dieser Diplomarbeit nicht ein vollständig ausimplementiertes Produkt wichtig ist, sondern das Ergebnis der Forschung die Hauptrolle spielt. Essenziell ist somit die richtige Technik, um die über Handhabung, Vorteile, Nachteile, Möglichkeiten usw. eines Technologiekandidaten zu dokumentieren, sodass den Entwicklern der Auris IT Consult GmbH die Technologien leicht verständlich und aussagekräftig vorgelegt werden können.

3.4.2.2 Verfehlung des Terminziels

Da der Abgabezeitpunkt für die Diplomarbeit ein Fixtermin ist, muss dieser auch eingehalten werden und kann nicht abweichen. Wird jedoch erkannt, dass der noch abzuarbeitende Funktionsumfang nicht in der verbleibenden Zeit realisierbar ist, stehen folgende Maßnahmen zur Verfügung:

Gegenmaßnahme	Gewünschte Auswirkung	Ungewünschte Auswirkungen
Reduzierung des Funktionsumfang	Terminsituation entspannt sich	nicht ausreichende Ergebnisse für ein aussagekräftiges Forschungsergebnis
Mehrarbeit (intensivere und längere Arbeitsabschnitte)	Der Abgabetermin kann eingehalten werden	durch Stress und Zeitdruck verursachte Mängel am Ergebnis geistiger Zustand und positiv gesinnte Einstellung der Projektmitglieder verschlechtert
Verzicht/Reduzierung von projektfremden Aktivitäten	Terminsituation entspannt sich	fehlender Ausgleich zur Arbeit erhöhte Stresssituationen

Tabelle 3: Verfehlung des Terminziels

3.4.2.3 Verfehlung des Qualitätsziels

Obwohl die Beurteilung der technologischen Kandidaten im Vordergrund steht, ist es dennoch wichtig, die Prototypen in der geplanten Qualität (siehe Mockups) zu realisieren, da erstens eine Beurteilung einer Technologie effizient nur auf Grundlage eines Prototypen sinnvoll ist und zweitens festgestellt werden muss, ob bestimmte Funktionalitäten, die unbedingt unterstützt werden müssen, auch tatsächlich mit der jeweiligen Technologie umgesetzt werden können.

Gegenmaßnahme	Gewünschte Auswirkung	Ungewünschte Auswirkungen
Schulung der Projektmitglieder durch externe Experten	Qualität verbessert sich	hohe Schulungskosten Verzögerung des Projektfortschritts wegen zusätzlichem Zeitaufwand
Reduzierung der Anforderungen (wenn diese kein essenzieller Faktor sind)	neues Qualitätsziel wird erreicht; Projektfluss kann schnell fortgesetzt werden und wird nicht verzögert	Auftraggeber erhält weniger Qualität

Tabelle 4: Verfehlung des Qualitätsziels

3.5 Projektvorgehensweise

Das gesamte Projekt wurde mit der agilen Methode "**Scrum**" abgewickelt. Scrum wird meist zur Abwicklung von kleineren Projekten angewendet.

Die Besonderheit an Scrum ist, dass sich das Projektteam selbst organisiert und keine Hierarchien vorgegeben werden.

(zum Folgenden vgl. Dreyer 2008: 2, 4, 5) [6]

> Die Projektabwicklung erfolgt dabei in regelmäßigen kleineren Abschnitten (Sprints). Alle zu erledigende Aufgaben bzw. Produktbestandteile sind im Product Backlog abgelegt. Für einen Sprint werden nun also einzelne Aufgaben aus dem Product Backlog herausgenommen, welche in diesem Sprint erledigt werden müssen. Nach Abwicklung eines Sprints (Dauer: ca. 4 Wochen) folgt eine Besprechung mit dem Auftraggeber. Hierbei wird das Sprint-Ergebnis besprochen und präsentiert und etwaige Verbesserungsvorschläge werden ermittelt. Ebenfalls wird der Plan für den nächsten Sprint anhand des Product Backlogs sowie des Zeitplans aufgestellt. Daraufhin beginnt der nächste Sprint.

Die Vorgehensweise Scrum wurde festgelegt, da sie hohe Flexibilität bietet. Das ist bei dieser Diplomarbeit von Vorteil, weil sich bei der Auswahl und Umsetzung der plattformübergreifenden Entwicklungstechnologien sowie bei den Prototypen selbst laufend Änderungen ergeben können.

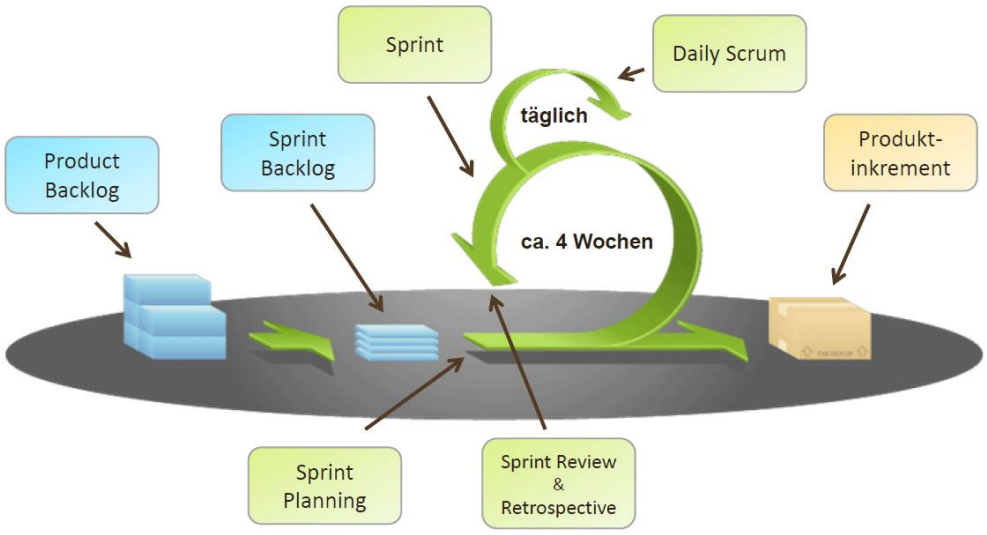


Abbildung 23: Scrum-Prozess [7]

3.6 Ressourcenplan

3.6.1 Personalressourcen

Ressource	Rolle
Paul Schmutz	Projektleiter: + Kommunikation mit Auftraggeber + Projektfluss kontrollieren + Programmierung
Paul Lindenberger	Chef-Programmierer: + Entwurf & Design + Hauptprogrammierung + Testen

Tabelle 5: Personalressourcen

3.6.2 Sach- und Anlagenressourcen

3.6.2.1 Projektentwicklung

- Laptops

  Die Entwicklung der Prototypen wurde auf den privaten Laptops der beiden Projektmitglieder vorgenommen.

3.6.2.2 Projekttestung

- iPhone
  Prototypen für iOS konnten mit einem iPhone getestet werden.
- Android-Phone
  Android-Prototypen wurden mit einem Android-Smartphone auf ihre Funktionalität kontrolliert.
- PC/Laptop (Windows 8 und Mac OS VM)
  Beide Projektmitglieder besitzen Windows 8 sowie eine virtuelle Maschine für Mac OS.
- Windows Phone (WP8)
  Der Ansprechpartner der Auris IT Consult GmbH konnte bei Meetings Prototypen mit einem Windows Phone prüfen.

3.7 Aufwandsschätzung

Kurz nach dem Projektstart wurde der Projektaufwand geschätzt und die jeweiligen Teilbereiche mit einer Stundenanzahl versehen. Die Aufwandsschätzung ergibt:

Modul	**Stunden**
Mockups	20
Pflichtenheft & Projekthandbuch	15
Mock-Webservice	50
Prototyp Adobe AIR	90
Auswertung Adobe AIR	10
Prototyp PhoneGap	70
Auswertung PhoneGap	10
Prototyp Eclipse Scout	80
Auswertung Eclipse Scout	10
Diplomschrift	100
Summe	**455**

Tabelle 6: Aufwandsschätzung

4 Vorlage-Produkt

Wie bereits erwähnt werden die Prototypen anhand des Zeiterfassungssystems „TeamPlay“ von der Auris IT Consult GmbH entwickelt. In diesem Kapitel wird ein Überblick über die Softwarelösung gegeben.

Der Inhalt und die Abbildungen wurden dabei von der TeamPlay-Hompage entnommen [8].

4.1 Zeiterfassung

Die Verbindung von Arbeitszeiterfassung und Aktivitätenverwaltung gibt einen Überblick über die Auslastung von Mitarbeitern. Die Stundenerfassung und Zuordnung zu Projekten schafft eine Abrechnungsgrundlage des Unternehmens zu Kunden und Mitarbeitern.

- Aktivitätsbasierte Zeiterfassung
- Prioritätslisten definierbar
- Automatische Benachrichtigungen
- Alle Aktivitäten auf einen Blick

4.2 Projekttracking

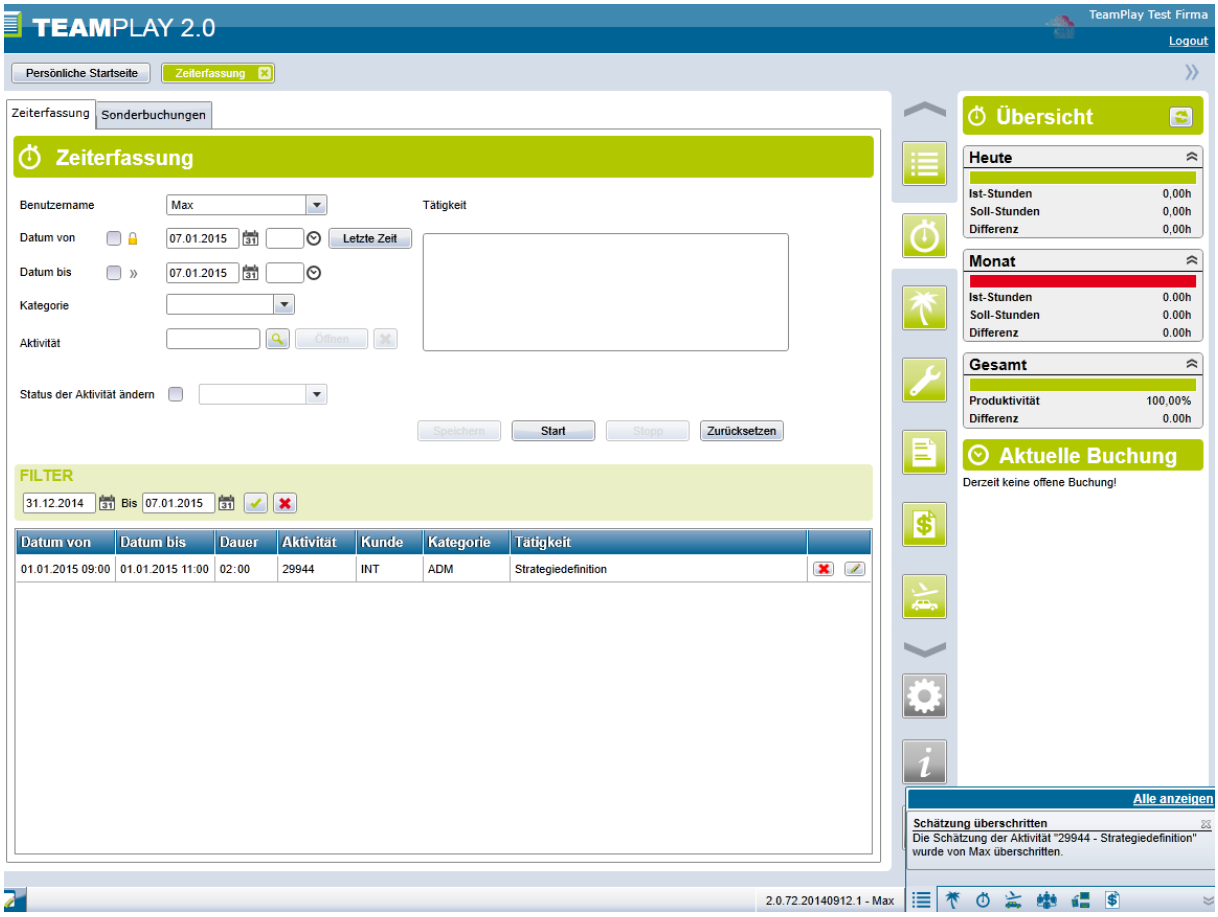


Abbildung 24: Projekttracking

Projekte lassen sich in Unterprojekte und Aktivitäten unterteilen. Die Festlegung von Budgets ermöglicht die volle Kostenkontrolle. Vom Erstkontakt bis zur Endabnahme werden alle Arbeitszeiten erfasst und an die Verrechnung weitergegeben.

- Volle Budgetkontrolle
- Verantwortlichkeiten definierbar
- Transparente Projektverrechnung
- Projektcontrolling

4.3 Urlaubsverwaltung

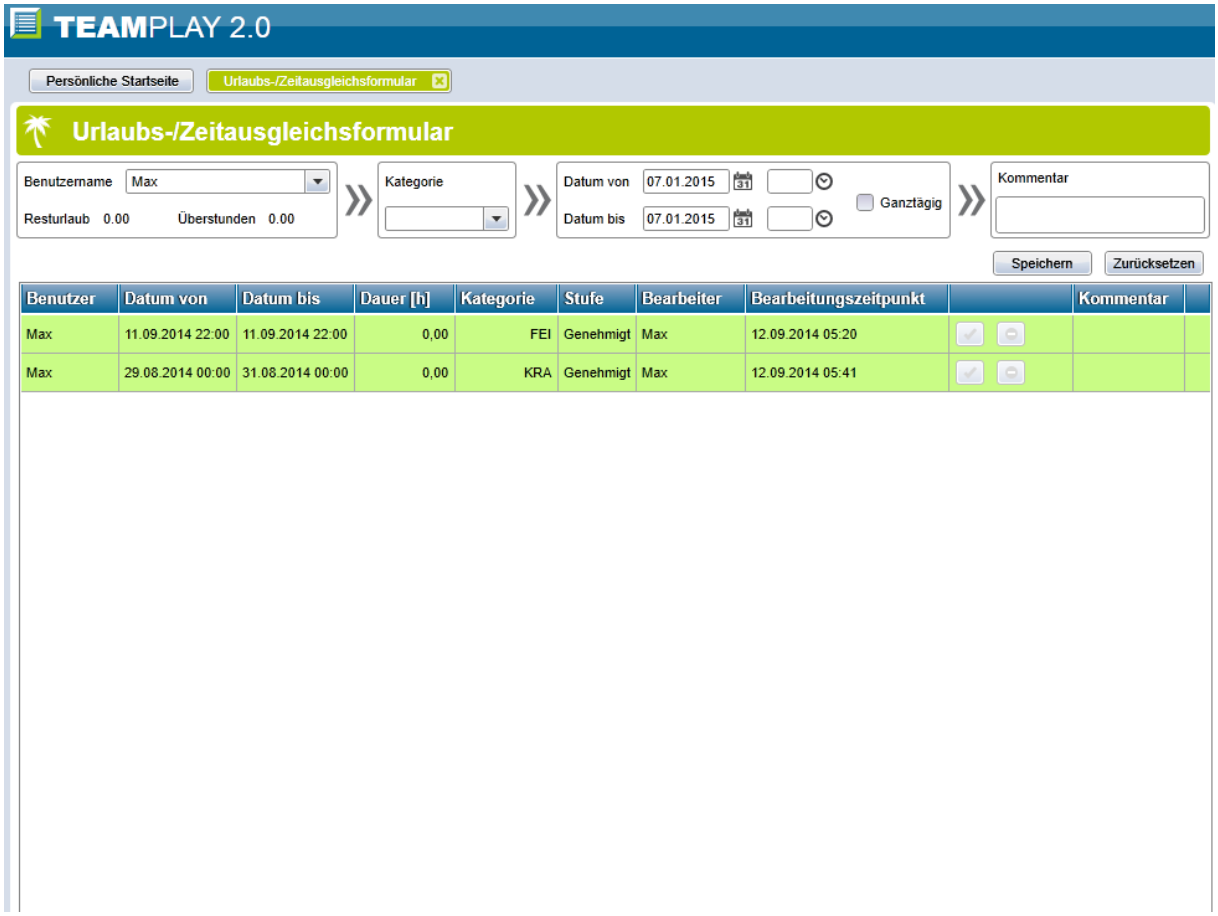


Abbildung 25: Urlaubsverwaltung

TeamPlay integriert die Administration von Urlaubs- und Zeitausgleichsanfragen. Anfragen werden direkt an Stellvertreter oder Vorgesetzte weitergeleitet. Genehmigungen werden durch einen Auslastungsüberblick vereinfacht.

- Urlaubsanträge stellen
- Zeitausgleichsanträge
- Nationale Feiertage
- Abwesenheitsassistenz

4.4 Reporting

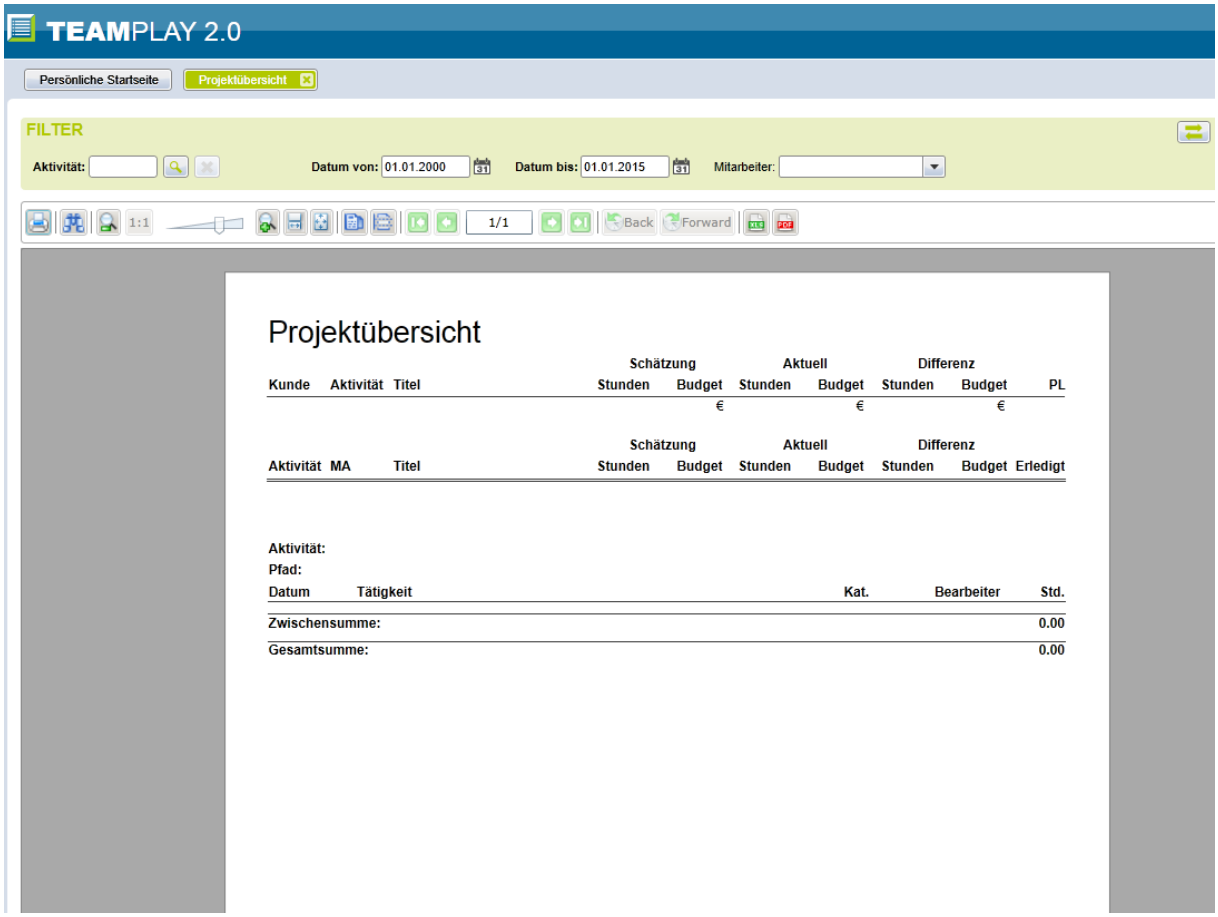


Abbildung 26: Reporting

Durch Auswertungen in Echtzeit behalten Sie jederzeit den Überblick und treffen die richtigen Entscheidungen. Ihre Auswertungen können Sie zur genaueren Analyse in verschiedene Formate exportieren.

- Soll-Ist Vergleiche
- Profitabilitätsvergleich
- Arbeitszeitauswertung
- Export von Reports

4.5 Dienstreisen

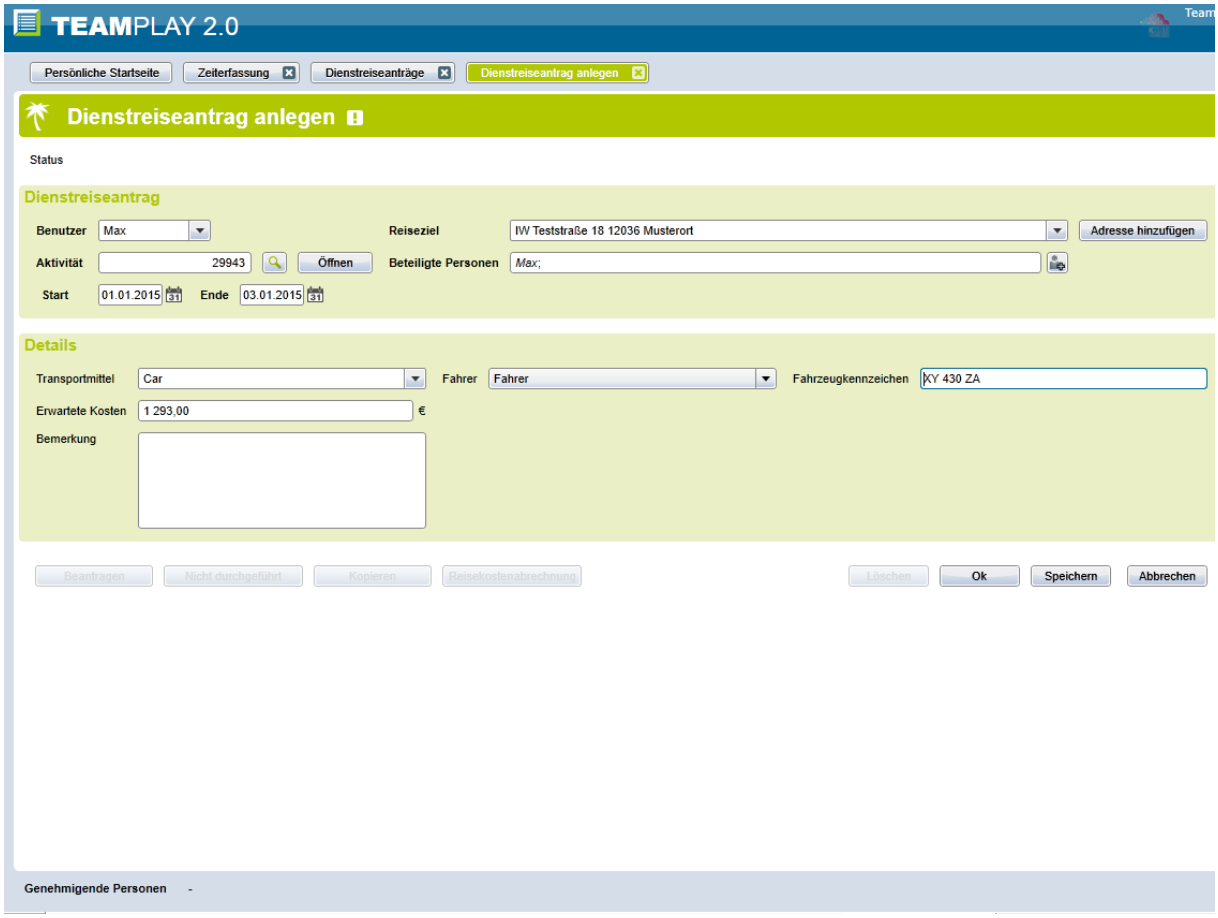


Abbildung 27: Dienstreisen

TeamPlay hilft bei der Organisation von Dienstreisen. Die Funktionalität reicht von der Antragstellung, der mobilen Erfassung von Belegen bis zur Verrechnung der Dienstreise. Dienstreisen können mit einem Klick von Vorgesetzten genehmigt werden.

- Erfassung von Dienstreisen
- Verrechnung von Dienstreisen
- Integrierte Belegerfassung

4.6 Verrechnung

Die Arbeitszeiterfassung erfolgt in TeamPlay über Zeitbuchungen auf die erfassten Projekte und Unterprojekte. Die mit TeamPlay erstellten Rechnungen oder Zeitbuchungen können in ein Buchhaltungsprogramm exportiert und sofort verbucht werden.

- Individuelle Stundensätze
- Abrechnung von Pauschalen
- Eigene Rechnungslayouts
- Automatische wiederkehrende Rechnungen

5 Realisierung und Umsetzung

5.1 Technologien

In diesem Kapitel werden die zur Erstellung der Softwarelösung der Diplomarbeit verwendeten Entwicklungstechnologien aufgelistet.

Vorweg ist zu beachten, dass die umgesetzten Prototypen nicht dem im Kapitel Planung vorgesehenen Konzept entsprechen. Der Grund liegt darin, dass ein Technologiekandidat die vom Auftraggeber geforderten Kriterien nicht erfüllte.

Überblicksmäßig werden hier die geplanten und die tatsächlich verwendeten Technologien dargestellt:

Geplante Technologien	**Tatsächlich eingesetzte Technologien**
Backend	Backend
ASP.NET	**ASP.NET**
Frontend	Frontend
Adobe AIR	**Adobe AIR**
PhoneGap	**PhoneGap**
Eclipse Scout	**Eclipse Scout** (frühzeitig abgebrochen, erfüllt Kriterien nicht)
	Embarcadero (Ersatz für Eclipse Scout; Basiskriterien grundsätzlich erfüllt, jedoch durch aufwändige Entwicklungsweise eingestellt)

Tabelle 7: Umgesetzte Technologiekandidaten

Realisierte Technologien

Nicht realisiert bzw. abgebrochen

5.1.1 Adobe AIR

5.1.1.1 Einleitung

Adobe AIR ist eine Laufzeit, die es Entwicklern ermöglicht, Applikationen mit dem gleichen Code für mehrere Betriebssysteme einzubinden. Der Entwicklungsvorgang deckt sich dabei zu einem großen Teil mit der Flash-Entwicklung.

5.1.1.2 Entwicklungsumgebung

Die bevorzugte Entwicklungsumgebung für Adobe AIR Apps ist **Adobe Flash Builder**. Wie in den meisten Fällen existieren kostenlose Alternativen oder Plug-Ins, die für eine effektive, reibungslose und uneingeschränkte App-Entwicklung jedoch meist nicht anwendbar sind. Gründe dafür sind unausgereifte Entwicklungsumgebungen, fehlende oder umständliche Build-Tools (z. B. Build for Android), fehlende Auto-Vervollständigung (Intellisense), sowie abweichende Projektstrukturen, die nicht den Adobe-Standards entsprechen. Weiters bezieht sich die gesamte im Internet befindliche Dokumentation fast ausschließlich auf Adobe Flash Builder. Folglich wurde für die Entwicklung des Prototyps ebenfalls Flash Builder verwendet.

Adobe Flash Builder 4.7
Betriebssysteme: Mac, Windows **Kurzbeschreibung:** eine von Adobe entworfene Umgebung zur Entwicklung von Flash-Anwendungen, die Eclipse als Basis verwendet

Abbildung 28: Adobe Flash Builder [9]

5.1.1.3 Zielplattformen

Mit Adobe AIR können Apps für

- Android
- iPhone
- Windows
- Mac

entwickelt werden.

5.1.1.4 Testen

Im Adobe Flash Builder können Adobe AIR Apps in kurzer Zeit getestet werden. Dabei kann die App auf zwei Arten ausgeführt werden:

- Integrierter Adobe AIR Emulator
- USB-Debug Android/iPhone

Beim Emulator besteht zusätzlich auch die Möglichkeit zur Auswahl des Zielbetriebssystems (Android, iOS) sowie eines konkreten Gerättyps (verschiedene Bildschirmauflösungen) und erlaubt somit eine vielseitige Testung der Applikationen.

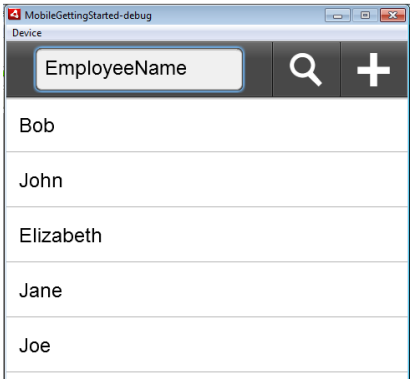


Abbildung 29: Beispielapplikation im Adobe AIR Emulator [10]

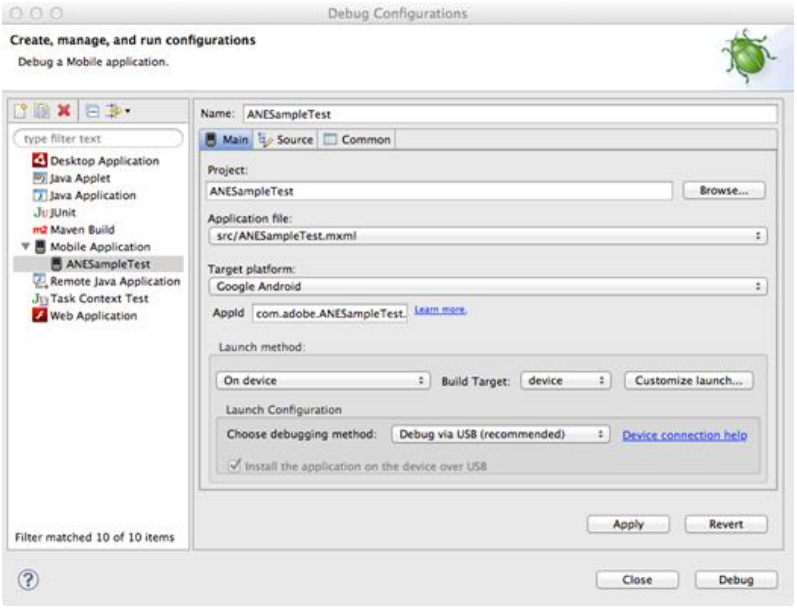


Abbildung 30: Debug-Menü im Adobe Flash Builder mit Konfiguration zum USB-Debugging auf einem Android-Smartphone [11]

5.1.1.5 Mindestanforderungen Adobe AIR für Clients

Da die Adobe AIR Runtime nicht auf allen Geräten lauffähig ist, werden im Folgenden die unterstützten Betriebssysteme mit den jeweiligen Mindestanforderungen aufgelistet.

Windows	
Prozessor:	x86-kompatibler Prozessor mit 2.33 GHz oder Intel Atom™ mit mindestens 1.6 GHz für Geräte der Netbook-Klasse
Betriebssystem:	Microsoft® Windows® XP, Windows Server 2008, Windows Vista® Home Premium, Business, Ultimate oder Enterprise (auch 64 Bit) mit Service Pack 2, Windows 7 oder Windows 8 Classic
RAM:	512 MB (1 GB empfohlen)
Mac OS	
Prozessor:	Intel® Core™ Duo mit 1,83 GHz oder schneller
Betriebssystem:	Mac OS X 10.6, 10.7, 10.8 und 10.9
RAM:	512 MB (1 GB empfohlen)
Android	
Prozessor:	ARMv7-Prozessor x86-Prozessor mit VFPU mind. 550 MHz, OpenGL ES 2.0, H.264- und AAC-Hardware-Decoder
Betriebssystem:	Android™ 2.3 und höher
RAM:	256 MB

iOS	
Geräte:	iPod touch 4, iPhone 4, iPhone 5, iPad 2, iPad mini oder iPad mit Retina-Display
Betriebssystem:	iOS 6.1 und höher

Tabelle 8: Anforderungen Adobe AIR Clients [12]

5.1.1.6 Mindestanforderungen für Entwicklung (Flash Builder)

Windows	
Prozessor:	2 GHz und schneller
RAM:	2 GB
Disk Space:	5 GB
Java:	JRE 1.6 oder 1.7 (32 oder 64 Bit)
Mac OS	
Prozessor:	Intel® Prozessor
RAM:	2 GB
Disk Space:	4 GB
Java:	JRE 1.6 (64 Bit)

Tabelle 9: Anforderungen Adobe AIR Entwicklung [13]

5.1.2 PhoneGap

5.1.2.1 Einleitung

PhoneGap ist ein Framework zur Erstellung hybrider Applikationen für mobile Endgeräte. Dieses Framework ermöglicht es, Anwendungssoftware für mobile Endgeräte mit JavaScript, HTML5 und CSS 3 zu schreiben, anstelle von gerätespezifischen Programmiersprachen wie Objective-C oder Java. Da gefordert war, dass JavaScript Programmierung soweit wie möglich vermieden werden soll, wurde das **GWT Framework** verwendet, welches aus Java Code HTML, CSS und JavaScript generiert.

Die dabei entstehenden Applikationen sind hybride Applikationen, das heißt es gibt zwar keine native UI, dafür kann aber auf Programmierschnittstellen der Betriebssysteme der Endgeräte zugegriffen werden.

5.1.2.2 Entwicklungsumgebung

Die geeignetste Entwicklungsumgebung für GWT Apps ist die kostenlose Entwicklungsumgebung **Eclipse**. Es existieren natürlich auch andere Alternativen, aber diese sind die bekanntesten:

- Eclipse
- NetBeans
- IntelliJ IDEA

Eclipse 4.4
Betriebssysteme: plattformunabhängig **Kurzbeschreibung:** ein quelloffenes Programmierwerkzeug, das von der Eclipse Foundation weiterentwickelt wird

Abbildung 31: Eclipse [14]

5.1.2.3 Zielplattformen

Mit PhoneGap können die Apps auf

- Android
- iOS
- Blackberry
- Windows Phone

- Windows
- Mac
- Linux

verwendet werden (nur die bekanntesten wurden hier aufgezählt; PhoneGap unterstützt noch weitere Plattformen).

5.1.2.4 Testen

Im Eclipse mit dem Google Plugin können die erstellten Apps im Browser mit dem Google Web Toolkit Developer Plugin getestet werden. Dabei gibt es 2 Möglichkeiten:

- Debugging mithilfe des Browser Plugins
  - Die App wird direkt von Eclipse aus gestartet und öffnet sich dann automatisch im Standardbrowser. Der Browser kann mithilfe des Plugins mit der Entwickler Shell über TCP/IP kommunizieren. Dies bietet einige Vorteile wie Hot Code Replacement oder die üblichen Debugging Möglichkeiten wie Break Points und Ansehen der Werte aller Variablen.
- GWT Compiler
  - Die in Java programmierte App wird vollständig in HTML, CSS und JavaScript kompiliert und kann in jedem Browser ohne der Notwendigkeit eines Plugins ausgeführt werden.

Die App kann nach dem Kompiliervorgang mit PhoneGap verpackt und dann im Emulator oder am echten Gerät getestet werden.

5.1.2.5 Mindestanforderungen für Entwicklung

Zum Entwickeln von GWT Apps werden die folgenden Mindestanforderungen benötigt:

Windows, Mac OS, Linux	
Betriebssysteme:	Windows Vista/XP/2000, Mac OS X 10.4+ (Tiger oder Leopard), oder Linux mit GTK+ 2.2.1+
RAM:	512 MB
Disk Space:	100 MB
Java:	Oracle Java 2 Runtime Environment 1.5

Tabelle 10: Anforderungen PhoneGap [15]

5.1.3 Eclipse Scout

5.1.3.1 Einleitung

Eclipse Scout ist ein Open Source Framework zur Entwicklung von Businessanwendungen. Mit Scout wird eine klassische Client-Server-Architektur unter Verwendung der Programmiersprache Java abgebildet. Gleichzeitig werden mit Eclipse Scout mehrere Client-Plattformen abgedeckt. Clientseitig werden Swing, SWT sowie Browser (Desktop, Mobile, Tablet) unterstützt.

5.1.3.2 Entwicklungsumgebung

Gleichzeitig mit Scout muss auch die Entwicklungsumgebung Eclipse verwendet werden. Eclipse Scout wird als Paket mit Eclipse gemeinsam bereitgestellt. Mit Eclipse Scout wird daher auch die Entwicklungsumgebung gleichnamig benannt.

Eclipse Scout
Betriebssysteme: plattformunabhängig (Windows, Mac, Linux) **Kurzbeschreibung:** ein quelloffenes Programmierwerkzeug, das von der Eclipse Foundation weiterentwickelt wird

Abbildung 32: Eclipse Scout [16]

5.1.3.3 Zielplattformen

Eclipse Scout Client Apps sind auf folgenden Plattformen lauffähig:

- Großteil der Smartphone- und Desktop Betriebssysteme (=>HTML-Seite im Browser)
- Windows (Swing, SWT)
- Mac (Swing, SWT)
- Linux (Swing, SWT)

(Swing und SWT sind auf weiteren Plattformen lauffähig, hier wurden jedoch die relevantesten aufgezählt)

5.1.3.4 Testen

Mit den in Eclipse integrierten Plugins für Scout werden verschiedene Möglichkeiten geboten, die Applikation zu auszutesten und zu debuggen. Grundlage ist das Starten des

Scout-Services, der im Hintergrund gestartet wird und als Backend für die verschiedenen Clients dient. Die verschiedenen Clients, die ausgeführt werden können, sind

- Desktop
  - Swing
  - SWT
- Web, Mobile, Tablet
  - RAP Widget Toolkit

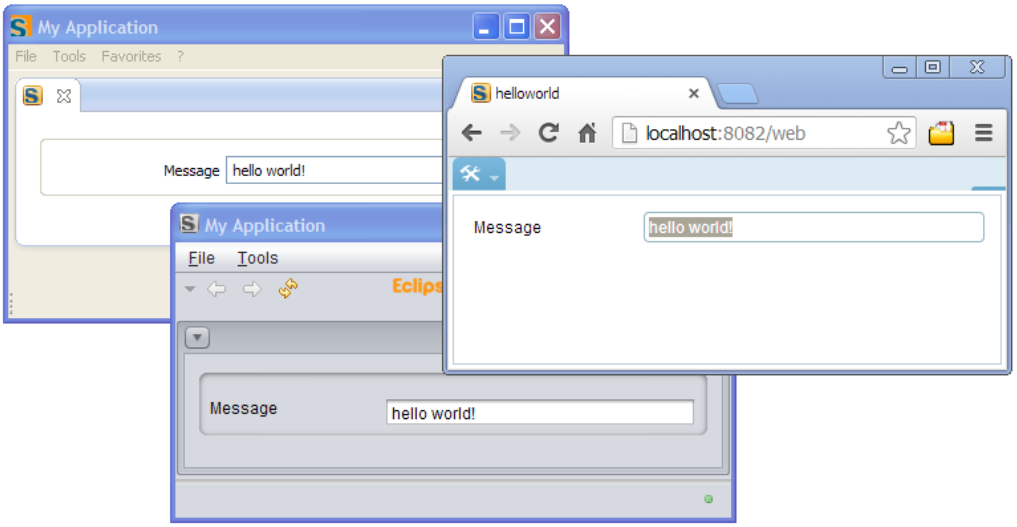


Abbildung 33: Eclipse Scout Applikation ausgeführt auf verschiedenen Clients [17]

5.1.3.5 Mindestanforderungen für Entwicklung (Eclipse)

Windows, Mac OS, Linux	
Java:	JRE 1.7 (32 oder 64 Bit) oder höher
Disk Space:	ca. 1 GB

Tabelle 11: Anforderungen Eclipse Scout Entwicklung [18]

5.1.4 Embarcadero

5.1.4.1 Einleitung

Embarcadero Technologies, Inc. ist ein amerikanisches Unternehmen mit Hauptsitz in San Francisco. Neben der Technologie für plattformübergreifende Entwicklung bietet das Unternehmen verschiedene andere Produkte an, z. B. Tools zur Datenbankentwicklung, Verwaltung und Leistungsoptimierung oder Modellierungswerkzeuge.

Die plattformübergreifende Entwicklungstechnologie von Embarcadero deckt die Zielplattformen Android, iOS, Windows und Mac ab. Die Besonderheit dabei ist, dass hier keine eigene Laufzeit verwendet wird oder Apps auf jedem Gerät gleich aussehen, sondern Embarcadero wandelt die entwickelten Apps in eine native App für die jeweilige Plattform um, wodurch die Performance hoch gehalten wird und die nativen GUI-Elemente verwendet werden.

5.1.4.2 Entwicklungsumgebung

Die Entwicklungsumgebung zur Erstellung der nativen Apps von einer Codebasis ausgehend, heißt RAD Studio. Für die Entwicklung können die Programmiersprachen C++ oder Delphi verwendet werden. Da die Technologie vom Hersteller Embarcadero Technologies, Inc. abhängig ist, besteht keine Möglichkeit, eine andere Entwicklungsumgebung zu verwenden.

RAD Studio XE7/8
Betriebssysteme: Windows **Kurzbeschreibung:** „Softwareentwicklungslösung für das schnelle Erstellen, Programmieren und Erweitern ‚vernetzter‘ Anwendungen für Windows, Mac, iOS, Android und das Internet of Things.“ (Embarcadero Technologies Inc. o. J.) [19]

Abbildung 34: RAD Studio

5.1.4.3 Zielplattformen

Ähnlich wie Adobe AIR zielt Embarcadero auf die Plattformen

- Android
- iPhone
- Windows
- Mac

ab.

5.1.4.4 Testen

RAD Studio bietet verschiedene Möglichkeiten, Applikationen zu testen. Einerseits können die Desktopsysteme auf der Plattform des Entwicklerrechners realitätsgetreu getestet werden und die mobilen System entweder auf einem Emulator (z. B. der Android-Emulator) oder über USB-Debugging mit einem echten Gerät. RAD Studio erkennt, ob ein Gerät angeschlossen ist bzw. ob ein Emulator läuft und anschließend kann die jeweilige Plattform zum Debugging verwendet werden.

Übersicht:

- Android (echtes Gerät oder Android-Emulator)
- iOS (echtes Gerät oder iOS-Simulator)
- Windows (auf Windows-Entwickler-PC)
- Mac (auf Mac-Entwickler-PC)

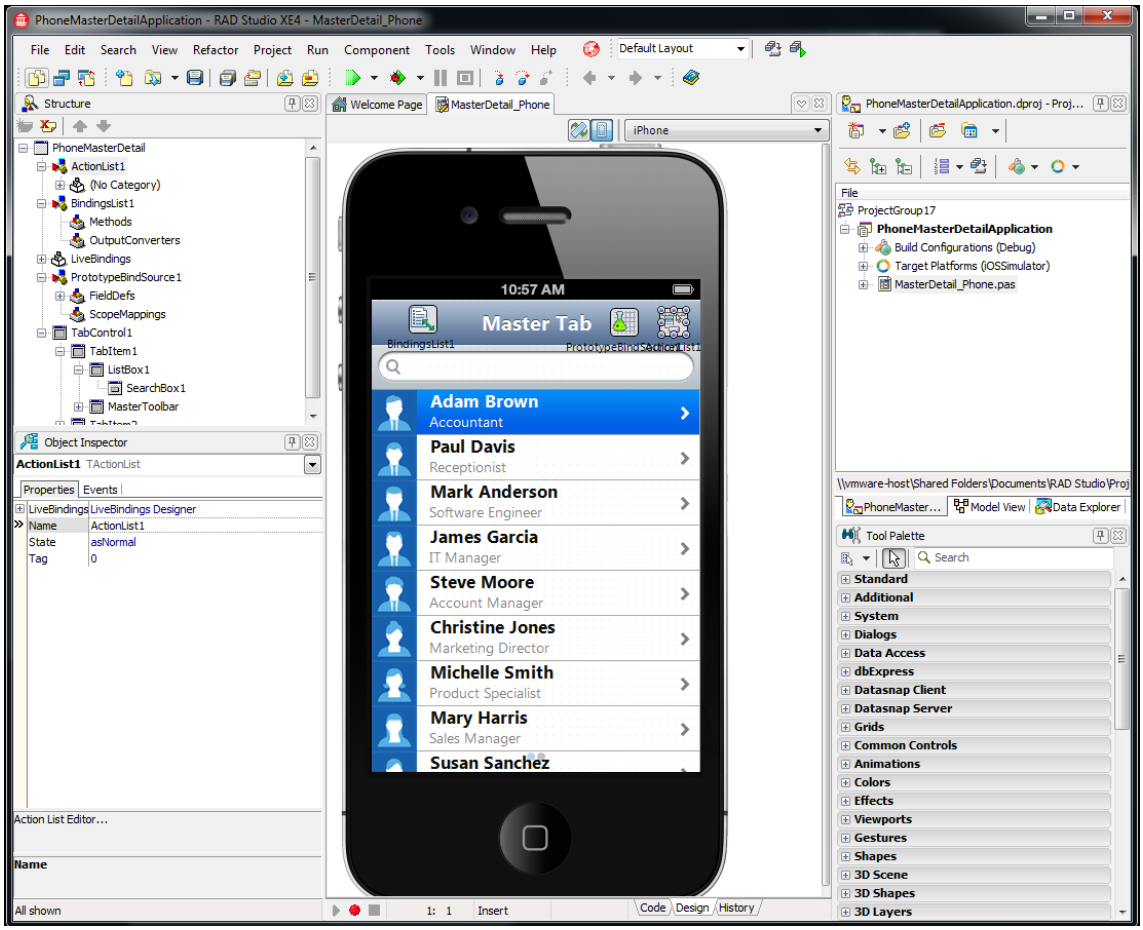


Abbildung 35: Screenshot RAD Studio XE4 mit Designer [20]

5.1.4.5 Mindestanforderungen (RAD Studio)

Windows	
Prozessor:	mindestens 1.4 GHz (2 GHz+ empfohlen)
RAM:	1 GB (2 GB empfohlen)
Disk Space:	4 GB
Java:	JRE (am besten aktuellste Version)

Tabelle 12: Anforderungen Embarcadero Entwicklung [21]

5.1.5 ASP.NET

5.1.5.1 Einleitung

Das ASP.NET Framework ist das von Microsoft entwickelte Werkzeug zur Entwicklung von dynamischen Webapplikationen.

> ASP.NET kommt auf ca. 17,2 % aller Websites als serverseitige Programmiersprache zum Einsatz und liegt damit nach PHP (82 %) und vor dem drittplatzierten Java (2,8 %) auf dem zweiten Platz der am häufigsten serverseitig verwendeten Sprachen zum Erstellen von Webseiten. (Anonymus 2015) [22]

Produkte, die auf ASP.NET aufbauen sind:

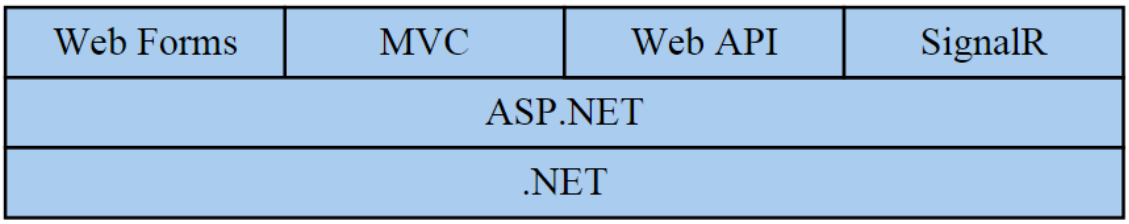


Abbildung 36: ASP.NET Produkte [23]

Die für die Diplomarbeit verwendeten Technologien sind MVC (für Login) und Web API (für die REST-Calls).

ASP.NET MVC sowie das ASP.NET Web API verwenden sogenannte „Controller“, die Funktionen bereitstellen, welche für Clients via HTTP aufrufbar sind. Controller werden jeweils durch eine eigene Klasse repräsentiert und stellen Implementierungen für die einzelnen HTTP-Methoden bereit (POST, PUT, GET, DELETE).

5.1.5.2 Entwicklungsumgebung

Als Entwicklungsumgebung wird Visual Studio 2013 Ultimate verwendet, da dies die tauglichste Umgebung für .NET-Applikationen ist.

Visual Studio 2013 Ultimate
Betriebssysteme: Windows **Kurzbeschreibung:** eine von Microsoft angebotene Entwicklungsumgebung zur Programmierung mit unterschiedlichen Hochsprachen

Abbildung 37: Visual Studio [24]

5.1.5.3 Testen

Durch Verwendung von Visual Studio bieten sich verschiedene Möglichkeiten zur Ausführung von Webapplikationen.

Es gibt 3 sinnvolle Möglichkeiten, Webapplikationen zu testen und diese mit verschiedenen Clients anzusteuern:

- Localhost
    - Die Webapplikation kann lokal ausgeführt und in einem Browser (z. B. Chrome, Firefox, Internet Explorer) geöffnet werden. Dabei ist die Applikation von außen (lokales Netzwerk) nicht zugänglich. Sie kann nicht mit separaten Clients getestet werden sondern nur mit dem Entwicklungsrechner selbst.
- IIS
    - Eine weitere Möglichkeit wäre, die Applikation auf einem lokalen IIS-Server zu hosten. Damit ist die Webapplikation auch für im lokalen Netz befindliche Clients zugänglich (z. B. Testen mit Smartphone im lokalen Netzwerk).
- Windows Azure
    - Neben lokaler Ausführung kann die Applikation auch in der Windows Azure Cloud veröffentlicht werden, wodurch sie weltweit über das Internet zugänglich ist.

5.2 Systemarchitektur

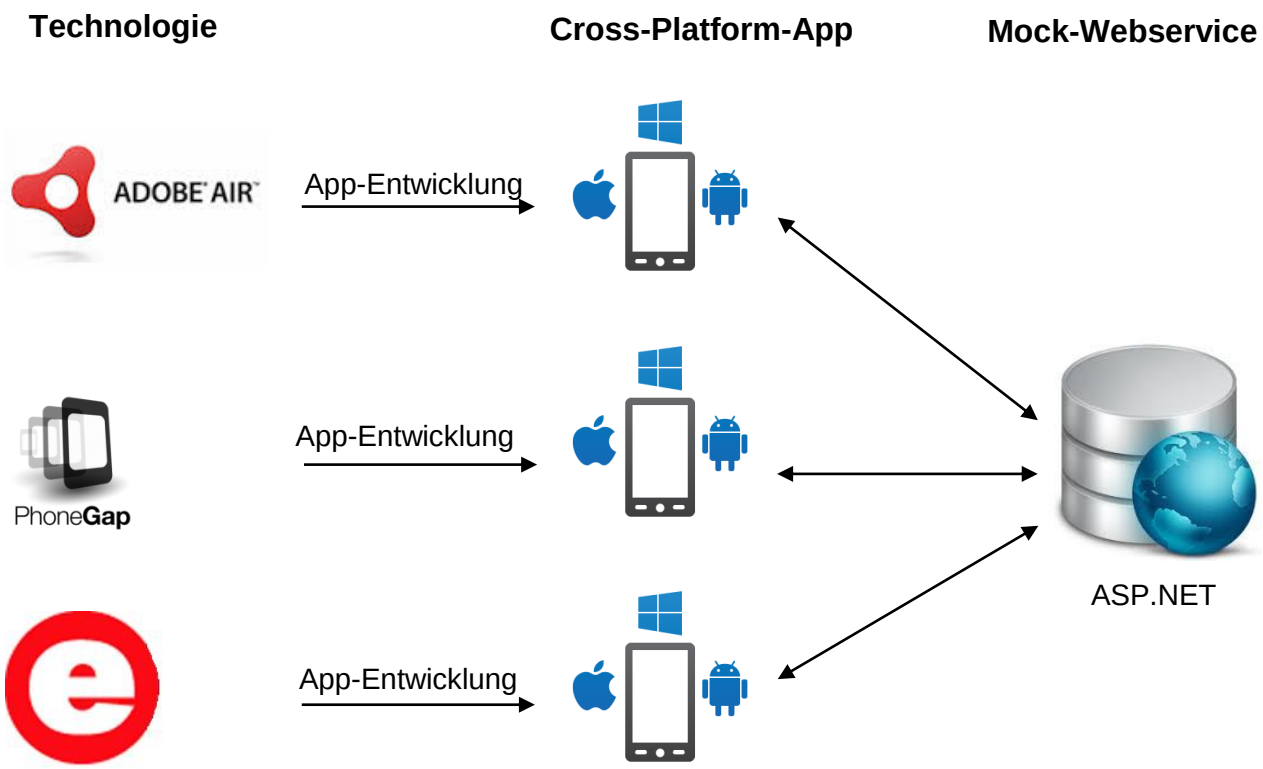


Abbildung 38: Systemarchitektur

Wie in der Grafik zu entnehmen ist, wurden insgesamt drei Prototypen mit den Technologien Adobe AIR, PhoneGap und zum Teil Embarcadero angefertigt. Diese sind dabei nach Priorität geordnet, auch die Entwicklung wurde in der dargestellten Reihenfolge abgewickelt. Die resultierenden Cross-Platform-App mussten im Allgemeinen das Aussehen der geplanten Mockups erfüllen. Neben der grafischen Benutzeroberfläche mussten auch HTTP-Requests eingebaut werden, da TeamPlay auch in der Praxis über ein zentrales Backend arbeitet. Für den Entwicklungsprozess wird jedoch vorerst ein Mock-Webservice verwendet, der lediglich Testdaten zur Verfügung stellt.

5.3 Schnittstellen

Zwischen dem Client-Programm und dem (Mock-)Webservice besteht eine wichtige Schnittstelle, die eine identische Grundlage für alle verschiedenen Client-Technologien repräsentiert.

Das Konzept sieht folgendermaßen aus:

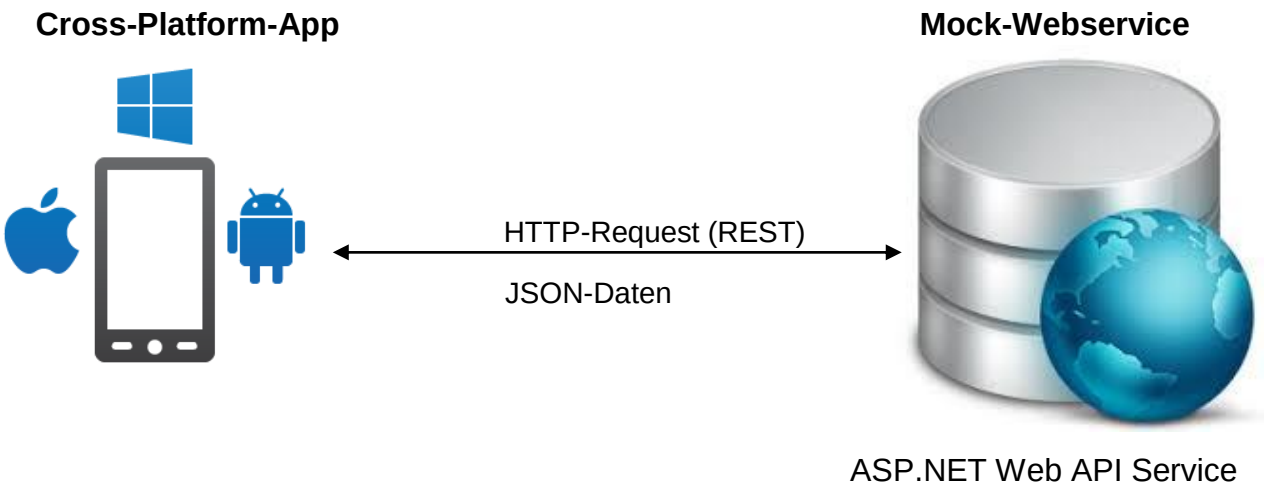


Abbildung 39: Schnittstellen

Das Client-Programm wird mit den Technologiekandidaten entwickelt und notwendige HTTP-Requests werden in den jeweiligen verschiedenen Programmiersprachen implementiert.

Auf der anderen Seite beantwortet der Mock-Webservice, der mithilfe eines ASP.NET Web API Services abgebildet wird, die Anfragen vom Client.

Weil die Daten in einem geeigneten Format übertragen werden müssen, wird hierbei durchwegs JSON verwendet, da damit eine schlanke, effiziente und gutleserliche Datenübertragung gewährleistet wird.

5.4 Implementierung

In diesem Kapitel wird eine Übersicht von allen vom Projektteam behandelten Technologien und den damit erstellten Applikation geboten. Für Technologiekandidaten, welche im Laufe der Diplomarbeit ausschieden, werden die Gründe für die Einstellung mit der jeweiligen Technologie gegeben.

5.4.1 ERD

Basis aller Softwareschichten des TeamPlay-Prototyps ist ein Datenmodell, das für Client und Server gleich sein muss. Das vom Projektteam erstellte Datenmodell entspricht nicht dem originalen TeamPlay-Modell, da bei der Projektabwicklung die wichtigsten Bestandteile des Modells ausreichen.

Die Visualisierung des Modells kann in folgendem ERD zusammengefasst werden:

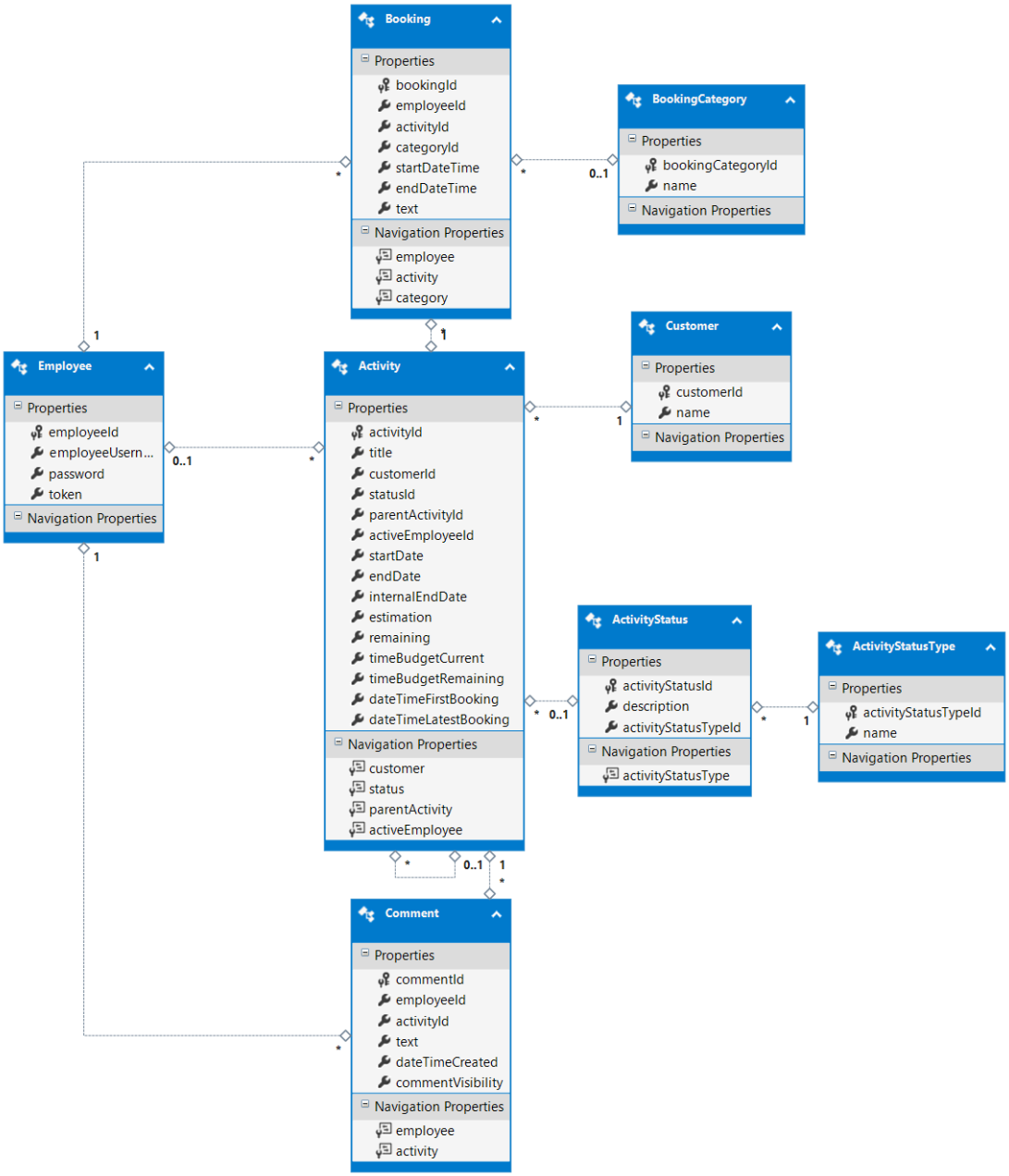


Abbildung 40: Datenmodell

5.4.2 PhoneGap

5.4.2.1 Entwicklungsschritte

Folgende Schritte sind notwendig bevor mit der Entwicklung gestartet werden kann:

- Eclipse von der Eclipse Website [25] herunterladen und installieren
- Google Plugin für Eclipse installieren
- Neues Projekt anlegen: New > Web Application Project
- Projekt debuggen: Debug As > Web Application
- GWT Developer Plugin für Internet Explorer, Firefox oder Chrome installieren
- Debug URL im Browser aufrufen

Um das App am Smartphone testen zu können muss es zunächst mit GWT zu HTML, CSS und JavaScript kompiliert werden. Dies erfolgt über das Menü: Google > GWT Compile. So wird ein Ordner mit dem Namen „war“ erstellt, der die kompilierten Dateien beinhaltet. Diese werden nun mit Phonegap in eine Applikationsdatei verpackt, die auf verschiedenen mobilen Plattformen installiert werden kann.

Screenshot der Login Seite des PhoneGap Prototyps:

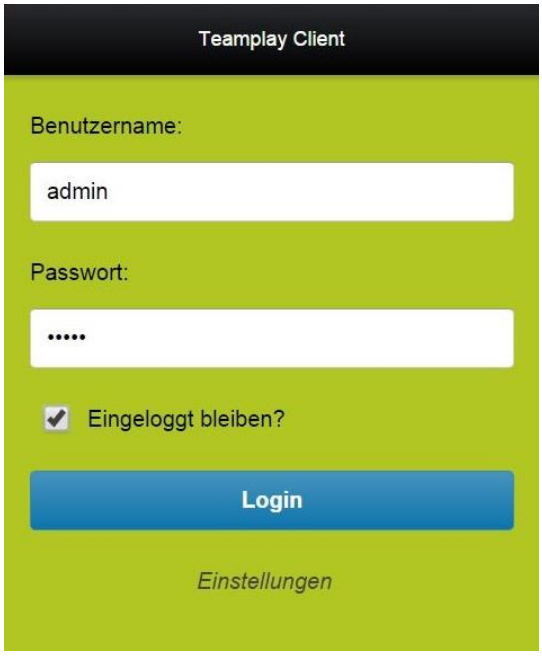


Abbildung 41: Login PhoneGap

5.4.2.2 MVP Pattern

Für die Entwicklung wurde ein Model-View-Presenter Pattern gewählt, da dadurch mehrere Entwickler gleichzeitig an einem Projekt arbeiten können und vergleichsweise wenig Code erforderlich ist um ein funktionsfähiges Projekt zu erstellen.

In seinem Kern ist MVP ein Entwurfsmuster, das in folgende Komponenten untergliedert werden kann:

Model

Das Model wird mit den Daten aus der Server Komponente gefüllt und von der View zur Darstellung verwendet.

View

Zum View gehören die UI Elemente, die Daten aus dem Model anzeigen und Benutzereingaben an den Presenter senden.

Presenter

Im Presenter befinden sich komplexe Anwendungen und die Geschäftslogik, welche sich um die Änderungen an der Benutzeroberfläche, sowie an dem Model kümmert.

5.4.2.3 Komplexität

Da GWT in Java programmiert wird, fällt es Programmierern mit Java Kenntnissen sehr leicht sich in GWT zurechtzufinden. Die Programmierung der Logik erfolgt ausschließlich in Java. Bei der UI Programmierung sind Grundkenntnisse in HTML und CSS von Vorteil, auch wenn nur Java Code geschrieben wird.

GWT Programmierung ist grundsätzlich sehr schreibintensiv, es gibt zwar einen GUI Builder, der aber nicht besonders hilfreich ist und extra installiert werden müsste. Am saubersten lässt sich das UI mit dem UIBinder programmieren, welcher die Möglichkeit bietet, das Layout in XML Form zu definieren.

Aufgrund dessen, dass GWT normalerweise für die Entwicklung von Webapps verwendet wird, ist die Unterstützung von Eingabefeldern für mobile Geräte (z. B. Date Picker) eher gering, kann aber durch die neuen „HTML5 Input Types“ erreicht werden.

Ein Vorteil, der durch die Programmierung mit Java kommt, ist, dass beliebige Design Patterns verwendet werden. Natürlich werden Standard Features wie Garbage Collection, IntelliSense oder Libraries unterstützt.

5.4.2.4 Performance

IDE

Eclipse ist eine sehr flexible und mächtige IDE, die keine Performance Probleme aufweist und allgemein sehr ausgereift ist. Einzig der erstmalige Start dauert einige Sekunden, was aber vernachlässigt werden kann, da die Entwicklungsumgebung normalerweise nur einmal am Tag gestartet wird.

Im Gegensatz zur Eclipse ist das GWT Plugin nicht so performant und benötigt für viele Dinge länger als die Release-Version. Dafür unterstützt dies beim Debuggen neben Standardfeatures wie Break Points, auch Hot Code Replacement.

Applikation

Die mit PhoneGap verpackte App läuft als Webview auf dem Smartphones und wird über den internen Browser gerendert. Da Browser inzwischen sehr schnell Webseiten darstellen können, ist auch die Performance der Webapp dementsprechend gut. Auch im Vergleich zu den anderen Technologien ist der RAM Verbrauch eher gering und die Reaktionszeit niedrig.

Emulator

Das GWT Projekt lässt sich jederzeit mithilfe des oben erwähnten GWT Plugin z. B. mit Chrome debuggen. Man sollte sich nicht von der Geschwindigkeit im Debug Modus täuschen lassen, da hier die Performance gegenüber der Release-Version um einiges schlechter ist. Performance Tests sind also unbedingt mit den kompilierten Dateien durchzuführen.

PhoneGap Build von Adobe bietet ihren Benutzern "Hydration", was den Vorteil mit sich bringt, dass für installierte Apps Updates ohne Neuinstallation über die Cloud durchgeführt werden können.

5.4.2.5 Evaluierung

Nach Abschluss der Prototypentwicklung wurde die Technologie anhand der folgenden Kriterien ausgewertet.

Die Punktevergabe bezieht sich auf den Vergleich zu anderen Technologien.

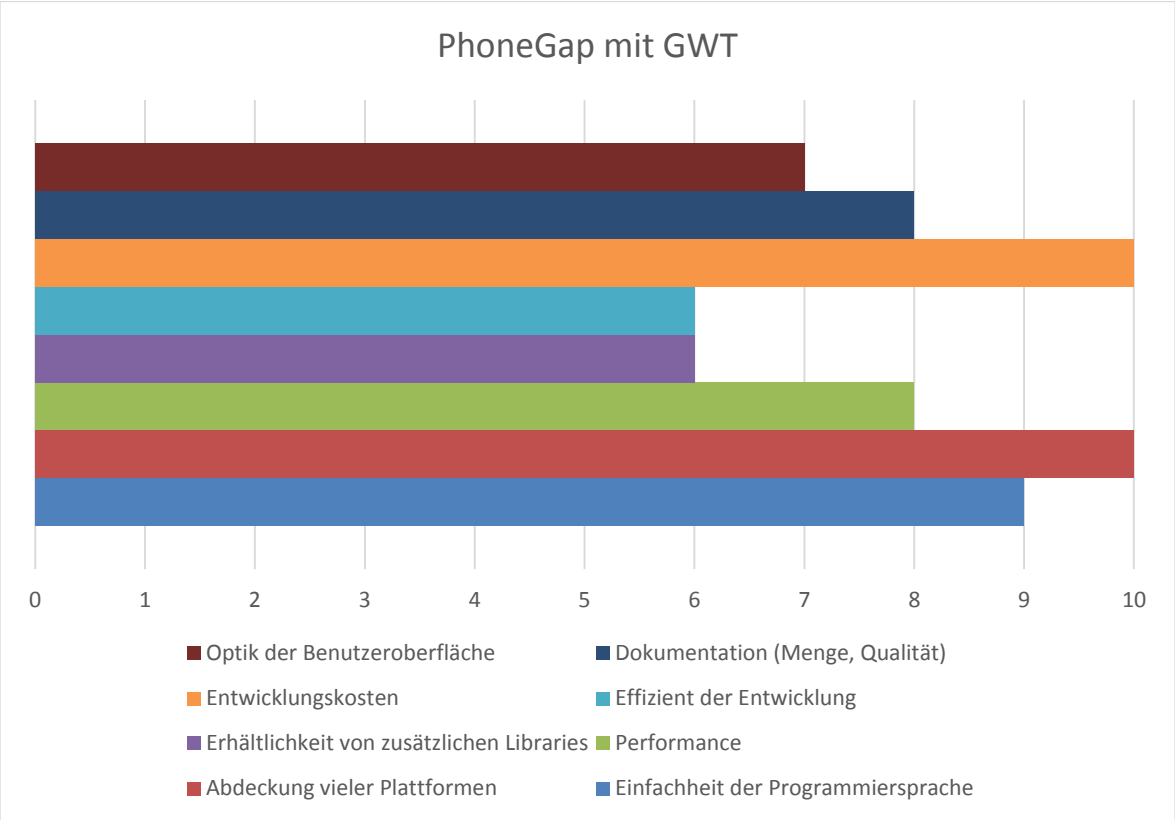


Abbildung 42: Evaluierung PhoneGap (GWT)

Optik der Benutzeroberfläche

7/10

Da GWT hauptsächlich auf den Desktopbereich abzielt, gibt es nur wenige Bibliotheken zur Entwicklung einer grafischen Oberfläche für mobile Geräte. Diese sind auch nur begrenzt zufriedenstellend, sodass in unserem Fall die grafische Oberfläche selbst entwickelt wurde. Adobe AIR bietet in diesem Bereich eine einfachere Lösung.

Dokumentation (Menge, Qualität)

8/10

Allgemein gibt es für GWT eine große Menge an Dokumentation, welche sich nicht nur auf die offizielle GWT-Seite beschränkt, sondern auch durch zahlreiche Foren und Blogs ergänzt wird. Leider gibt es diese Vielfalt an Dokumentation für die GWT Entwicklung im mobilen Bereich nicht.

Entwicklungskosten

10/10

Es gibt keine Lizenzkosten für PhoneGap oder für GWT, was diese Technologie eindeutig zur kostengünstigsten unserer getesteten Technologien macht.

Effizienz der Entwicklung

6/10

GWT wird durch die weit verbreitete Programmiersprache Java entwickelt, was natürlich alle Vorteile von Java mit sich bringt. Trotzdem muss z. B. das Model in einer Mischung aus Java und JavaScript geschrieben werden, um die einwandfreie Konvertierung von JSON zu Modellen sicherzustellen. Außerdem sind HTML und CSS Kenntnisse bei der GUI Entwicklung dringend notwendig, da in Java die GUI nicht sehr gut designt werden kann und dazu ein eigener CSS Code notwendig ist.

Zudem ist GWT eher codeintensiv, was aber durch die Verwendung des UIBinders verbessert werden kann. Durch den UIBinder kann die grafische Oberfläche mit einer XML-Datei erstellt werden und verringert so die Zeilen an Code, die benötigt werden um eine grafische Oberfläche zu erstellen.

Erhältlichkeit von zusätzlichen Libraries

6/10

Für GWT sind sehr viele Libraries vorhanden. Aber speziell für die Entwicklung für mobile Geräte gibt es – ähnlich wie bei der Dokumentation – nur sehr wenige, welche alle Performance-Probleme mit sich bringen und daher nicht zu empfehlen sind.

Performance

8/10

Natürlich kommt die Leistung nicht an die einer nativen App heran, aber bei einem eigens erstellten einfachen Design und ohne der Verwendung von übertriebenen grafischen Effekten, gibt es keine Performance Probleme.

Abdeckung vieler Plattformen

10/10

Alle wichtigen Plattformen werden unterstützt und durch die HTML, CSS und JS Basis kann neben den Apps für die jeweiligen Betriebssysteme auch eine Web Applikation erstellt werden.

Einfachheit der Programmiersprache

9/10

Aufgrund der Verwendung der sehr weit verbreiteten Programmiersprache Java hat PhoneGap mit GWT hier einen klaren Vorteil. Ein Punkt wurde abgezogen, da neben Java Kenntnissen auch Fähigkeiten in HTML, CSS und JavaScript notwendig sind.

5.4.3 Adobe AIR

Der erste entwickelte Prototyp wurde mit der Technologie Adobe AIR durchgeführt. Im aktuellen Kapitel werden der Entwicklungsprozess und die Erfahrungen daraus genauer etwas genauer erläutert.

5.4.3.1 Entwicklungsschritte

Die Schritte vom Einrichten der Entwicklungsumgebung bis zur Fertigstellung setzen sich folgendermaßen zusammen:

1. **Adobe AIR**

   Zu Beginn wird die Adobe AIR Runtime auf Testgeräten sowie auf dem Entwicklungsrechner installiert. Adobe AIR ist für Desktop-Systeme unter Adobe AIR Downloadseite [26] erhältlich.

2. **Entwicklungsumgebung**
   Als nächstes wird Adobe Flash Builder, die Entwicklungsumgebung selbst, installiert. Adobe bietet die Software zum Kauf bzw. als 60-Tage-Demoversion an [27].
3. **Android Development Kit**
   Mit Flash Builder werden prinzipiell gleichzeitig auch Android-Werkzeuge mitgeliefert. Im Zuge der Diplomarbeit wurde hier jedoch ein Problem entdeckt, da diese Android-Werkzeuge veraltet waren. Weiters konnte im Flash Builder aufgrund den Android-Tools kein Debug-Vorgang auf ein mit USB-Kabel verbundenes Android-Smartphone durchgeführt werden, weil das Gerät nicht in der Geräteliste aufscheint bzw. als „offline“ angezeigt wird.
   Im letzteren Fall erschien auch folgende Fehlermeldung:
   "Device appears to be offline. Restarting the device may fix the problem."
   Das Problem wurde so gelöst:
   - Download des aktuellsten Android SDKs [28] (GET THE SDK FOR AN EXISTING IDE)
   - Im Ordner <Installationsverzeichnis Flash Builder>\sdks\4.6.0\lib\android\bin werden die Dateien
     - aapt.exe
     - dx.jar
     - adb.exe

- AdbWinApi.dll
- AdbWinUsbApi.dll

durch die gleichnamigen Dateien vom aktuellen Android SDK ersetzt.

Hinweis: Die Dateien im Android SDK befinden sich unter folgenden Pfaden:

Datei	Pfad
aapt.exe	Android SDK\build-tools\< latest_version>\aapt.exe
dx.jar	Android SDK\build-tools\<latest_version>\lib\dx.jar
adb.exe	Android SDK\platform-tools\adb.exe
AdbWinApi.dll	Android SDK\platform-tools\AdbWinApi.dll
AdbWinUsbApi.dll	Android SDK\platform-tools\AdbWinUsbApi.dll

Tabelle 13: Wichtige Dateien im Android SDK

Die ursprüngliche Quelle für diese Strategie wurde in einem Adobe-Internetforum gefunden [29].

4. **Prototypentwicklung**
   Nachdem alle Voraussetzungen für die Programmierung erledigt wurden, galt es, die Technologie zu erlernen. Mithilfe der angeeigneten Kenntnisse wurde der Prototyp mit Adobe AIR schließlich entwickelt.

Screenshots des Adobe AIR Prototyps (Android-Smartphone):

Abbildung 43: Hauptmenü (Adobe AIR)

Abbildung 44: Aktivitätenansicht (Adobe AIR)

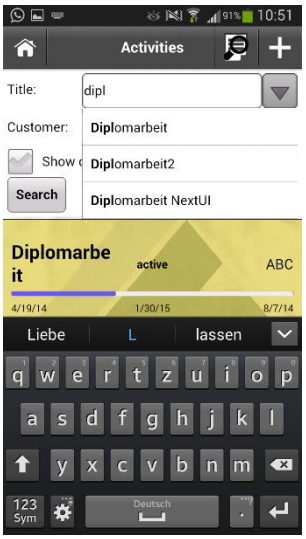


Abbildung 45: Suchfilter (Adobe AIR)

Abbildung 46: Aktivitätsübersicht (Adobe AIR)

Abbildung 47: Subaktivitäten einer Aktivität (Adobe AIR)

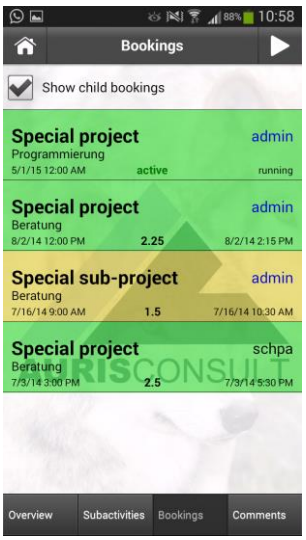


Abbildung 48: Buchungen einer Akivität (Adobe AIR)

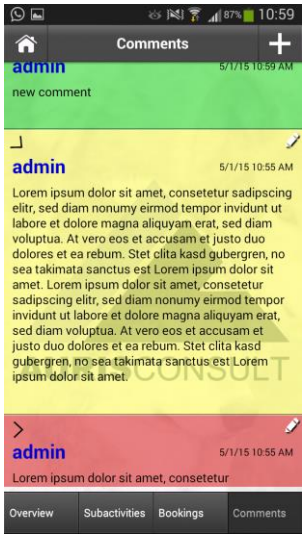


Abbildung 49: Kommentare einer Aktivität (Adobe AIR)

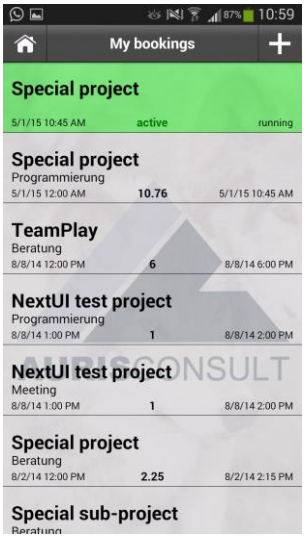


Abbildung 50: Buchungsübersicht (Adobe AIR)

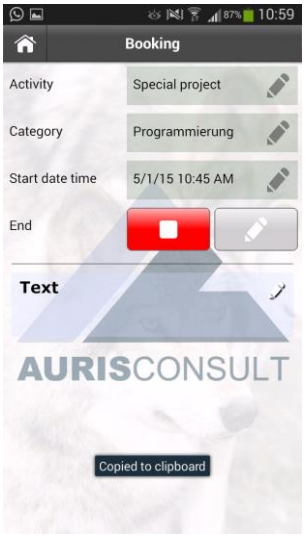


Abbildung 51: Buchung (Adobe AIR)

5.4.3.2 Bestandteile

Hinter der Adobe AIR App-Erstellung steckt nicht nur ActionScript sondern es sind verschiedene Bestandteile zu unterscheiden:

ActionScript

ActionScript ist die Programmiersprache zur Einbindung von Logik und Interaktivität in Adobe AIR Applikationen.

Flex

Flex oder das Flex Framework ist eine Sammlung von ActionScript-Klassen, die das gesamte Aussehen des User Interfaces beschreiben. Spark ist beispielsweise eine Komponentenarchitektur für Flex (Spark Components).

MXML

…Macromedia eXtensible Markup Language

MXML ist eine XML-basierte Beschreibungssprache und wird in Kombination mit Flex verwendet, um die grafische Oberfläche zu gestalten. MXML wird während des Kompilierens nach ActionScript umgewandelt.

5.4.3.3 Komplexität

5.4.3.3.1 Data Binding

In Flex Mobile wird eine simple Form von DataBinding unterstützt.

Dabei wird in XML bei einem GUI-Element auf ein Property eines Objektes verwiesen, mit welchem das Data Binding hergestellt werden soll.

Die Syntax in XML ist dabei folgende:

```
<s:Label text="{obj.text}"/>
```

Wichtig dabei ist, dass das Objekt „obj” eine Variable bzw. ein Property mit entsprechender Annotation definiert:

```
[Bindable]
public var text:String = “Hello world!”;
```

Ist eine Variable bzw. ein Property auf diese Weise implementiert, kopiert Flex automatisch den neuen Wert in die GUI-Komponente (per Event), falls sich der Wert der Variable bzw. des Properties verändert.

Performance-Problem

Da Bindings in Flex die Performance beeinträchtigen – vor allem, wenn diese auf einer View zahlreich auftreten – sollten Data Bindings nicht zu häufig in einer Applikation vorkommen. Besonders in **Listenelementen** ist von der Verwendung abzuraten, um ein stabiles App-Verhalten zu gewährleisten.

5.4.3.3.2 Garbage Collection

Adobe AIR sowie der Flash Player verwenden eine Kombination von „Reference Counting“ und „Mark-and-sweep“. Der Grund für die Kombination ist, dass beim reinen

„Reference Counting“ Objekte mit zirkulären Beziehungen nicht gelöscht werden. Dieses Problem wird mit dem „Mark-and-sweep“-Prinzip gelöst.

Während dem Entwicklungsprozess fallen bezüglich dem Speichermanagement keine Komplikationen bzw. kein zusätzlicher Aufwand an.

Weak references

Weak references in Adobe AIR spielen eine wichtige Rolle, können allerdings nur für manche Objekttypen verwendet werden (EventListeners, Dictionaries). Objekte, die als „weak reference“ markiert sind, sind für den Garbage Collector während dem Suchen von erreichbaren Objekten unsichtbar. Der Pfad zu dieser Referenz wird während dem Markierungsprozess des GCs somit nicht besucht. Folglich wird das betreffende Objekt nicht markiert und vom GC aufgesammelt. Dieses Prinzip wird beispielsweise bei Event-Listeners angewendet. Wichtig ist, dass nicht mehr benötigte Referenzen entfernt werden, indem beispielsweise „removeEventListener()“ aufgerufen wird.

Genauere Details sind auf Lernseiten von Adobe zu finden [30].

5.4.3.3.3 Designer

Flash Builder stellt für MXML-Dateien eine **Designer-Ansicht** zur Verfügung. Diese ist ähnlich wie jene bei der Android-Entwicklung.

Ob der Designer verwendet wird oder nicht hängt von der bevorzugten Vorgehensweise des Programmierers ab. Da der Aufbau der GUI in XML ohnehin zuerst erlernt werden muss, empfiehlt es sich, in der XML-Ansicht zu entwickeln, um volle Übersicht und Kontrolle über die View zu haben. Möchte man nicht auf den Designer verzichten kann sich eine Kombination von XML und Designer durchaus als vorteilhaft erweisen.

Anmerkung: In der Prototypentwicklung vom TeamPlay-Client wurde ausschließlich die XML-Ansicht verwendet.

5.4.3.3.4 Standard-GUI-Elemente

Die für die GUI verwendete Library „Spark“ bietet ein breites Angebot an Standard-GUI-Elementen. Sogar Komponenten für Datums- und Zeitangaben sind vorhanden.

Nicht alle Komponenten sind für mobile Apps anwendbar. Im Flash Builder wird darauf hingewiesen, welche Komponenten für Desktop bzw. für mobile System bevorzugt werden.

5.4.3.3.5 Eigene GUI-Elemente

Die Spark-Library bietet weiters die Möglichkeit, selbstgestaltete GUI-Elemente einzuführen.

Dazu gibt es zwei Möglichkeiten:

Skins

Mithilfe von Skins kann das Aussehen bestehender Elemente neu definiert werden. Es wird ein Elementen nicht in seiner Grundstruktur verändert, sondern nur anders angezeigt.

Skin-Definitionen können allgemein oder speziell für einen bestimmten Elementtyp definiert werden.

Da sich viele und/oder komplexe Skin-Definitionen beachtlich auf die Performance auswirken, sollten diese so einfach wie möglich gehalten oder vermieden werden.

Klassen

Weiters können in eigenen MXML-Dateien oder ActionScript-Klassen eigene GUI-Elemente gebaut werden.

Dieses Prinzip wird beispielsweise bei ItemRenderern für ListItems angewendet. Dort werden beliebige weitere Elemente hinzugefügt und formatiert.

Speziell bei Listenelementen sollte das Aussehen nicht durch XML sondern direkt in Action Script definiert werden, da mit reinem Action Script die GUI wesentlich schneller aufgebaut werden kann.

5.4.3.4 Programmierpattern

Ähnlich wie bei PhoneGap wurde auch bei Adobe AIR ein Model-View-Presenter-Pattern angewendet. Zusätzlich wurde ein Controller implementiert, der das Datenmodell hält und die von den Presenter geforderten Operationen durchführt. Der Controller modifiziert also das Datenmodell und benachrichtigt die View über Veränderungen. Die View kommuniziert dabei ausschließlich mit dem jeweils passenden Presenter. Weiters wird auch ein Singleton eingebunden, welcher die Presenter und das Datenmodell hält.

Model-View-Controller-Presenter

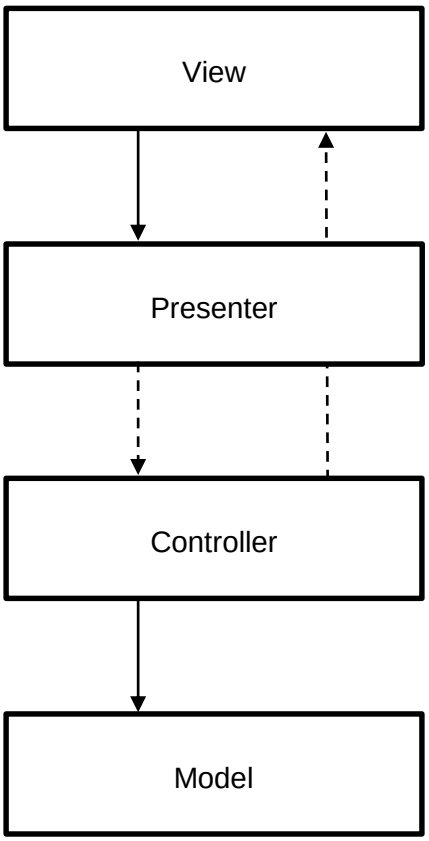


Abbildung 52: Pattern bei Adobe AIR

Ein gestrichelter Pfeil bedeutet, dass die Kommunikation über Events abgehandelt wird und die betroffenen Schichten somit lose gekoppelt sind.

Durchgehende Pfeile signalisieren eine Verbindung über eine tatsächliche Referenz.

Der große Vorteil des Patterns ist, dass die View rasch ausgetauscht werden kann, während alle Schichten unterhalb der View unverändert bleiben. Das ergibt sich aus der Verwendung von Events. Der View obliegt es dann, wie sie die Events behandelt.

Klassenarchitektur

Die grundsätzliche Strukturierung der Applikation besteht aus folgenden Bestandteilen:

- Views (jede View wird durch jeweils eine Klasse/Datei abgebildet)
- Presenters (zu jeder View existiert eine Presenter-Klasse)
- AppModel (Singleton-Klasse, hält das Datenmodell und die Presenter)
- BaseAppController (verändert Daten des AppModels, benachrichtigt Presenter über Änderungen, View führt die Benachrichtigungen durch)
- StorageProvider und ServiceProvider zur Abwicklung von File- und Weboperationen.

Die Klassenstruktur setzt sich wie folgt zusammen:

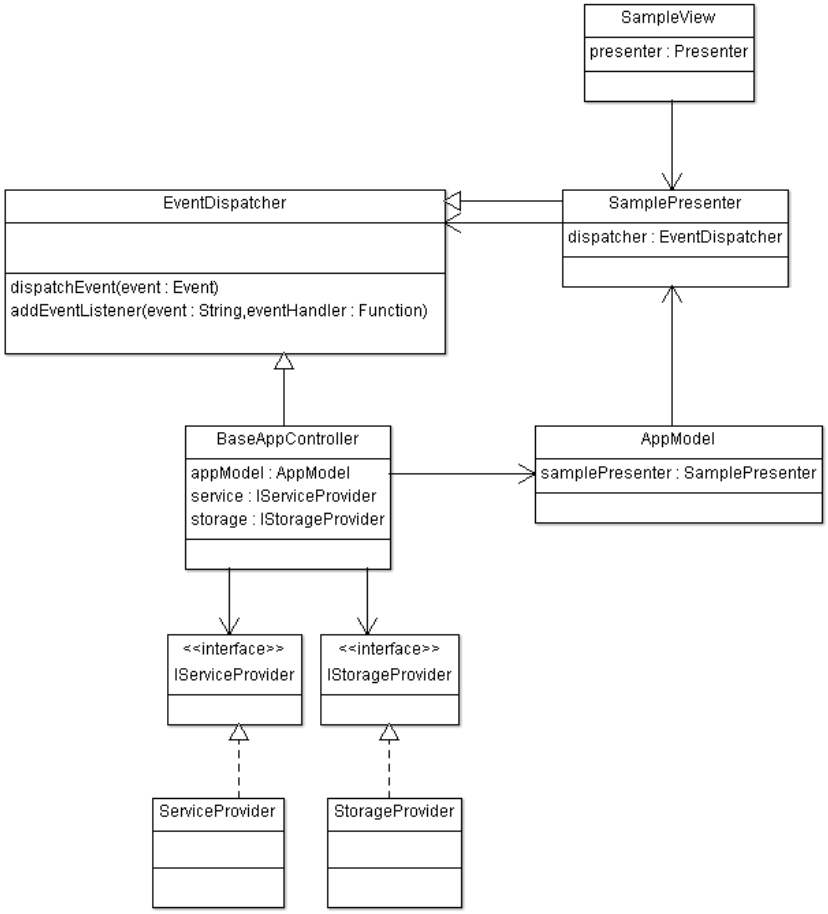


Abbildung 53: Klassenarchitektur

In diesem Diagramm wurde anstatt der vielen implementierten Presenter nur eine Klasse „SamplePresenter“ zur Veranschaulichung gewählt. Dasselbe gilt für „SampleView“.

Die Klasse EventDispatcher ist in ActionScript schon vordefiniert und dient zur Kommunikation zwischen den einzelnen Klassen.

Zusammenfassend werden hier die einzelnen Aufgaben der Bestandteile aufgelistet:

- Views
    - Abbildung aller im Planungsprozess vorgesehenen grafischen Benutzeroberflächen
    - Befehle an den für die jeweilige View zugeordneten Presenter geben (ausgelöst durch beispielsweise einem Klick auf einen Button)
    - auf Veränderungen im Modell reagieren, die vom Controller mitgeteilt werden (z. B. Text in einem Textfeld mit den veränderten Modelldaten aktualisieren)
- Presenter
    - Befehlsaufnahmestellen für die Views (für jede View einen dazugehörigen Presenter)
    - Weiterleitung von Befehlen an den Controller

    - Repräsentation der auf der Benutzeroberfläche dargestellten Daten
- AppModel
    - ist ein Singleton und lässt deshalb während der gesamten Programmausführung nur eine Instanz dieser Klasse zu
    - Haltung des Datenmodells (Aktivitäten, Buchungen, Mitarbeiterinformationen) und der Presenter
- BaseAppController (=Controller)
    - Implementierung der Businesslogik
    - Befehle von Presenter entgegennehmen und die geforderten Operationen konkret durchführen
    - Weiterreichen von Benachrichtigungen über Änderungen oder abgeschlossene bzw. fehlgeschlagene Operationen an die View (via Event, welches an den Presenter gesendet wird; die View hat davor einen Event-Handler am Presenter registriert und verarbeitet schlussendlich das Event)
- StorageProvider und ServiceProvider
    - dienen dem Controller zur Abwicklung von Web- und Dateioperationen

5.4.3.5 Performance

Im Allgemeinen wurde bei Adobe AIR festgestellt, dass die Performance stark von der Anzahl der MXML-Komponenten abhängt. Vor allem eine häufige Verwendung von Layout-Elementen mit ihren aufwändigen Berechnungen verringert die Performance.

Am besten ist es, Views nicht in XML sondern mithilfe von ActionScript zu definieren, da ActionScript wesentlich schneller ausgeführt werden kann als komplexe Strukturen in XML.

In einem Adobe-Artikel ist eine Checkliste zu Performance-Verbesserungen von Flex-Anwendungen zu finden [31].

5.4.3.6 Evaluierung

Nach Abschluss der Prototypentwicklung wurde die Technologie anhand der folgenden Kriterien ausgewertet.

Die Punktevergabe bezieht sich auf den Vergleich zu anderen Technologien.

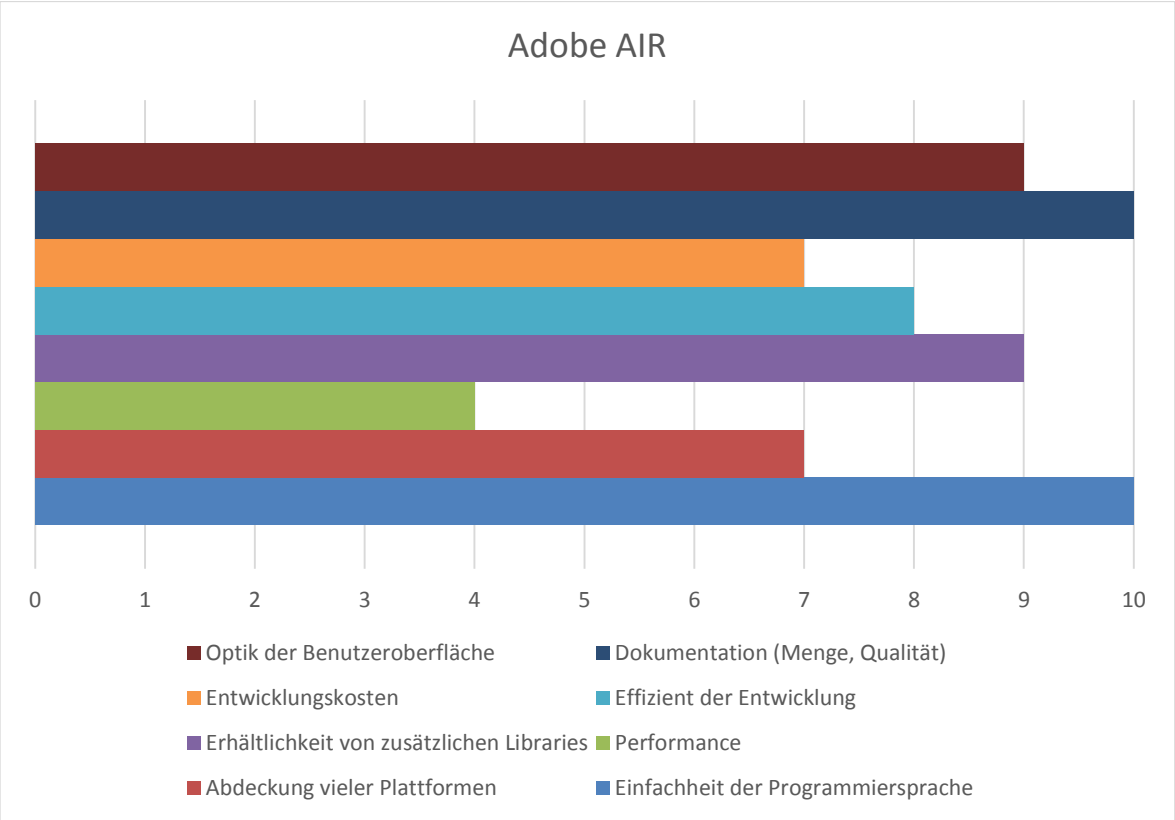


Abbildung 54: Evaluierung Adobe AIR

Optik der Benutzeroberfläche

9/10

Die grafische Benutzeroberfläche von Adobe AIR ist verglichen zu PhoneGap sehr einfach zu gestalten und vom optischen Gesamteindruck sehr zufriedenstellend.

Dokumentation (Menge, Qualität)

10/10

Aufgrund der weiten Verbreitung von Adobe-Produkten sind die Flash-Plattformen in großen Mengen dokumentiert, nicht nur auf offiziellen Adobe-Seiten sondern auch in vielen Foren befinden sich hilfreiche Dokus. Die Dokumentation ist überwiegend auf Englisch erhältlich, gut verständlich und übersichtlich strukturiert.

Entwicklungskosten

7/10

Im Vergleich zu PhoneGap mit GWT sind die Kosten für die Entwicklungsvoraussetzungen (Flash Builder) hoch (wenngleich niedriger bei Embarcadero).

Effizienz der Entwicklung

8/10

Es besteht nur wenig Einarbeitungsaufwand durch die einfache Sprache ActionScript, die GUI-Library „Spark“ (=> XML) und deren Komponenten müssen jedoch erst erlernt und verstanden werden. Ein Nachteil ist, dass ein passendes Pattern gewählt werden muss, damit die View einfach ausgetauscht werden kann, denn für Desktop- und Mobilsysteme müssen verschiedene Benutzeroberflächen gestaltet werden.

Erhältlichkeit von zusätzlichen Libraries

9/10

Auf offiziellen Adobe-Seiten sowie von Projekten auf GitHub sind ausreichend viele (hilfreiche) Libraries vorhanden.

Performance

4/10

Wegen Performance-Problemen in der Bedienung der App (langsame Reaktionszeiten, Lag) wird einer der wichtigsten und für den Kunden ausschlaggebendsten Aspekte, nämlich die User-Experience und Benutzerfreundlichkeit, verletzt, daher wurde die Auswertung auf 40% festgelegt

Abdeckung vieler Plattformen

7/10

Zwar sind ausreichend viele Plattformen abdeckbar, jedoch ist anzumerken, dass Desktop-App (Mac, Windows) und Mobile-App (iOS, Android) in getrennten Flash-Projekten zu erstellen sind.

Einfachheit der Programmiersprache

10/10

Aufgrund Verwendung der einfach erlernbaren Sprachen ActionScript und XML wurde volle Punktzahl vergeben. Verglichen zu Embarcadero oder PhoneGap ist Adobe AIR um ein Vielfaches einfacher erlernbar.

5.4.4 Eclipse Scout

Die geplante Technologie „Eclipse Scout“ wurde bei der Diplomarbeit nicht umgesetzt. Schon nach einer Woche Entwicklungszeit wurde die Fortführung des Prototyps abgebrochen. Der Grund dafür ist, dass Eclipse Scout grundlegende Anforderungen, die der Auftraggeber stellte, nicht erfüllen kann und auf lange Sicht im Unternehmen keine effiziente Entwicklungsmöglichkeit darstellt.

Die Basiskriterien sind im Punkt 2.4 zu finden.

Die Untauglichkeit der Technologie wurde durch Überprüfung und Auswertung auf nachfolgende Kriterien festgestellt. Das Ergebnis führte schließlich auf Auftraggeber-Seite auch zum Ausschluss der Technologie.

5.4.4.1 Kriterien im Überblick

Kriterium	**von Eclipse Scout unterstützt?**
Erzeugung einer nativen (mobilen) App	nein
Erstellung von Desktop-Applikation	ja (Swing und SWT; JavaFX geplant)
Offlinefähigkeit von Applikationen	nur für Swing-/SWT-Anwendungen (Desktop)
Beschreibungssprache für grafische Oberfläche	nein (durch Einbindung von JavaFX zukünftig unterstützt)
Designer zur Gestaltung der grafischen Oberfläche	nein
Im Web aufrufbar	ja
JSON-Parser vorhanden	nur mit externer Library
Cloud-Deployment (Backend)	ja
Standalone-Deployment (Backend)	ja
Wiederverwendbarkeit von bestehenden Backends	nur bedingt durch Kommunikation des Eclipse Scout Backend mit dem bestehenden Backend, was programmiertechnisch aufwändig ist und die Webrequests aus der Clientsicht verlangsamt
Dokumentation ausreichend	ja

Tabelle 14: Erfüllung der Kriterien bei Eclipse Scout

5.4.4.2 Maßgebliche Ausscheidungskriterien

Die ausschlaggebenden Gründe, die zur Ausscheidung der Technologie führten, sind folgende:

Native mobile Apps können nicht erzeugt werden

Zwar können Desktop-Applikationen als ausführbare Programme entwickelt werden, so ist Vergleichbares für mobile Betriebssysteme nicht möglich. Das Defizit wird anhand dieser Grafik zum Ausdruck gebracht:

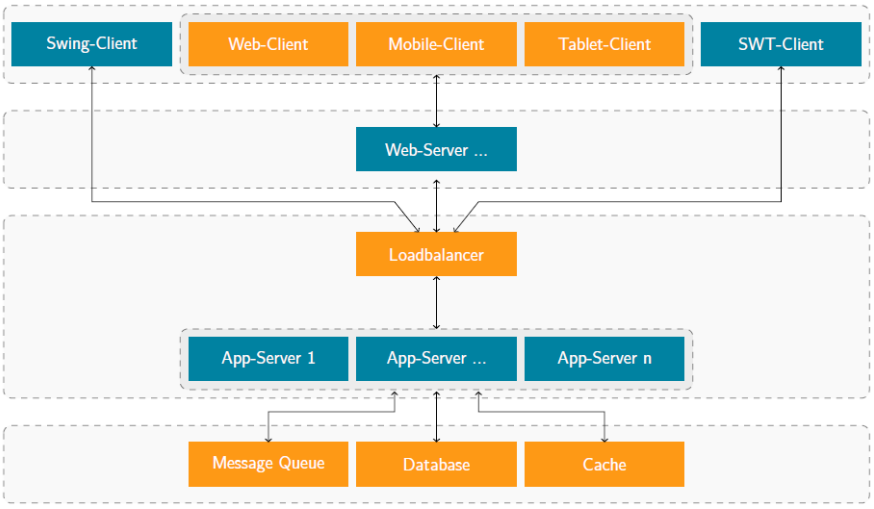


Abbildung 55: Eclipse Scout Architektur [32]

Anhand der Abbildung ist zu erkennen, dass Mobile-Client sowie Tablet-Client einem Scout-Webserver abhängig sind und somit keine nativen Mobile-Apps erstellt werden können.

> For Scout web, tablet and mobile clients, the Eclipse RAP framework is used. The RAP framework provides an API that is almost identical to the one provided by SWT and allows to use Java for server-side Ajax. This setup implies that Scout tablet and mobile clients are not native clients but browser based. (Zimmermann 2014: 41) [33]

Bestehendes Backend nur schwer weiterzuverwenden

Der zweite Hauptgrund ist, dass Eclipse Scout die Entwicklung eines eigenen Backends voraussetzt. Da TeamPlay jedoch schon mit einer anderen Backend-Technologie realisiert wurde, würde das Konzept durch Eclipse Scout komplizierter, wartungsaufwändiger sowie programmieraufwändiger werden.

5.4.5 Embarcadero

Als Ersatz für Eclipse Scout gilt die Prototypentwicklung mit der Technologie Embarcadero. Als Programmiersprache wurde C++ angewendet.

Die Prototypentwicklung wurde bei Embarcadero jedoch schon nach wenigen Wochen eingestellt aus Gründen der unhandlichen Entwicklungsweise. Embarcadero erfüllte zwar vorläufig die wichtigsten Kriterien, doch stellte sich später als untauglicher Kandidat für plattformübergreifende Entwicklung größerer Businesslösungen heraus.

5.4.5.1 Programmierpattern

Die Technologie macht es schwierig, Entwurfsmuster wie MVC oder MVP einzusetzen. Embarcadero gibt hier ein striktes Muster vor, nach welchem die Entwicklungsweise vorgesehen ist. Der Entwickler kann dennoch ein Entwurfsmuster seiner Wahl anwenden.

Das Konzept ist folgendermaßen aufgebaut:

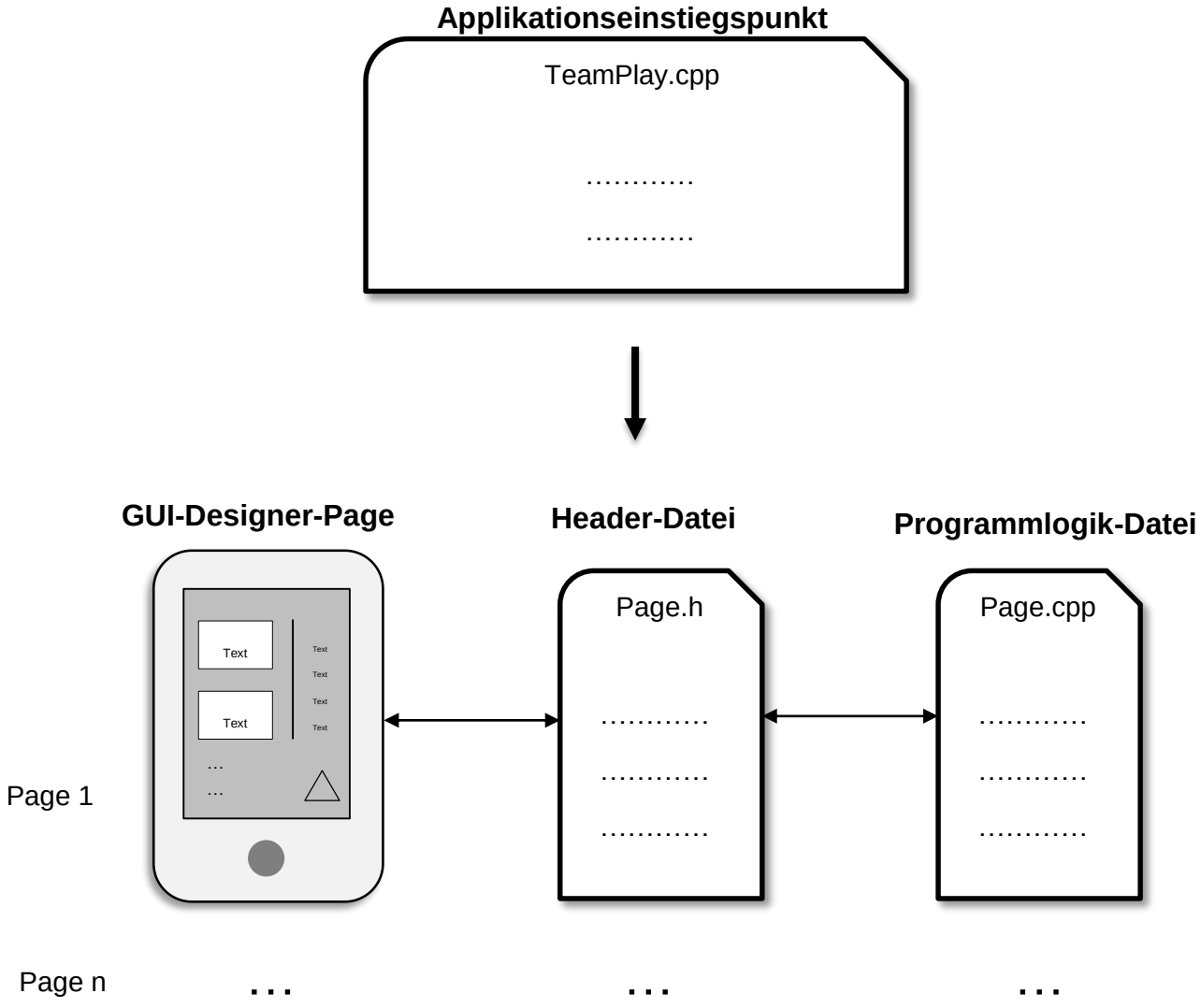


Abbildung 56: Entwicklungskonzept bei Embarcadero

Applikationseinstiegspunkt

Als Einstiegspunkt in die Applikation wird eine zentrale Klasse definiert. Diese Klasse ruft die erste Page der Applikation auf.

GUI-Designer-Page

Zur Gestaltung der Oberfläche stehen kein Texteditor und somit auch keine Beschreibungssprache (z. B. XML) zur Verfügung. Elemente werden per Drag & Drop von Toolboxes auf die grafische Designer-Seite platziert und mit Setzen von entsprechenden Eigenschaften formatiert.

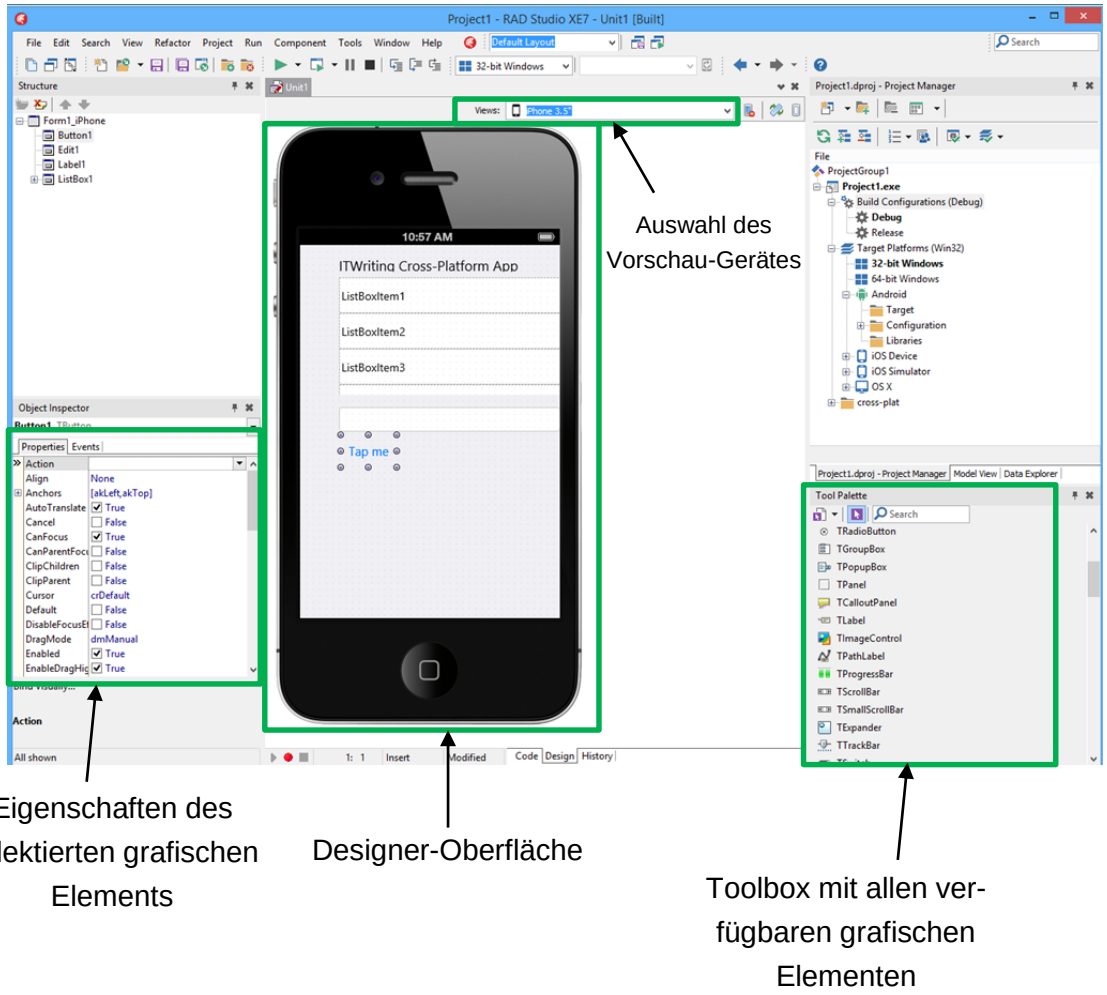


Abbildung 57: GUI-Designer RAD Studio [34]

Header-Datei

Zu einer Applikationspage wird in Embarcadero auch eine Header-Datei generiert. Diese Datei wird hauptsächlich vom GUI-Designer beschrieben, da dieser hier die einzelnen GUI-Elemente als Klassenvariablen definiert. Im Designer besteht weiters die Möglichkeit, Funktionen festzulegen, die bei bestimmten Events (z. B. Click) ausgeführt werden. Die Funktionsköpfe werden ebenfalls in der Header-Datei automatisch angelegt.

Programmlogik-Datei

Die in der Header-Datei Funktionsköpfe werden in der Programmlogik-Datei implementiert (Funktionsrümpfe). Beispielsweise könnte hier das Verhalten bei Klick auf einen Button behandelt werden und im Hintergrund HTTP-Requests ausgeführt werden. Weiters ist in der Programmlogik-Datei die Steuerung des Programms möglich (z. B. Navigation auf andere Pages). Die Datei nimmt also die Rolle einer Art Controller (~MVC-Modell) an.

5.4.5.2 Entwicklung

Trotz der Ausscheidung der Technologie wurde der Prototyp zu einem gewissen Grad entwickelt.

Von den geplanten Pages wurden folgende Pages (Mockups) umgesetzt:

- Login
- Hauptmenü
- Aktivitäten
- Aktivitätsdetails (Übersicht)

Beispiele für den Embarcadero-Prototyp (Android)

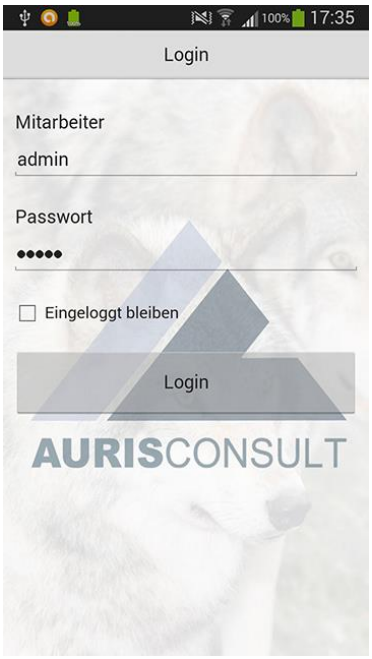


Abbildung 58: Login (Embarcadero)

Abbildung 59: Hauptmenü (Embarcadero)

5.4.5.3 Kriterien im Überblick

Kriterium	**von Embarcadero unterstützt?**
Erzeugung einer nativen (mobilen) App	ja (Android, iOS)
Erstellung von Desktop-Applikation	ja (Windows, Mac)
Offlinefähigkeit von Applikationen	ja
Gängige Programmiersprache (aus Sicht Auris IT Consult bzw. Diplomarbeits-team)	nein (C++ weder vom Unternehmen noch vom Projektteam regelmäßig verwendet)
Beschreibungssprache für grafische Oberfläche	nur visueller Designer mit RAD (nur per Drag & Drop aus der Toolbox; oder in der Programmlogik-Datei)
Flexibilität bei Wahl des Programmierpat-terns	nein (strenges Konzept von Embarcadero vorgegeben)
Designer zur Gestaltung der grafischen Oberfläche	ja
JSON-Parser vorhanden	ja
Dokumentation ausreichend	Dokumentation vorhanden jedoch oftmals nicht zielführend

Tabelle 15: Erfüllung der Kriterien bei Embarcadero

5.4.5.4 Maßgebliche Ausscheidungskriterien

Nachstehende Gründe trugen signifikant zur Ausscheidung der Technologie bei:

Unüberschaubarkeit

Nachdem in Embarcadero sogar HTTP-Requests oder Dateioperationen nur über Designer-Elemente gehandhabt werden können, wird der GUI-Designer nach kurzer Zeit sehr unübersichtlich, da bei Wachsen der Applikation die Anzahl der Elemente anstieg. Folglich wurde es schwierig, die tatsächliche GUI effizient zu gestalten, da die verschiedensten operativen Elemente die eigentliche Designer-Fläche überdeckten.

Keine Wahl eines Programmierpatterns

Durch die starke Bindung der View (Designer-Ansicht) an dahinterliegende Programmlogik ergibt sich der Nachteil, nicht beliebige Patterns anwenden zu können. Embarcadero weicht hier stark von konventionellen Entwicklungsmethoden ab. Der Effekt von diesem Umstand ist, dass somit die View nicht von der Programmlogik getrennt werden kann und den Entwicklungsprozess signifikant erschwert. Für Apps mit geringem Entwicklungs- und Wartungsumfang (2-3 Views) stellt Embarcadero eine praktische Methode dar (auch Embarcadero-Tutorials beziehen sich auf die schnelle Entwicklung kleiner Apps; siehe beispielsweise youtube.com/user/EmbarcaderoTechNet), für umfassendere Businessanwendungen ist Embarcadero als Technologiekandidaten auszuschließen.

5.4.6 Mock-Webservice (ASP.NET)

Zur Simulation eines TeamPlay-Backends für die Clientprototypen wurde vom Projektteam ein Mock-Webservice bzw. Dummy-Webservice entwickelt, der mit Testdaten ausgestattet wird. Die Testdaten können von den Clients anschließend angefordert und visualisiert werden.

Der Webservice wurde mit der Technologie ASP.NET mit der Programmiersprache C# umgesetzt (siehe 5.1.5).

5.4.6.1 Bestandteile

Nachdem der Webservice wie auch die Prototypen keine einsatz. B.ereiten Endprodukte darstellen, werden auch einzelne Funktionen wie z. B. der Login beim Webservice vereinfacht realisiert und nicht mit fortschrittlichen Sicherheitsmaßnahmen ausgestattet, da die Projektarbeit primär von einer aussagekräftige Veranschaulichung des TeamPlay-Systems handelt.

Folgende Bestandteile wurden für den Dummy-Webservice implementiert:

AccountController

System.Object

System.Web.Mvc.ControllerBase

System.Web.Mvc.Controller

AccountController

Login

```
public string Login(string username, string password, string callback = null)
```

Die Funktion Login ermöglicht es Mitarbeitern, sich am TeamPlay-System einzuloggen. Der Mitarbeiter gibt dafür seine Anmeldedaten bekannt (Parameter username und password). Bei erfolgreicher Anmeldung wird der Mitarbeiter als JSON-String zurückgegeben sowie ein Cookie (Authentication Ticket) zurückgegeben.

Parameters:

username:	Benutzername des Mitarbeiters (=Mitarbeiterkürzel)
password:	Passwort des Mitarbeiters
callback:	Optionaler Callback-String für JSONP-Aufrufe

Returns:

Mitarbeiter-Objekt als JSON-String

Logout

```
public void Logout()
```

Damit werden eingeloggte Mitarbeiter (verfügen über ein Authentication Ticket → Cookie) wieder ausgeloggt. Dabei wird vorausgesetzt, dass der Aufrufer das Authentication Ticket liefert, damit das Ausloggen funktioniert.

ActivityController

System.Object

System.Web.Http.ApiController

ActivityController

GetTopLevelActivities

```
public IEnumerable<ActivityDTO> GetTopLevelActivities(int userId,
                                                      string title, string customer)
```

Diese Funktion erlaubt es, die Aktivitäten des Unternehmens nach bestimmten Kriterien abzufragen. Die Kriterien sind:

- Nur meine Aktivitäten anzeigen → userId
- Nach Titel filtern → title
- Nach Kundenbezeichnung filtern → customer

Bei den Parameters title und customer wird keine Überprüfung durchgeführt, ob der Wert 1:1 dem Parameter entspricht, sondern es wird nach beliebigen Übereinstimmungen gesucht.

Parameters:

userId:	Optionale ID des anfragenden Mitarbeiters
title:	Optionaler Suchtext des Aktivitätstitels
customer:	Optionaler Suchtext für den Kunden

Returns:

Liste von Aktivtäten, die den optionalen Suchkriterien (title, customer) entsprechen

GetActivities

```
public IEnumerable<string> GetActivities(int userId, string searchText)
```

GetActivities gibt eine Liste von Aktivitätsnamen zurück, die den übergebenen Suchtext beinhalten. Optional kann wieder die ID des Mitarbeiters übergeben werden, wodurch nur jene Aktivitäten, bei denen der Mitarbeiter der aktive Mitarbeiter ist, infrage kommen. Die Funktion findet beispielsweise bei Autovervollständigungstextfeldern Anwendung, wo nach Eingabe von Zeichen eine Liste von ähnlichen Aktivitäten geboten wird.

Parameters:

userId:	Optionale ID des anfragenden Mitarbeiters
searchText:	Optionaler Suchtext des Aktivitätstitels

Returns:

Liste von Aktivitätsnamen, die den optionalen Suchkriterien (userId, searchText) entsprechen

GetAllActivities

```
public IEnumerable<ActivityDTO> GetAllActivities()
```

Die Funktion GetAllActivities wird dazu verwendet, alle Aktivitäten des Unternehmens abzufragen. Dasselbe Ergebnis könnte auch unter Aufruf von GetTopLevelActivitites ohne Filterung erzielt werden. Die Filterkriterien müssten jedoch dennoch angegeben werden. Zur Vereinfachung der Abfrage aller Elemente wurde diese Funktion eingeführt. Sie wird beispielsweise bei den Prototypen verwendet, um in einem Dropdown eine Auswahl zwischen allen Aktivitäten zu bieten.

Returns:

Liste von allen Aktivitäten in Form von Objekten

GetActivityOverview

```
public ActivityDTO GetActivityOverview(int activityId)
```

Die Funktion sammelt alle notwendigen Details der angeforderten Aktivität und gibt sie als Aktivitätsobjekt zurück. Beispielsweise wird diese Funktion benötigt, wenn bei einem der Prototypen die Übersichtsseite einer Aktivität aufgerufen wird.

Parameters:

activityId: ID der gewünschten Aktivität

Returns:

Aktivitätsobjekt mit allen für die Übersicht einer Aktivität benötigten Details

GetSubactivities

```
public IEnumerable<ActivityDTO> GetSubactivities(int activityId)
```

GetSubactivities fragt die zu einer Aktivität gehörenden Subaktivitäten ab und liefert diese zum Aufrufer.

Parameters:

activityId: ID der gewünschten Aktivität

Returns:

Liste von Aktivitäten, welche die Subaktivitäten der angeforderten Aktivität sind

GetBookings

```
public IEnumerable<BookingDTO> GetBookings(int activityId,
                                           bool showChildBookings)
```

Diese Funktion gibt alle Buchungen zu einer Aktivität zurück.

Parameters:

activityId: ID der gewünschten Aktivität

Returns:

Liste von Buchungen zur geforderten Aktivität

GetComments

```
public IEnumerable<CommentDTO> GetComments(int userId, int activityId)
```

Diese Funktion gibt alle Kommentare einer Aktivität zurück.

Parameters:

activityId: ID der gewünschten Aktivität

Returns:

Liste von Kommentaren zur geforderten Aktivität

GetActivityStatusList

```
public IEnumerable<ActivityStatusDTO> GetActivityStatusList()
```

Die Funktion dient zur Rückgabe der möglichen Status, die für eine Aktivität eingestellt werden können. Ein Status repräsentiert somit den aktuellen Zustand einer Aktivität.

Returns:

Liste von Status für Aktivitäten (z. B. aktiv, inaktiv, …)

BookingController

System.Object

System.Web.Http.ApiController

BookingController

GetBookings

```
public IEnumerable<BookingDTO> GetBookings(int userId)
```

GetBookings retourniert alle zu einem Mitarbeiter gehörigen Buchungen.

Parameters:

userId: ID des anfragenden Mitarbeiters

Returns:

Liste von Buchung für den anfragenden Mitarbeiter

GetBookingCategories

```
public IEnumerable<BookingCategoryDTO> GetBookingCategories()
```

Diese Funktion sorgt für die Abfrage aller möglichen Kategorien, mit der Zeit für eine Aktivität gebucht werden kann. Beispiele für Kategorien: Meeting, Beratung, Programmierung, …

Returns:

Liste von Buchungskategorien

CustomerController

System.Object

System.Web.Http.ApiController

CustomerController

GetCustomers

```
public IEnumerable<string> GetCustomers(string searchText)
```

Diese Funktion hat die Aufgabe, anhand eines Suchtextes alle dazu ähnlichen Kunden zu suchen und deren Namen an den Anfragenden zurückzuschicken. Ein Einsatzszenario dazu ist beispielsweise ein Textfeld mit Autovervollständigung, bei der nach Tippen von Buchstaben die dazu passenden Kunden über diesen Funktionsaufruf gefunden werden.

Parameters:

searchText: Optionaler Suchtext des Kundennamens

Returns:

Liste von Kundennamen, die den Suchttext beinhalten

GetCustomers

```
public IEnumerable<CustomerDTO> GetCustomers()
```

Im Unterschied zur vorigen Funktion liefert diese Funktion eine Liste von Kundenobjekten zurück. Weiters werden hier alle Kunden zurückgegeben. Es kann also kein Suchtext übergegeben werden. In den Prototypen wird die Funktion beispielsweise für Dropdowns verwendet, bei denen einer der Kunden ausgewählt wird.

Returns:

Liste mit allen gespeicherten Kundenobjekten

EmployeeController

System.Object

System.Web.Http.ApiController

CustomerController

GetEmployees

```
public IEnumerable<EmployeeDTO> GetEmployees()
```

Die Funktion GetEmployees hat die Aufgabe, alle Mitarbeiter des Unternehmens in einer Liste bereitzustellen.

Returns:

Liste aller Mitarbeiter in Form von Objekten

5.4.6.2 Verwendete Toolkits

Zur Erleichterung von bestimmten Operationen werden externe Toolkits verwendet.

Beim Mock-Webservice wurde zur Umwandlung von Instanzen von Modellklassen (z. B. Activity) in entsprechende Data Transfer Objects (z. B. ActivityDTO) ein Toolkit „Automapper“ zur Hilfe herangezogen.

Automapper 3.2.1

Verfasser:

Jimmy Bogard

Beschreibung:

Automapper ist eine Library zur Lösung des aufwändigen Objekt-Mapping-Problems. Üblicherweise werden die Mappings händisch implementiert, wodurch eine Menge Code anfällt. Das Automapper erspart einem diesen Aufwand und ermöglicht einem, Objekt-Mappings in wenigen Zeilen Code zu realisieren.

Abbildung 60: Automapper

6 Qualitätssicherung

Im Rahmen der Qualitätssicherung wird auf die korrekte Darstellung der Benutzeroberfläche sowie auf die Zuverlässigkeit am größter Wert gelegt. Benutzerfreundlichkeit und Effizienz sind ebenfalls maßgebliche Kriterien.

Die Reaktion auf den Klick eines Benutzer soll schnell erfolgen (kurze Wartezeiten), besonders bei oft genutzten Funktionsaufrufen.

Weiters sollen bei der Bedienung der Applikation zwischen den verschiedenen Plattformen gar keine oder wenige Unterschiede sein.

Als wichtigste Grundlage für die Qualitätssicherung sind vom Projektteam regelmäßig Tests durchzuführen, damit die Korrektheit der Software sichergestellt wird. Mit dem Auftraggeber vereinbarte Funktionalitäten werden nicht ohne vorheriges Testen präsentiert. Diese Maßnahme garantiert folglich, dass im fertigen Prototyp keine oder nur wenige Fehler gesucht werden müssen, da regelmäßige Testung und Kontrolle viel späteren Aufwand erspart.

6.1 Einflussgrößen

Folgende Faktoren wurden als Einflussgrößen auf die Qualität identifiziert:

Abbildung 61: Einflussgrößen

Sozial • Zeitdruck	Organisatorisch • Kommunikation mit dem Auftraggeber
Werkzeuge • Entwicklungsumgebung (Testversionen) • Webservice	Technisch • Aufwändige Technologie • Programmiersprachen (z. B. Embarcadero: Projektteam muss Programmiersprache zuerst erlernen)

6.2 Qualitätsmerkmale

Qualitätsmerkmale sind Produkt- und Dienstleistungsmerkmale, welche für die Kaufentscheidung eines Kunden wichtig sind. Die wichtigsten und relevantesten kundenkritischen Qualitätsmerkmale sind in unserem Fall:

- Gebrauchstauglichkeit
  - Zur Gebrauchstauglichkeit zählt eine problemlose Bedienung und hohe Benutzerfreundlichkeit der Prototypen.
- Funktionstüchtigkeit
  - Eines der wichtigsten Kriterien ist die volle Funktionstüchtigkeit des Ergebnisses.
- Zuverlässigkeit
  - Der Prototyp sollte stabil laufen und es sollten keine unerwarteten Abstürze auftreten.
- Technische Anforderungen
  - Niedrige technische Hardwareanforderungen für das fertige App sind vorteilhaft, um auf mehr Geräten flüssig laufen zu können.
- Servicefreundlichkeit
  - Die Servicefreundlichkeit bezieht sich auf die Serviceleistungen der Technologieanbieter.
- Individualität
  - Wichtig ist die Möglichkeit individuelle GUI-Elemente zu erstellen und nicht nur auf Standardelemente angewiesen zu sein.
- Moderne Technologie
  - Eine moderne Technologie kann länger verwendet werden und sich eher an neue Trends der mobilen Plattformen anpassen.
- Entwicklungskosten
  - Entwicklungskosten gehören nicht im Fall von Auris IT nicht zu den wichtigsten Kriterien, sind aber dennoch nicht zu vernachlässigen.

6.3 Maßnahmen zur Qualitätssicherung

6.3.1 Cloud Storage

Um auf Dokumente und Dateien rund um die Diplomarbeit von überall aus zugreifen zu können, wurde ein Cloud-Speicher eingerichtet. Dieser ermöglicht, dass beide Diplomanden Zugriff auf die aktuellste Version aller Dokumente und Dateien haben.

Als Cloudspeicherdienst wurde Microsoft OneDrive verwendet, welches früher auch unter dem Namen Microsoft Skydrive bekannt war. OneDrive erlaubt einfachen Down- und Upload über das Webinterface oder über einen sogenannten OneDrive Ordner.

6.3.2 Meetings

Seit Beginn der Diplomarbeit wurde regelmäßig Statusmeetings mit dem Auftraggeber abgehalten, bei denen immer die beim letzten Meeting vereinbarten Aufgaben kontrolliert und getestet wurden. Falls Änderungen erwünscht waren, wurden diese zu den Aufgaben bis zum jeweils nächsten Meeting aufgenommen. Dadurch wurden Fehler (Bugs) schon während dem Entwicklungsprozess schnell behoben und die Qualität des Ergebnisses erhöht.

Dadurch gab es kein Risiko, dass eine Lösung entwickelt wird, die eigentlich an der Zielvorstellung des Auftraggebers vorbeigeht, da mindestens einmal im Monat ein solches Meeting mit der Auris IT stattgefunden hat.

6.4 Probleme

Im Laufe des Projekts traten auch einige Probleme auf, die Auswirkungen auf den Projektfluss hatten, wobei dabei der Projekterfolg nicht gefährdet wurde.

Folgende nennenswerte Umstände wurden bewältigt:

- **Performance der Adobe AIR App**

  Während die grafische Anzeigequalität der Entwicklungstechnologie Adobe AIR die meisten anderen Technologien übertraf, litt der Prototyp unter Performanceproblemen. Die Reaktion auf Benutzereingaben sowie das Scrollen bei Listen erfolgte leicht verzögert, was jedoch ausschlaggebend für die Benutzerfreundlichkeit ist. Aus diesem Grund wurde hier viel Zeit investiert, um Performanceverbesserungen einzupflegen. Die Performance konnte dadurch leicht gesteigert werden. Zum Projektabschluss konnte die Performance dennoch nicht auf die von üblich gebräuchlichen Apps gesteigert werden.

- **Eingeschränkte Nutzung der Entwicklungsumgebungen**

  Die meisten plattformübergreifenden Entwicklungstechnologien sind kostenpflichtig. Ausnahme bei diesem Projekt stellen PhoneGap und Eclipse Scout dar.

Im Falle von Embarcadero und Adobe AIR konnten nur Testversionen genutzt werden, wodurch die Prototypentwicklung zeitlich begrenzt war.

6.5 Risikomanagement

Risiko	Eintrittswahrscheinlichkeit	Auswirkungsgrad	Maßnahmen	Verantwortlichkeit	Terminsituation
Ressourcen stehen nicht zur Verfügung (Rechner, Testgeräte, …)	10%	gering projektverzögernd	Ankauf der nötigen Geräte	entweder Auftraggeber oder Projektteam (je nach Vereinbarung)	bis Oktober 2014
Das Pflichtenheft ist nicht stabil (Anforderungen ändern sich ständig)	15%	projektverzögernd	Endgültige Festlegung von Zielen und Einigung zwischen Auftraggeber und Projektteam	Auftraggeber und Projektteam	bis Oktober 2014
Die Machbarkeit der Technologien ändert sich schwerwiegend	20%	gefährdet Projekterfolg	Experten aufsuchen, die die Technologien beherrschen; genauere Dokumentationen aufsuchen (z. B. Bücher kaufen); Reduzierung des funktionellen Umfangs	Projektteam	bis Dezember 2014
hoher Zeitdruck/zu großer Projektumfang	15%	projektverzögernd	Vereinbarung mit Auftraggeber, um einen technologischen Kandidaten auszulassen	Projektteam	bis Jänner 2015

Tabelle 16: Risikomanagement

Legende

Risiko…Beschreibung des Risikos

Eintrittswahrscheinlichkeit…Prozentangabe; Wahrscheinlichkeit, mit der das Risiko eintritt

Auswirkungsgrad…gibt an, wie schwer die Folgen bei Risikoeintritt sind (gering projektverzögernd, projektverzögernd, gefährdet den Projekterfolg)

Maßnahmen…einzuleitende bzw. bereits eingeleitete Maßnahmen, um dem Risiko entgegenzuwirken

Verantwortlichkeit…wer muss die Maßnahmen einleiten

Terminsituation…bis wann müssen die Maßnahmen eingeleitet werden

7 Offlinefähigkeitskonzept und Release-Verwaltung

In Ausblick auf zukünftige Weiterentwicklung der Prototypen erarbeitete das Projektteam auch ein Konzept zur Offlinefähigkeit von TeamPlay. Durch Umsetzung dieses Konzepts soll eine Offline-Verwendung der Applikation möglich sein. Im Hintergrund erfolgt bei wiederkehrender Internetverbindung eine Synchronisation mit den Daten des Backends. Aufgetretene Synchronisationskonflikte mit anderen Benutzern werden angezeigt und können vom Benutzer gelöst werden.

Den zweiten Teil der Konzeptausarbeitung stellt das Thema der Release-Verwaltung dar. Das Thema wird in der Praxis oft vergessen und sorgt für Probleme, sobald bei einer Softwarelösung neue Releases, also Änderungen bzw. Erweiterungen, freigegeben werden. Daher wurde ein Konzept angefertigt, womit Softwarelösungen mögliche Release-Wechsel reibungslos überstehen.

7.1 Datenstrukturen (TeamPlay-spezifisch)

Für die Anwendung TeamPlay konkret, wird die Offlinefähigkeit auf bestimmte Arten von Daten ermöglicht. Die Synchronisation wird also nicht auf alle Datenelemente angewendet. Die Datenstrukturen, welche von den Benutzern offline bearbeitet werden können, sind

- **Aktivität (Activity)**
  - Eine Aktivität repräsentiert eine Aufgabe oder ein Projekt, welches von einem Mitarbeiter gebucht werden kann. In einer Aktivität sind verschiedene Mitarbeiter involviert, wobei einer dieser Mitarbeiter hauptverantwortlich für die Aktivität ist.
  - Eine Aktivität wird als Projekt bezeichnet, wenn sie keine übergeordneten Aktivitäten aufweist.
  - Somit ist jede Aktivität, die eine übergeordnete Aktivität besitzt, ein Subprojekt oder eine konkrete Aufgabe.
- **Buchung (Booking)**
  - Eine Buchung wird dann vorgenommen, wenn ein Mitarbeiter Arbeitszeit in eine gewisse Aktivität investiert. Ein Mitarbeiter kann immer nur eine Aktivität buchen, er kann nicht an mehreren Aktivitäten gleichzeitig arbeiten. Weiters dürfen nur in der Aktivität involvierte Mitarbeiter eine Buchung dieser vornehmen.

In der Datenbank ist darüber hinaus zu beachten, dass die beschriebenen Datenstrukturen jeweils einer Versionsverwaltung (=>Versionsnummer) unterliegen, damit ein Synchronisationsverfahren überhaupt anwendbar ist.

7.2 Handhabung einer Web-Anfrage

Die Client-Applikation arbeitet während der aktiven Verwendung und Interaktion des Benutzers stets mit dem Backend zusammen. Das erfordert demnach das ständige Vorhandensein einer Internetverbindung. Damit die Verwendung des Systems offline möglich ist, muss jede einzelne vom Benutzer erzeugte Anfrage zum Backend speziell betrachtet und gehandhabt werden.

Das folgende Ablaufdiagramm beschreibt den Aufruf einer Aktion am Backend und visualisiert die möglichen Ausgänge. Der Begriff „Queue“ weist dabei auf die im nächsten Kapitel bearbeitete „Workitem-Queue“ hin.

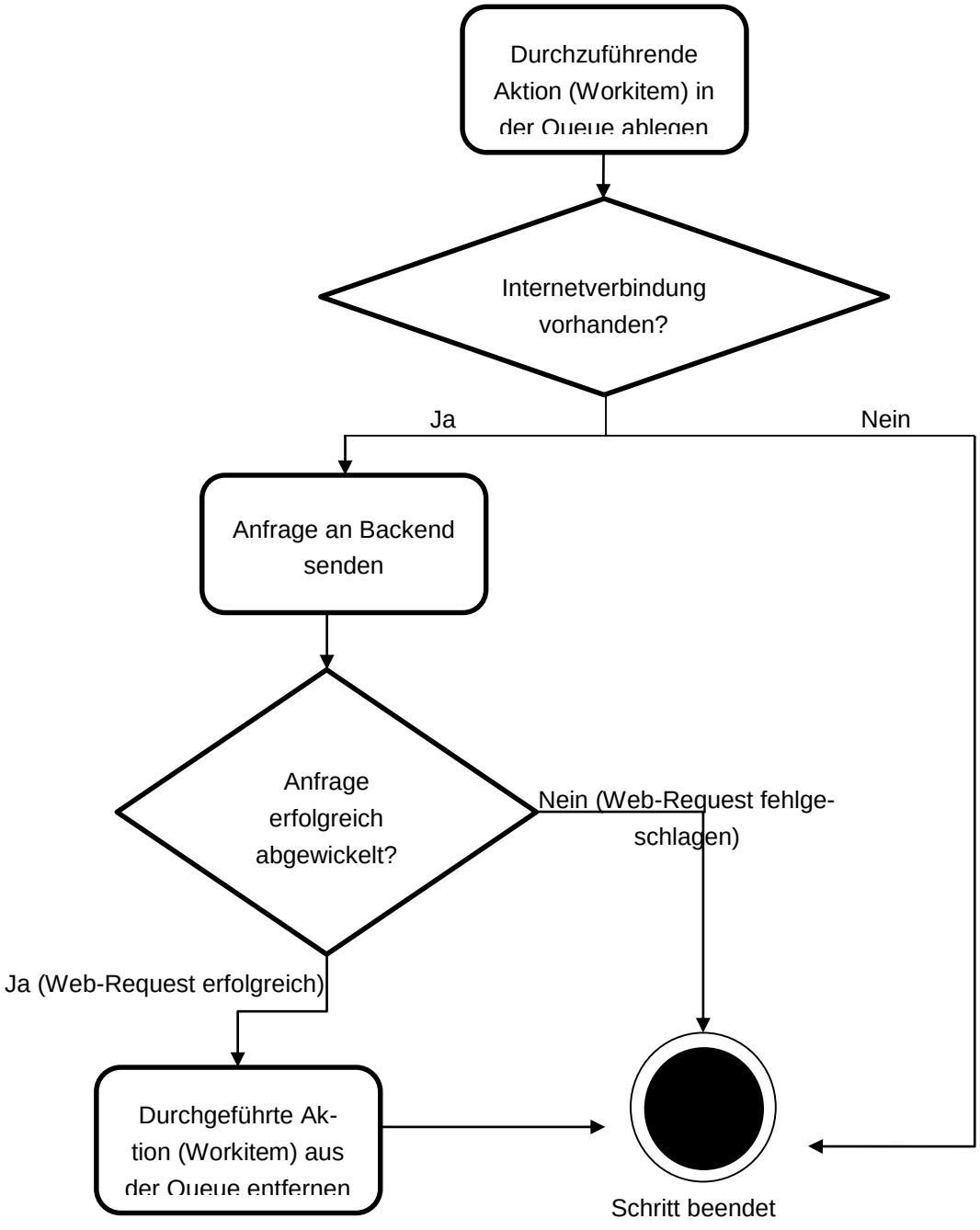


Abbildung 62: Handhabung einer Webanfrage

1. Durchzuführende Aktion (Workitem) in der Queue ablegen

   Zu Beginn wird die Aktion sofort in die Workitem-Queue abgelegt. Daraufhin wird die gesamte Queue am Gerät gespeichert, damit die Information über die durchzuführende Aktion nicht verloren geht. Der Grund für die vorzeitige Speicherung eines Workitems in der Queue ist, dass, wenn bei einem der folgenden Schritte die Client-Applikation unerwartet abstürzt, beendet wird oder sonstige Fehler aufweist, welche die nächsten Schritte verhindern, das Workitem auf jeden Fall gespeichert ist und somit bei der nächstmöglichen Gelegenheit die Ausführung des Workitems erneut versucht werden kann. Auf diesem Weg wird garantiert, dass jede vom Benutzer verlangte Aktion zum jetzigen oder zu einem späteren Zeitpunkt ausgeführt bzw. rekonstruiert werden kann.

2. Abfrage: Internetverbindung vorhanden?
   a. Ja

      Besteht zum Zeitpunkt, zu der die vom Benutzer verursachte Web-Operation aufgerufen werden soll, eine Verbindung zum Internet, wird sofort mit der Durchführung der Web-Anfrage fortgefahren; siehe Schritt 3.

   b. Nein

      Ist keine Internetverbindung vorhanden, kann die Web-Anfrage zu diesem Zeitpunkt nicht durchgeführt werden. Da das eben fehlgeschlagene Workitem bereits in Schritt 1 vermerkt wurde, muss die Aktion zu einem späteren Zeitpunkt durchgeführt werden.

3. Anfrage an Backend senden

   In diesem Schritt wird die gewünschte Aktion tatsächlich über das Internet an das Backend gesendet, das für die weitere Verarbeitung zuständig ist.

4. Abfrage: Anfrage erfolgreich abgewickelt?

   Nachdem der Client vom Server eine Antwort erhalten hat, wird überprüft, ob der Server (=Backend) den Aufruf richtig verarbeitet hat. Das Backend vermerkt dazu den Ausgang der Aktion im dreistelligen Status-Code des HTTP-Headers. Nur wenn dieser ein Muster von „2xx“ aufweist, darf der Client von einer erfolgreichen Verarbeitung ausgehen. Status-Codes, die mit „2“ beginnen weißen darauf hin, dass die Anfrage erfolgreich verarbeitet wurde.

   a. Ja

      Im Erfolgsfall wird Schritt 5 ausgeführt.

   b. Nein

      Ist die Anfrage zum Backend nicht erfolgreich verlaufen, muss die Aktion abgebrochen werden, da der Server derzeit die Anfrage aus welchem

Grund auch immer nicht korrekt verarbeiten konnte. Da das eben fehlgeschlagene Workitem bereits in Schritt 1 vermerkt wurde, muss die Aktion zu einem späteren Zeitpunkt durchgeführt werden.

5. Durchgeführte Aktion (Workitem) aus der Queue entfernen

   Wenn das Backend die Operation erfolgreich abgeschlossen hat, wird das Workitem aus der Workitem-Queue entfernt, weil nur nicht-erfolgreiche Workitems in der Queue abgelegt sind.

7.3 Die Workitem-Queue

7.3.1 Grundprinzip

Da am Client nicht immer alle Web-Anfragen erfolgreich ablaufen und vor allem mobile Endgeräte nicht dauerhaft eine ausreichend starke Verbindung zum Internet haben, muss sich der Client fehlgeschlagene Operationen „merken“, damit die Eigenschaft der Offline-Fähigkeit erreicht wird. Den Vorgang des „Merkens“ erreicht man durch die Erzeugung eines Workitems (siehe Spezifikation Workitem). Dieses Workitem repräsentiert die Aktion, die der Client ausführen möchte. Das Workitem wird anschließend in einer Liste bzw. einer Queue – nämlich der Workitem-Queue – abgelegt. Wie ein Workitem in die Queue gelangt (und möglicherweise dort vorerst verbleibt), ist dem Abschnitt 7.2 zu entnehmen.

In der Client-App steht die Workitem-Queue in einem globalen Kontext stets zur Verfügung. Da diese Queue am Gerät dauerhaft gespeichert wird, muss sie beim Start der App in den Arbeitsspeicher des Client-Geräts geladen werden. Damit eine Speicherung überhaupt möglich ist, muss das Workitem so gestaltet werden, dass es serialisierbar ist.

7.3.2 Spezifikation Workitem

Ein Element in der Workitem-Queue wird als Workitem bezeichnet. Das Workitem wird durch die Definition einer Klasse in der jeweiligen verwendeten Programmiersprache am Client definiert und wird durch nachstehende Felder spezifiziert:

Operation Type enum (CREATE, UPDATE, DELETE)	Object type enum (ACTIVITY, BOOKING)	Data Object

Tabelle 17: Workitem

Diese Form wird gewählt, damit die Queue auch am Gerät persistiert werden kann. Für die Speicherung werden die Daten beispielsweise in das JSON-Format serialisiert, wobei die Form der Serialisierung keine ausschlaggebende Rolle spielt.

Entscheidend für ein Workitem ist also

- welche Modifikation (Operation Type)
- für welche Art von Daten (Object type)
- mit welchen Eigenschaften (Data)

vorgenommen wurde.

7.3.2.1 Art der Modifikation (Operation Type)

Die Art der Modifikation gibt an, welche Operation später mit dem dazugehörigen Objekt erfolgen muss.

Folgende drei Modifikationen sind erlaubt:

- CREATE (Anlegen)
- UPDATE (Aktualisieren)
- DELETE (Löschen)

7.3.2.2 Objekttyp (Object type)

Das Feld „Objekttyp“ gibt mithilfe einer Enumeration an, welche Art von Objekt im Workitem gespeichert wird. Die Spezifikation des Objekttyps dient der Erleichterung bei der Deserialisierung der am Gerät gespeicherten Workitem-Queue.

Folgende Objekttypen werden für „TeamPlay“ definiert (Enumeration):

- ACTIVITY
- BOOKING

7.3.2.3 Eigenschaften (Data)

Im Feld „Eigenschaften“ bzw. „Data“ ist das betroffene Objekt selbst enthalten. Damit ist neben der Art des Objektes und der Operation (Objekttyp, Art der Modifikation) bekannt, welches Objekt konkret betroffen ist. Es ist zu beachten, dass diese Objektinstanz selbst bzw. schon die Definition der dazugehörigen Klasse serialisierbar ist für JSON, XML oder andere Serialisierungsverfahren.

7.3.2.4 Beispiel

UPDATE	ACTIVITY	{
		Name: „Projekt XYZ“,
		Aktiver Mitarbeiter: „SCDA“,
		…
		Versionsnummer: xy
		}

Tabelle 18: Beispiel-Workitem

In diesem Beispiel führt der Benutzer eine Aktualisierung auf eine bestimmte Aktivität aus. Die Klasse „Activity“ hat dabei die Version des Releases Nr. 3. Das Objekt selbst besitzt ebenfalls eine Versionsnummer, welche in späterer Folge zur Konflikterkennung verwendet wird.

7.3.3 Regeln

Für Offline-Veränderungen gelten grundlegende logische Vorschriften, die für eine effiziente und sinnhafte Speicherung eingehalten werden müssen:

1. Wird auf ein Objekt öfters als einmal offline eine Änderung (UPDATE) getätigt, so wird der jeweils bereits vorhandene UPDATE-Eintrag mit den neuen Werten überschrieben (z. B. verändert der User mehrmals hintereinander die Eigenschaftswerte eines Objektes). Dabei geht keine Information von vorherigen Änderungen verloren.
2. Ähnlich ist es mit dem Fall, dass ein neues Objekt offline angelegt wird (CREATE) und dieses im Nachhinein (immer noch offline) verändert wird (UPDATE) (kann auch mehrmals verändert werden). Für die jeweilige Veränderung wird kein neuer Eintrag erzeugt, sondern die Daten im CREATE-Eintrag entsprechend überschrieben.
3. Weiters kann auf zum Löschen markierte Daten (DELETE) keine Bearbeitung mehr durchgeführt werden.
4. Wird ein Objekt, welches offline angelegt wurde (CREATE) und sich nach wie vor in der Queue befindet, zum Löschen markiert (DELETE), so verschwindet das Workitem aus der Workitem-Queue.

7.3.4 Mapping-System

Zu Beginn des Client-Applikationsstarts werden **Mappings** konfiguriert. Mit diesen Mappings wird definiert, welcher **Objekttyp** mit einer bestimmten **Art von Operation** welche Funktion aufrufen muss, damit die richtige Aktion ausgeführt wird. Die Mappings werden anschließend in einer Liste verwaltet. Über die Mappings-Konfiguration kann ein gegebenes Workitem einem konkreten Funktionsaufruf zugeordnet werden.

Zum Beispiel wird eingerichtet, dass eine Instanz der Klasse **Activity** mit der Operationsart **UPDATE** die **UpdateActivity(Activity a)-**Methode (das gespeichert Objekt wird dabei in diese Methode als Parameter übergeben) aufrufen muss.

7.3.4.1 Darstellung der Mappings-Liste

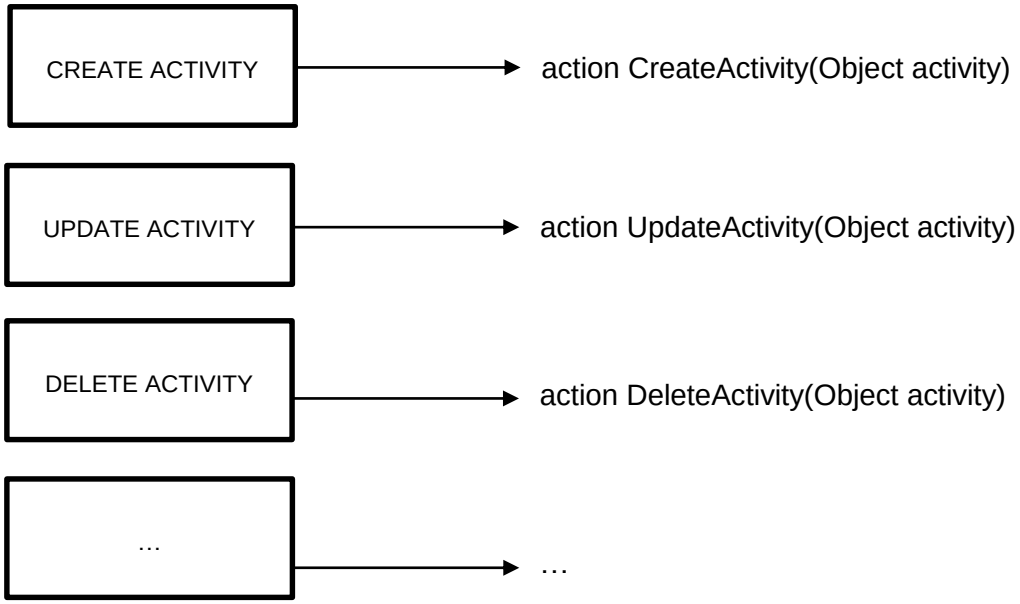


Abbildung 63: Beispiel-Mappings-Liste

Die Mappings-Liste enthält alle Kombinationen der zu behandelnden Objekttypen (CREATE, UPDATE, DELETE → jeweils auf ACTIVITY, BOOKING).

7.3.4.2 Vorteil der Mappings-Konfiguration

Der Grund für Einführung einer Mappings-Konfiguration liegt auf der Hand: damit können beliebige Objekttypen und Operationsarten mit beliebigen Funktionen verknüpft werden. Jene Mappings, die im Programm notwendig sind, werden beim Start der Applikation konfiguriert. Die Festlegung, welche Objekttyp-Operationsart-Kombinationen gewählt werden, erfolgt an einer zentralen Stelle beim Applikationsstart. In späterer Folge greift das Programm lediglich auf die Mappings-Konfiguration zu und versorgt sie mit Workitems. Der Vorteil ist, dass ein Workitem mittels einem einzigen Befehl der Mappings-Konfiguration übergeben werden kann. Würde keine Mappings-Konfiguration implementiert werden, so würden bei jenen Programmstellen, wo ein Workitem verarbeitet werden soll, mehre Abfragen bezüglich des Objekttyps und der Operationsart notwendig sein.

Unter Verwendung der Mappings-Konfiguration jedoch wird dieser Aufwand erspart und die Konfiguration verarbeitet transparent die Workitems.

7.3.5 Abarbeiten der Queue

7.3.5.1 Vorgehensweise

Während der Nutzung der App wird im Hintergrund versucht, die offline getätigten Änderungen zum Backend zu senden, indem die Queue Element für Element abgearbeitet wird.

CREATE	ACTIVITY	{ … }
UPDATE	BOOKING	{ … }
DELETE	ACTIVITY	{ … }
CREATE	BOOKING	{ … }
…	…	…

Tabelle 19: Beispiel-Queue für noch zu erledigende Aktionen

Wird nun die Queue durchlaufen, wird zu einem Eintrag in der Mappings-Konfiguration die richtige Funktion in der Mappings-Liste gesucht. Die Funktion wird anschließend aufgerufen.

Beispielsweise wird gerade folgender Queue-Eintrag bearbeitet:

CREATE	**ACTIVITY**	{
		Name: „Projekt XYZ“,
		Aktiver Mitarbeiter: „SCDA“,
		…
		}

Tabelle 20: Behandelter Beispiel-Mappings-Eintrag

Demnach muss der Eintrag CREATE ACTIVITY in der **Mappings**-Liste gesucht werden:

…	…	…
CREATE	ACTIVITY	action CreateActivity(Object activity)
UPDATE	ACTIVITY	action UpdateActivity(Object activity)
…	…	…
…	…	…

Tabelle 21: Suche in der Mappings-Liste

Ist der Eintrag gefunden, wird die dazugehörige Funktion aufgerufen. Dabei wird das Objekt als Parameter an die Action übergeben. In der Action selbst erfolgt anschließend der konkrete Aufruf des HTTP-Calls an das Backend.

Wenn der Aufruf erfolgreich ist und das Backend die Anfrage verarbeiten konnte, wird der Eintrag aus der Queue entfernt.

Bei Misserfolg wird der Grund für das Fehlschlagen der Anfrage analysiert werden. Retourniert das Backend eine Meldung, dass ein Konflikt (siehe Konflikterkennung) aufgetreten ist, muss der Eintrag speziell als Konflikt-Objekt markiert werden, damit der Konflikt später vom Benutzer gelöst werden kann.

7.3.5.2 Zeitpunkt der Abarbeitung der Queue

Um möglichst ressourcensparend vorzugehen, was besonders bei mobilen Applikationen mit stark begrenzter Akku-Lebensdauer wichtig ist, wird eine intelligente und schonende Strategie für den Abarbeitungszeitpunkt gewählt.

Voraussetzung dafür ist, dass die Programmiersprache am Client über die Funktionalität verfügt, auf die Änderung des Internet-Empfangs zu reagieren. Das erfolgt mit Ereignissen (Events), welche eben bei der Änderung des Internet-Empfangs auftreten. Die Ereignisse können im Programm anschließend mit Event-Handler abgefangen und behandelt werden.

Weiters wird ein Timer zur Hilfe herangezogen. Der Timer wird passend „Sync-Timer“ genannt, da er die Aufgabe hat, bei bestehender Internetverbindung in regelmäßigen Abständen zu versuchen, die Workitem-Queue mit dem Backend abzugleichen. Damit werden Client und Server auf den neuesten Stand miteinander synchronisiert. Tritt das Ereignis der Änderung der Internetverbindung ein, wird der Sync-Timer bei vorhandener Internetverbindung gestartet und im Falle des Verbindungsverlusts wird der Timer wieder gestoppt.

Folgende Grafik zeigt die genaue Strategie bei Änderung des Internet-Empfangs:

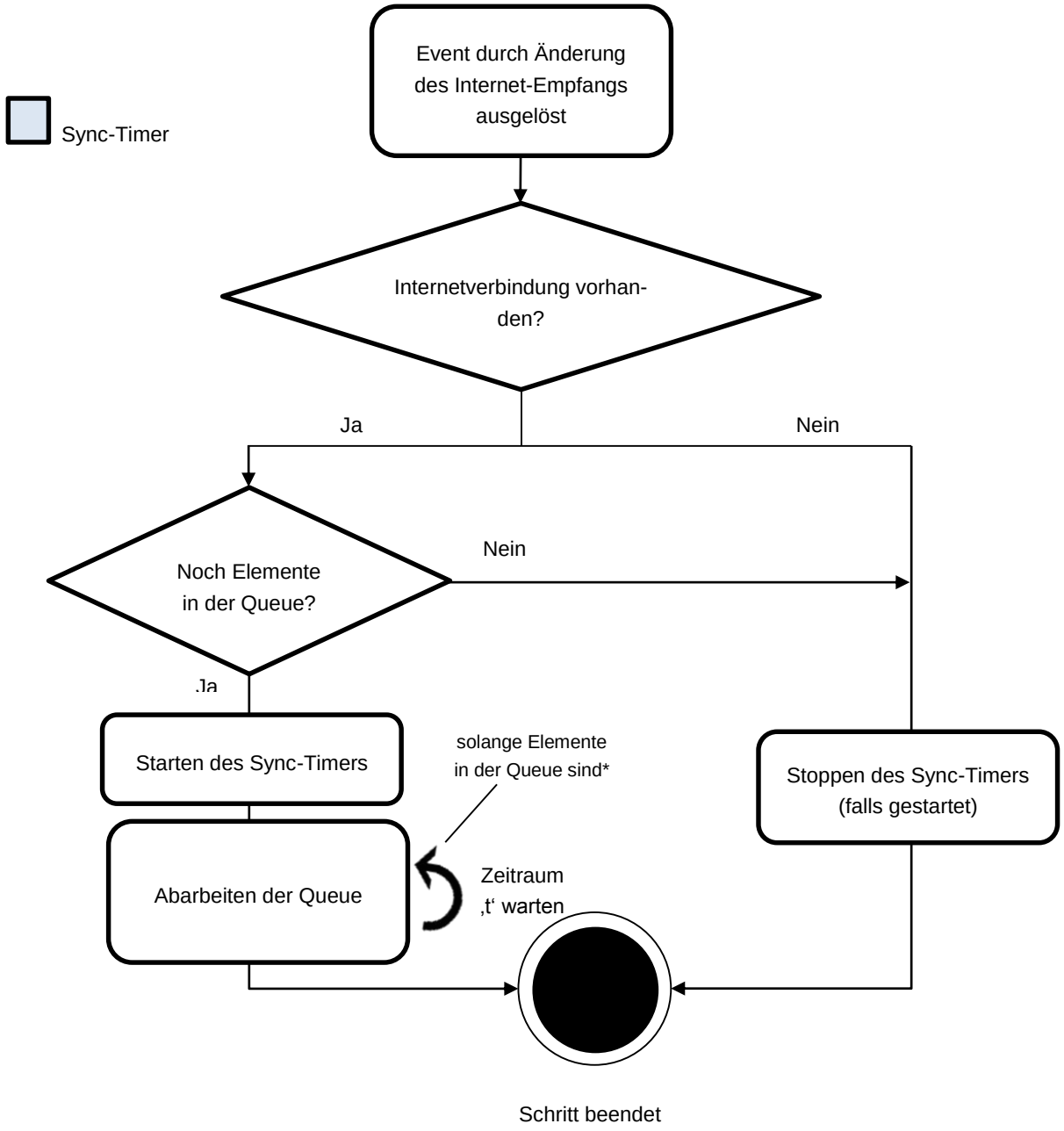


Abbildung 64: Synchronisationsstrategie

**Elemente bleiben dann in der Queue, wenn der Synchronisationsversuch des Elements abermals fehlschlägt oder ein möglicherweise damit verbundener Konflikt vom Benutzer noch nicht aufgelöst wurde*

7.4 Konflikterkennung & Konfliktlösung

Die Konflikterkennung und die Konfliktlösung beim Synchronisationskonzept können entweder auf der Client-Seite oder auf der Server-Seite erfolgen. Dabei gibt es für beide Varianten unterschiedliche Ansätze, welche Vor- und Nachteile mit sich bringen.

7.4.1 Verfahren zur Konflikterkennung

Die prinzipielle Vorgehensweise zur Aufspürung von Konflikten ist unabhängig davon, ob die Erkennung am Server oder am Client erfolgt, gleich anzuwenden.

Das anzuwendende Verfahren der Konflikterkennung hängt von der vorgenommenen Operation ab (CREATE; UPDATE; DELETE).

7.4.1.1 Verfahren bei CREATE

Im Falle von CREATE existiert sinngemäß **keine Konflikterkennung**, da die vorliegenden Daten erst auf der Client-Seite existieren aber noch nicht am Server gespeichert wurden.

7.4.1.2 Verfahren bei UPDATE

Wurde ein bereits auf dem Server existierendes Objekt offline auf einem Client verändert, so sind folgende Möglichkeiten zu bedenken:

7.4.1.2.1 Das Server-Objekt wurde inzwischen gelöscht

Wurde das Objekt am Server während der Offline-Phase des Clients gelöscht, liegt auf jeden Fall ein **Konflikt** vor.

7.4.1.2.2 Das Server-Objekt existiert nach wie vor

Im üblichen Fall ist ein betreffendes Objekt immer noch am Server vorhanden. In diesem Fall werden die Versionsnummern des Server-Objektes und des offline veränderten Client-Objekts verglichen.

Nun sind wieder zwei mögliche Ausgänge zu beachten:

7.4.1.2.2.1 Der Server besitzt eine neuere Version des Objektes

In diesem Fall wurde das Objekt bereits von anderen Clients während des Offline-Zustands verändert und auf eine neue Version (Erhöhung der Versionsnummer) gebracht. Ein Konflikt ist nun schon sehr wahrscheinlich. Ein Konflikt liegt jedoch erst dann vor, wenn der Client Werte von Attributen verändern möchte, welche während seiner Offline-Phase schon von anderen Clients verändert wurden. Das setzt voraus, dass am Server die Attributswerte des Objektes zu jeder Version gespeichert werden. Damit erhält man einen Verlauf aller früheren Versionen mit ihren dazugehörigen Attributswerten, welche bei der jeweiligen Version vorherrschten. Erst dann kann festgestellt werden, ob tatsächlich ein Konflikt aufgetreten ist.

Bei der Überprüfung wird also zuerst festgestellt, welche Attribute sich von der Version vom Client-Objekt auf die aktuelle Version geändert haben. Diese Informationen sind auf dem Server gespeichert, da dieser alle Versionen des Objekts kennt. Alle Attributswerte, die sich von der Client-Objekt-Version auf die aktuelle Version nicht verändert haben, dürfen vom Client-Objekt zweifelsfrei überschrieben werden, da niemand sonst während der Offline-Phase des Clients den Wert des Attributs verändert hat.

Hat sich ein Attribut von der vom Client angegebenen Version auf die neueste Version verändert und entspricht der vom Client gelieferte Attributswert nicht jenem Attributswert, welchen der Server bei der Client-Objekt-Version gespeichert hat, ist ein **Konflikt** aufgetreten. In einfachen Worten heißt das: Der Client hat während seiner Offline-Phase Attributswerte des Objekts verändert, welche in der Zwischenzeit auch von einen oder mehreren Clients auf einen anderen Wert geändert wurden.

7.4.1.2.2.2 Der Server besitzt die gleiche Version des Objektes

Stimmen die Versionsnummern von Client-Objekt und Server-Objekt überein, steht fest, dass während der Offline-Phase des Clients keinerlei Veränderungsanfragen am Server umgesetzt wurden. Alle Änderungswünsche vom Client können somit übernommen werden.

7.4.1.3 Verfahren bei DELETE

7.4.1.3.1 Das Server-Objekt wurde inzwischen gelöscht

Wurde das Objekt bereits vom Service gelöscht, besteht keine Konfliktsituation. Die vom Client gewünschte Operation wurde bereits von einem anderen Client umgesetzt.

7.4.1.3.2 Das Server-Objekt wurde inzwischen verändert (neue Versionsnummer)

Wurde das Objekt während der Offline-Phase des Clients verändert, so darf die gewünschte Löschung nicht vorgenommen werden, da möglicherweise ein anderer Client wichtige Änderungen getätigt hat und keine Absicht auf Löschung des Objektes hat. Es kommt also hier wiederum zum **Konflikt**.

7.4.1.3.3 Das Server-Objekt wurde inzwischen nicht verändert (gleiche Versionsnummer)

Wurde das Objekt während der Offline-Phase des Clients nicht modifiziert, ist die Löschung wie geplant vorzunehmen.

7.4.1.4 Überblick Konflikterkennung (Diagramm)

Objekt C…Objekt, dass offline vom Client verändert wurde

Objekt S…gleiches, am Server befindliches Objekt, welches stets die aktuellste Version hat

Das folgende Diagramm zeigt die Überprüfung für ein einziges Objekt an. Diese Schritte werden selbstverständlich für alle vom Client offline-geänderten Objekte durchgeführt.

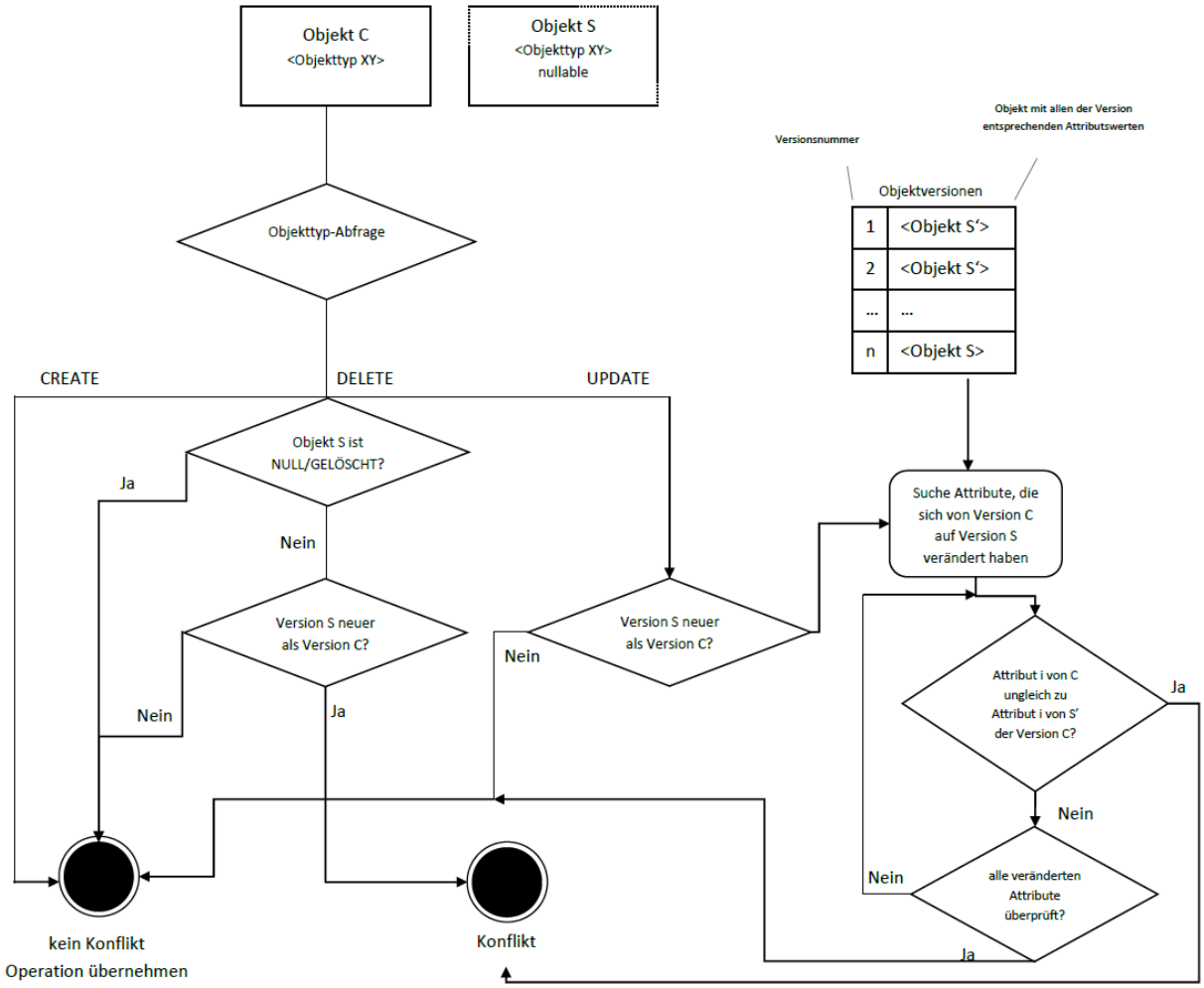


Abbildung 65: Ablauf Konflikterkennung

7.4.2 Zuständigkeiten Konflikterkennung am Server (Backend)

Voraussetzung bei der Konflikterkennung ist, dass der Client stets mitprotokolliert, welche Änderungen offline vorgenommen wurden. Wichtig ist, dass bekannt ist, welche Datensätze betroffen sind, welche Änderungen vorgenommen wurden (Attributswerte) und auch, welche Version das Objekt besitzt (Versionsnummer; zählt zu den Attributswerten). Die Vorgehensweise der Konflikterkennung kann dabei auf folgenden Seiten erfolgen:

7.4.2.1 Übersicht

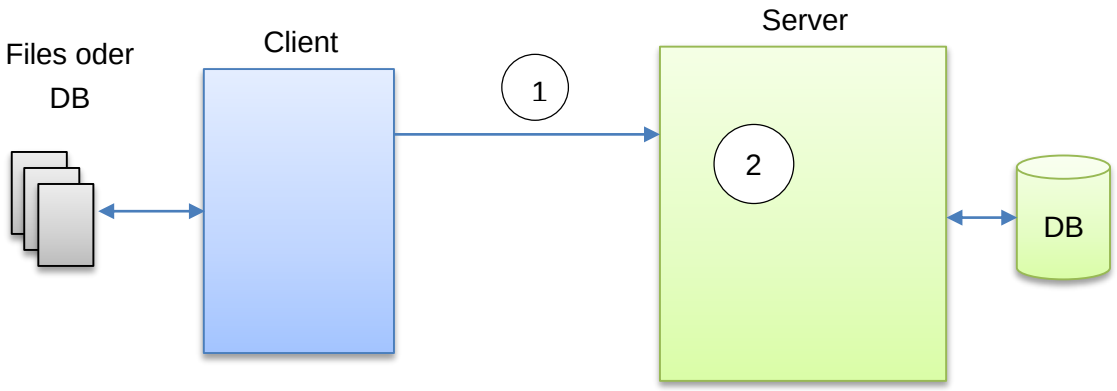


Abbildung 66: Konflikterkennung am Server

1. Der Client schickt alle offline veränderten Datensätze zur Synchronisation an den Server.

2. Abgleichen der empfangenen Daten mit den lokalen Daten am **Server**: Alle Datensätze werden überprüft und auf **Konflikte** untersucht.

7.4.2.2 Vorteile

- zentrale Positionierung der Konflikterkennungslogik auf dem Server
- Konflikterkennungslogik wird **einmal** implementiert, somit können Änderungen rasch durchgeführt werden
- bei Software-Updates sind nicht die Clients sondern nur der Server betroffen
- Server nimmt seine konventionelle Stelle als „Vertrauensposition“ ein (=> Clients vertrauen dem Server)

7.4.2.3 Nachteile

- leicht erhöhte Auslastung durch das Erledigen der Konfliktlösung aller Clients (dieser Nachteil ist minimal und kaum erkennbar)

- Änderung der Konflikterkennungslogik sorgt möglicherweise für kurzfristige Nicht-Verfügbarkeit des Servers

7.4.2.4 Fazit

Die Variante „**Server-Seite (Backend)**“ beschriebene Lösung ist auch in der Praxis die am üblichsten angewendete. Hier überwiegen die Vorteile eines zentralen Backends. Die Lösung auf Seiten des Clients weisen zu viele ausschlaggebende Nachteile auf, weshalb sie in den meisten Fällen nicht angewendet wird.

7.4.3 Konfliktlösung am Client

Bei diesem Lösungsansatz darf der Benutzer entscheiden, wie er Konflikte auflösen möchte (=> welche Version einer Änderung schließlich gewählt wird).

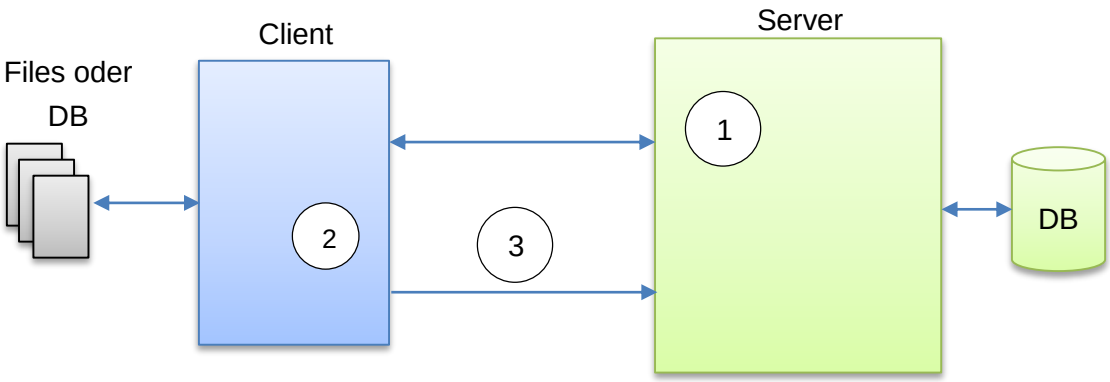


Abbildung 67: Konfliktlösung am Client

1. Konflikterkennung vom Server
2. Auf der Client-Seite entscheidet der Benutzer darüber, wie die Konflikte aufgelöst werden (welche Version der Daten wird nun übernommen). Dazu wird der Benutzer am Client durch einen „Guide“ geführt, der die möglichen **Lösungsvorschläge** darstellt. Die Benutzerentscheidungen werden während der Führung durch den Guide gesammelt.
3. Alle Benutzerentscheidungen werden an den Server gesendet, welcher anschließend die Änderungen so übernimmt. Ausnahme: wurde ein zu einem Konflikt gehörender Datensatz inzwischen erneut verändert, wird der Konflikt als „nicht gelöst“ markiert und muss beim nächsten Konflikterkennungsvorgang erneut behandelt werden.

7.4.3.1 Vorteile

- Benutzer hat Entscheidungsmacht über die Lösung der Konflikte und kann Verlust wichtiger Daten vorbeugen (großer und signifikanter Vorteil gegenüber Server-Lösung)

- Benutzerfreundlichste Lösung, da der Benutzer über die Konflikte benachrichtigt wird und er sich mit dem konfliktverursachenden Benutzer über die Änderungen absprechen kann

7.4.3.2 Nachteile

- Der Benutzer muss immer jeden Konflikt einzeln auflösen und wird durch das Bedienen des „Guides“ von seiner Arbeit gestört
- Konfliktlösungs-„Guide“ am Client aufwändig zu implementieren
- erhöhter Aufwand, wenn mehrere verschiedene Client-Plattformen vorhanden sind
- Änderung des „Guides“ erfordert Anpassung und Updates für alle unterstützten Client-Plattformen
- Konflikterkennungslogik am Server zwingend, weil betroffene Datensätze zwischen dem Zeitpunkt der Benutzerbedienung und dem Senden der Lösung an den Server verändert werden können und somit zu Bestehenbleiben eines aufzulösenden Konflikts führen

7.4.3.3 Fazit

Die Vorgehensweise „**Client-Seite**“ ist dem Ansatz am Backend vorzuziehen. In der Praxis müssen solche Entscheidungen ebenfalls vom Benutzer getroffen werden. Automatische Konfliktlösung vom Backend kann zu unvorhergesehenen und unterwünschten Datenmanipulationen führen. Hier ist es also sinnvoll, die menschliche Intelligenz für eine sinnerfassende Konfliktlösung heranzuziehen.

7.4.4 Konzept Konfliktlösung am Client

Die bevorzugte Konfliktlösungsvariante auf der Client-Seite wird im aktuellen Abschnitt genauer erläutert.

7.4.4.1 Grundidee

Um die Konflikte aufzulösen, wird dem Benutzer eine grafische Oberfläche zur Verfügung gestellt, wo alle Konflikte übersichtlich aufgelistet sind und verwaltet werden können. Die Konzeption wird im Folgenden als „**Konflikt-Manager**“ oder „**Konflikt-Guide**“ bezeichnet.

7.4.4.2 Abzubildende Konflikte

Wie bereits erläutert können Konflikte durch die Operationen

- UPDATE und
- DELETE

auf Aktivitäten bzw. Buchungen auftreten.

Im Konflikt-Manager müssen also verschiedene Fälle beachtet werden. Hat ein Benutzer A offline eine Änderung oder Löschung vorgenommen (UPDATE, DELETE), können in weiterer Folge Konflikte auftreten. Die Konflikte werden dabei im Regelfall vom Backend erkannt, das Client-Gerät hat die Aufgabe, dem Benutzer jene Konflikte zur Konfliktlösung grafisch darzulegen.

Folgende Fälle können somit auftreten:

Gewünschte Aktion (offline)	Mögliche Konflikte
UPDATE	• ein anderer Benutzer hat Änderungen auf dieselben Attribute vorgenommen • ein anderer Benutzer hat das betreffende Objekt gelöscht
DELETE	• ein anderer Benutzer hat das betreffende Objekt in der Zwischenzeit geändert

Tabelle 22: Relevante Konfliktfälle für den Konflikt-Manager

7.4.4.3 GUI-Abbildungen

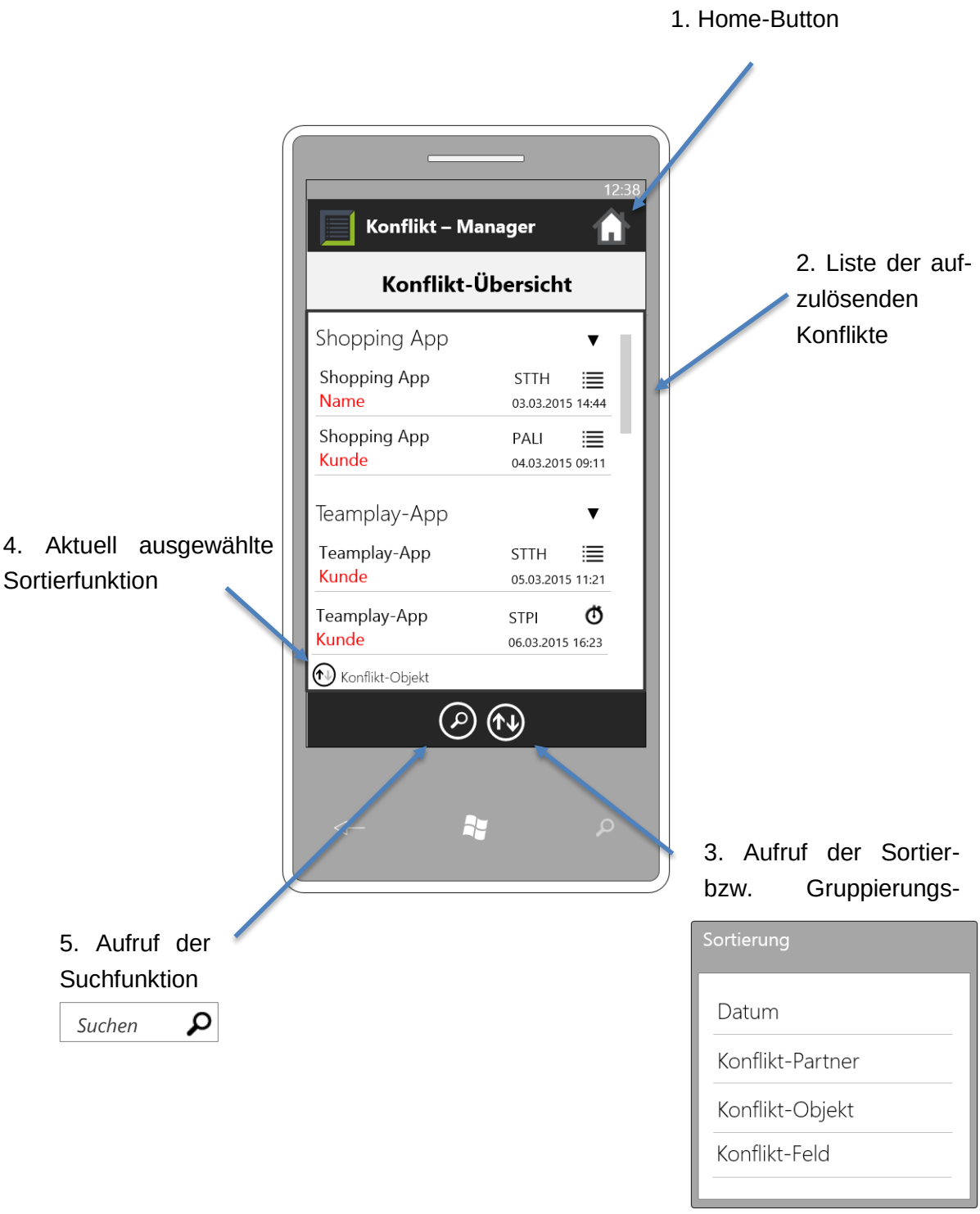
Der Konflikt-Manager bietet dem Benutzer eine übersichtlich aufbereitete Oberfläche zur Auflösung der aufgetretenen Konflikte.

Es muss dabei zwischen den verschiedenen Konfliktarten unterschieden werden, da jede Art eine eigene Herangehensweise bei der Konfliktauflösung des Benutzers verlangt.

Damit das Konzept möglichst nachvollziehbar ist, wurden zur genauen Erläuterung GUI-Abbildungen entworfen. Folgende Ansichten stehen dem Benutzer zur bedienungsfreundlichen Konfliktlösung zur Verfügung:

7.4.4.3.1 Konfliktübersicht

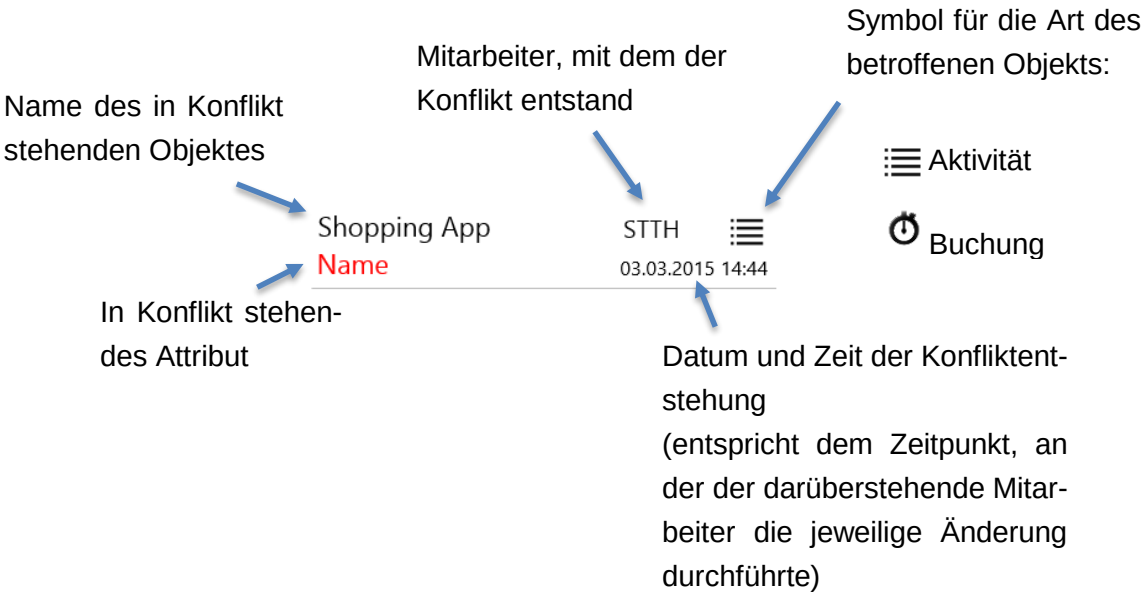
Die Konfliktübersicht ist die erste Ansicht, die dem Benutzer unmittelbar nach dem Aufruf des Konfliktmanagers präsentiert wird. Hier stehen eine Sammlung der zu lösenden Konflikte sowie Sortier- bzw. Gruppierungs-Funktionen sowie eine Suchfunktion zur Verfügung.


1. Home-Button
12:38
Konflikt – Manager
Konflikt-Übersicht
2. Liste der auf-
zulösenden
Konflikte
Shopping App
Shopping App
Name
STTH
03.03.2015 14:44
Shopping App
Kunde
PALI
04.03.2015 09:11
Teamplay-App
Teamplay-App
Kunde
STTH
05.03.2015 11:21
Teamplay-App
Kunde
STPI
06.03.2015 16:23
Konflikt-Objekt
4. Aktuell ausgewählte
Sortierfunktion
3. Aufruf der Sortier-
bzw. Gruppierungs-
Sortierung
Datum
Konflikt-Partner
Konflikt-Objekt
Konflikt-Feld
5. Aufruf der
Suchfunktion
Suchen

1. Home-Button

Der Home-Button dient dazu, dem Konfliktmanager jederzeit beenden zu können und zur Startansicht der Applikation zu navigieren.

2. Listenelement der Konflikt-Liste



3. Aufruf der Sortier- bzw. Gruppierungs-Funktion

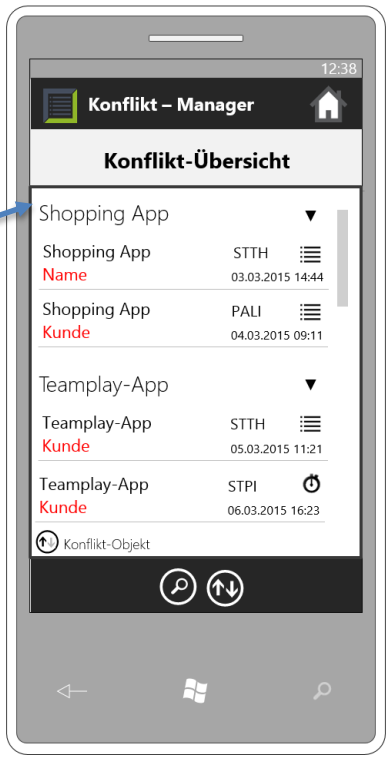
Möchte der Benutzer die Liste sortieren bzw. gruppieren nach bestimmten Kriterien, stehen ihm folgende Möglichkeiten zur Verfügung:

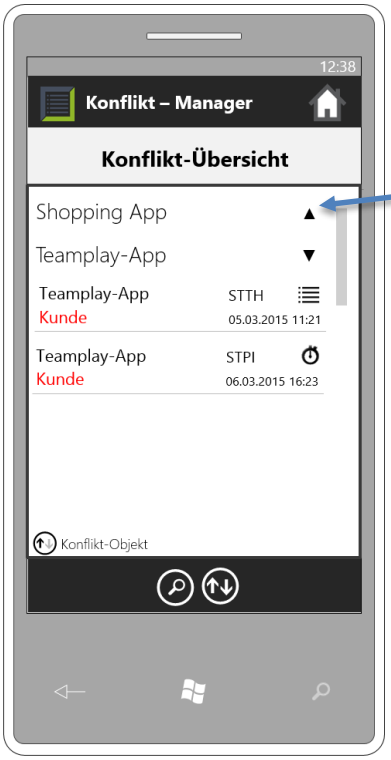
- Sortierung nach Datum
- Gruppierung und Sortierung nach Konflikt-Partner (Mitarbeiter)
- Gruppierung und Sortierung nach Aktivitäts- bzw. Buchungs-Name (Objektname)

Die Gruppierung und Sortierung nach Konflikt-Partner sowie Objektname sehen folgendermaßen aus:

Gruppierung & Sortierung nach **Konflikt-Objekt**

Gruppen-Überschrift (Name der Aktivität bzw. Buchung)

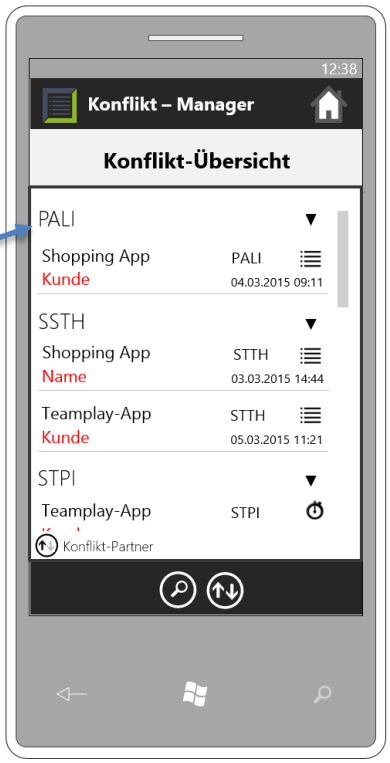


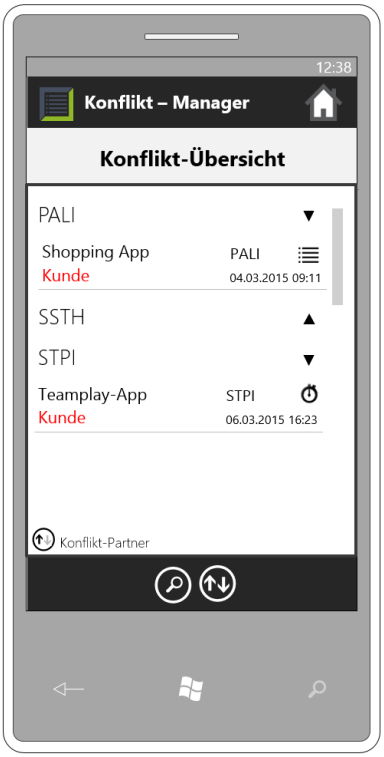


Inhalt einer Gruppe kann auf- und zugeklappt werden

Gruppierung & Sortierung nach **Konflikt-Partner** (Mitarbeiter)

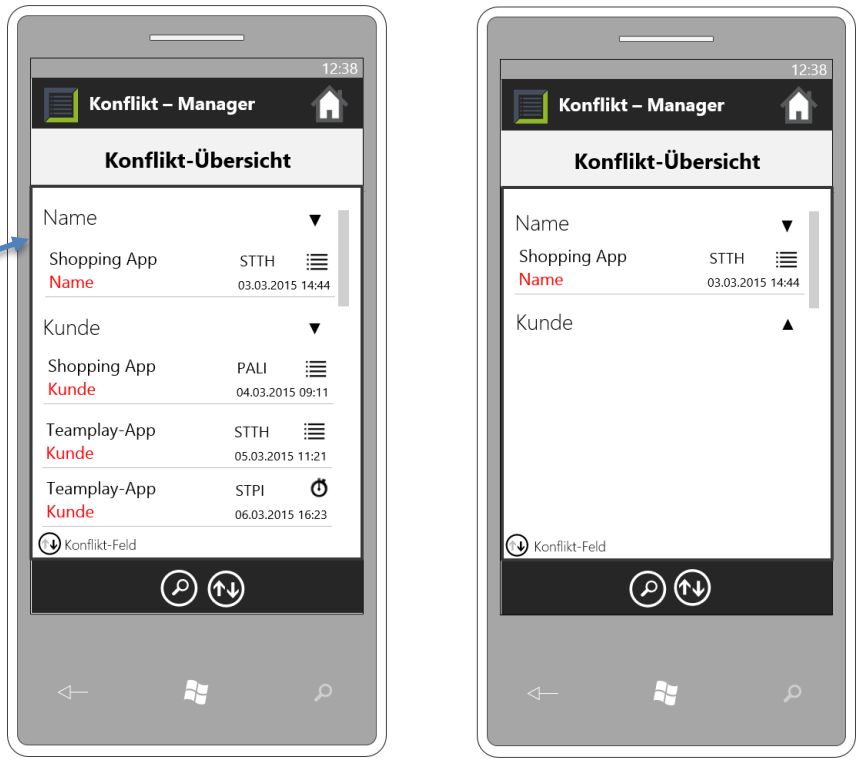
Gruppen-Überschrift (Mitarbeiter-Kürzel)





Gruppierung & Sortierung nach **Konflikt-Feld** (Attribut)

Gruppen-Überschrift (Name des Konflikt-Attributs)



4. Aktuell ausgewählte Sortierfunktion

Im linken unteren Bereich wird eine Schaltfläche angezeigt, mit der die Sortierreihenfolge im aktuellen Such-/Gruppier-Kriterium umgeschaltet werden kann. Ist momentan aufsteigende Reihenfolge einstellt, ist der Pfeil nach oben im Icon schwarz hervorgehoben; bei absteigender Reihenfolge wird der Pfeil nach unten mit schwarz betont.

5. Aufruf der Suchfunktion

Falls der Benutzer gezielt einen Konflikt rasch finden möchte, steht ihm eine Suchfunktion zur Verfügung. Mit dem Klicken des Suchen-Buttons wird im oberen Bereich des Displays eine Sucheingabe geöffnet. Hier kann der Benutzer seine Eingabe tätigen, nach welcher anschließend alle zutreffenden Ergebnisse gesammelt werden.

Die Kriterien der Suche umfassen

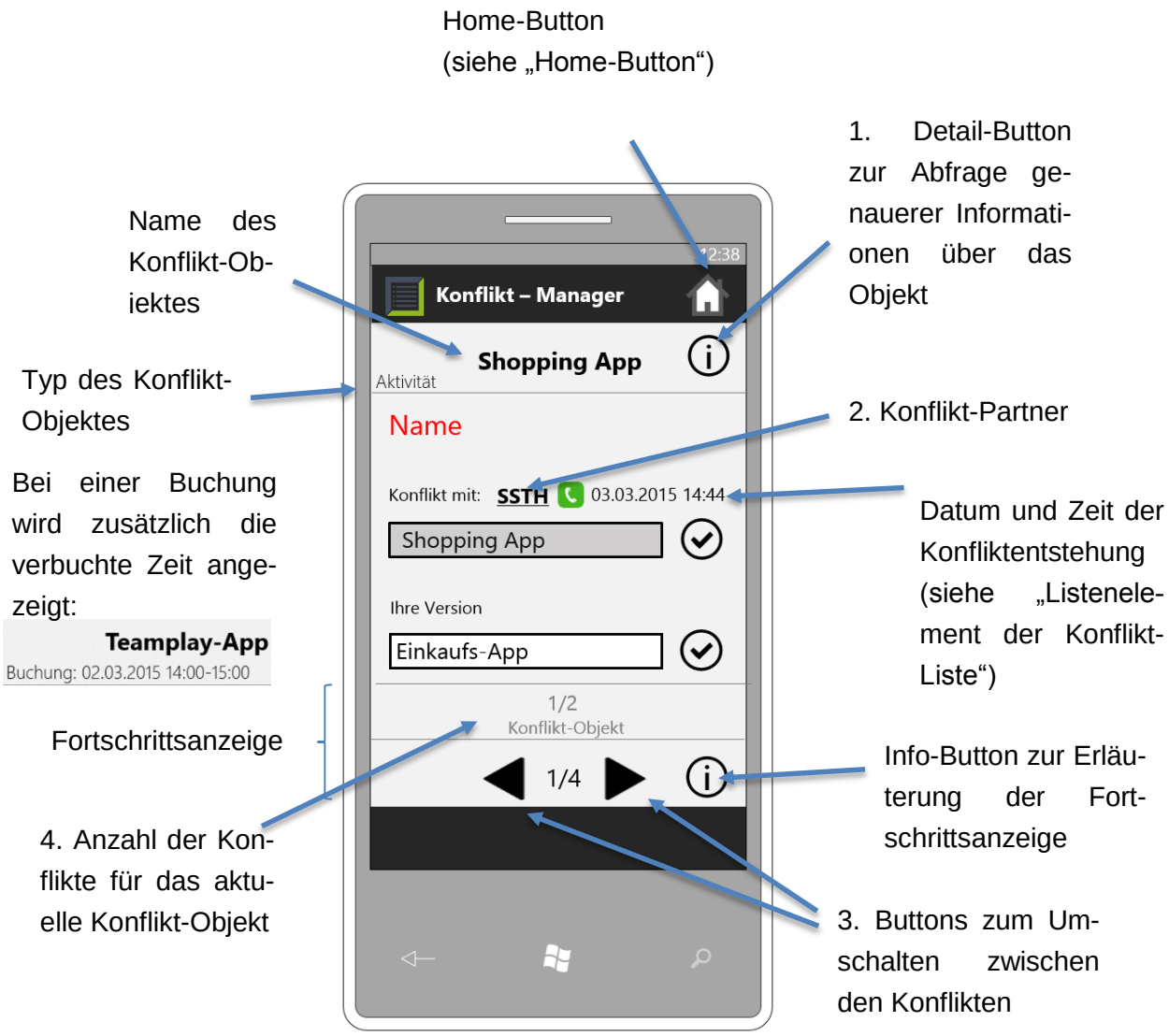
- Aktivitäts-/Buchungsname
- Konflikt-Partner (Mitarbeiterkürzel).

Der vom Benutzer eingegebene Suchbegriff wird mit den eben aufgelisteten Eigenschaften der Konflikte auf (teilweise) Übereinstimmung überprüft und zutreffende Konflikte werden somit aufgelistet.

7.4.4.3.2 Konflikt-Detailansichten

Wird ein Konflikt vom Benutzer ausgewählt, können genauere Details angesehen werden und der Konflikt aufgelöst werden. Die jeweilige Detailansicht unterscheidet sich je nach Art des aufgetretenen Konflikts.

Folgende Daten werden bei jeder Art von Konfliktlösungs-Ansicht angezeigt:



1. Detail-Button

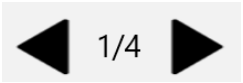
Das Betätigen des Detail-Buttons ermöglicht das Einsehen aller Details des betreffenden Objektes. Das dient dazu, dass der Benutzer sicherstellen kann, dass es sich um das von ihm vermutete Objekt handelt, falls es für ihn unklar ist, welches Objekt nun tatsächlich gemeint ist.

2. Konflikt-Partner

Mit Anzeigen des Konflikt-Partners wird das Kürzel des Mitarbeiters dargestellt, mit dem der Konflikt entstand. Weiters können durch einen Klick auf das Kürzel weitere Informationen zum Mitarbeiter abgefragt werden.

Vorteilhaft am Anzeigen des verantwortlichen Konflikt-Partners ist nun, dass somit jederzeit Rückfragen an diese Person getätigt werden können, um zu entscheiden, welche Version der Änderungen nun tatsächlich übernommen werden soll. Dazu ist auf den grünen Anruf-Button neben dem Kürzel zu tippen, um einen Direktanruf zu starten.

3. Buttons zum Umschalten zwischen den Konflikten



Zur Navigation zu anderen Konflikten können die Pfeil-Buttons im unteren Bildschirmbereich verwendet werden. Damit wird vermieden, dass der Benutzer zuerst umständlich zur Konflikt-Übersicht zurückkehren muss, um den nächsten Konflikt unter die Lupe zu nehmen.

Zwischen den beiden Buttons wird angezeigt, wie viele Konflikte es gesamt gibt und welcher Konflikt gerade angezeigt wird. „1/4“ beispielsweise bedeutet, dass gerade der erste von insgesamt vier Konflikten geöffnet ist. Wird der Pfeil-Button nach rechts betätigt, wird der nächste Konflikt angezeigt und der Text muss somit „2/4“ lauten.

Wird ein Konflikt gelöst, so verschwindet er aus der Liste und kann damit auch nicht mehr mit den Pfeil-Buttons aufgerufen werden.

Zu beachten ist, dass beim Umschalten zum nächsten Konflikt zuerst jener angezeigt wird, der ebenfalls das gleiche Konflikt-Objekt (gleiche Aktivität/Buchung) betrifft, da eine Konfliktauflösung auf Objekt-Ebene praktischer erscheint.

4. Anzahl der Konflikte für das aktuelle Konflikt-Objekt

Wird über die Buttons (siehe 3.) zum nächsten oder vorigen Konflikt gewechselt, so verändert sich auch diese Anzeige. „1/2“ bedeutet also, dass man aktuell den 1. Konflikt von insgesamt 2, die dasselbe Objekt betreffen, ansieht. Schaltet man zum nächsten Konflikt um, wird der nächste dieses Objekt betreffende Konflikt angezeigt und dargestellt und die Anzeige wechselt zu „2/2“.

Die nachstehenden Ansichten sind zu den einzelnen Konfliktarten geplant:

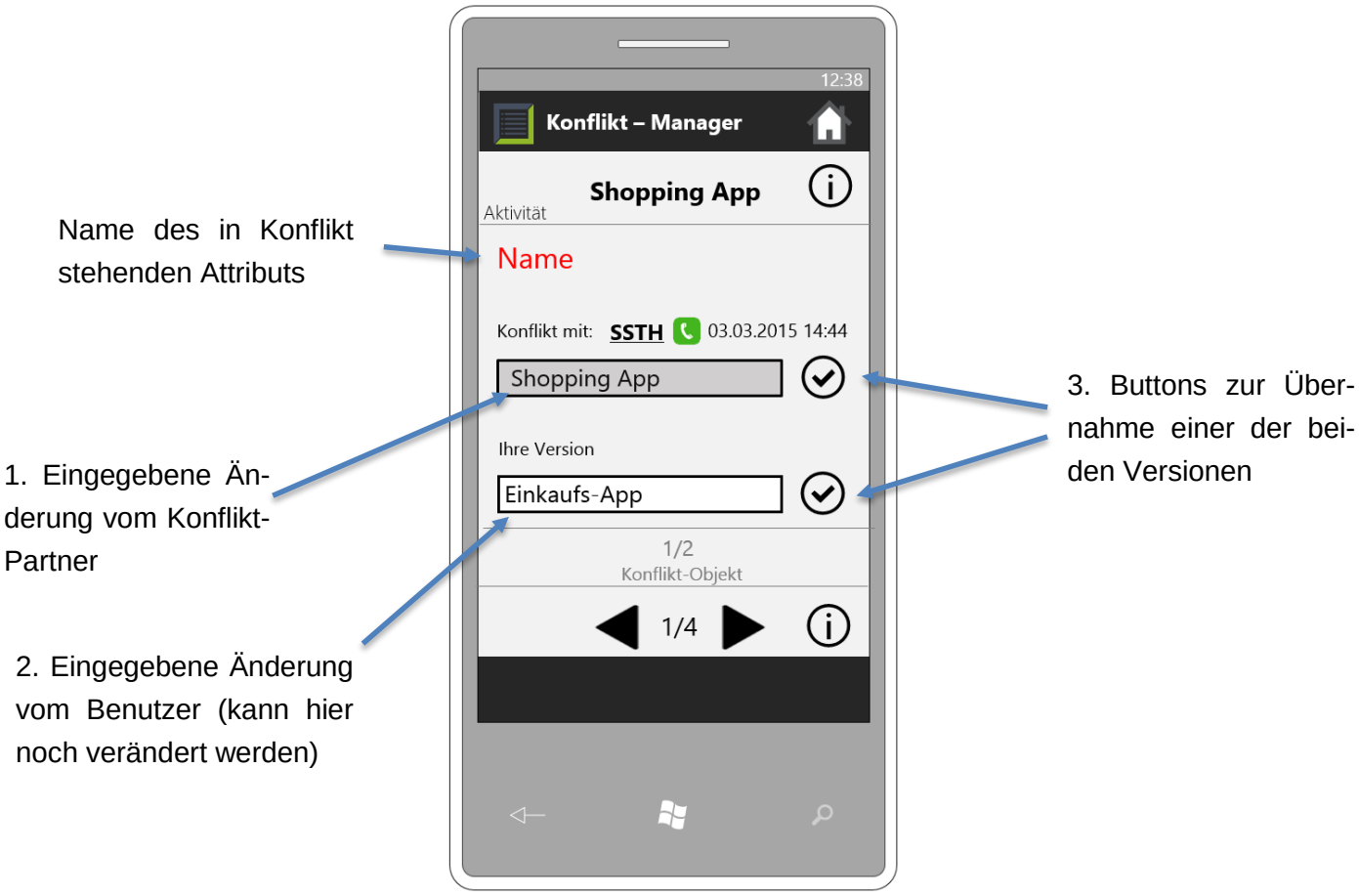
7.4.4.3.2.1 UPDATE-UPDATE

Gewünschte Benutzer-Aktion: UPDATE

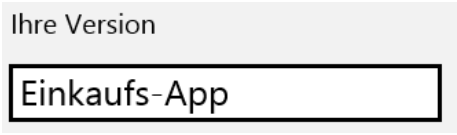
Aktion des Konflikt-Partners: UPDATE

Situationsbeschreibung: Der Benutzer führte offline Änderungen an einem oder mehreren Attributen eines Objekts durch. Ein anderer Mitarbeiter (Konflikt-Partner) hat gleiche Attribute an diesem Objekt in der Zwischenzeit verändert.

Für diesen Konfliktfall steht folgende Art von Bildschirmmaske zur Verfügung:



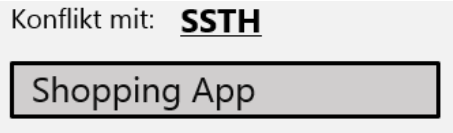
1. Eingegebene Änderung vom Benutzer



In diesem Feld wird jene Änderung am betroffenen Datenobjekt angezeigt, welche der Benutzer ursprünglich durchführen wollte. Hierbei besteht zusätzlich die Möglichkeit, die Änderung erneut anzupassen. Falls wie im Beispiel das Attribut „Name“ einer Aktivität in Konflikt steht, wird ein Textfeld mit dem Inhalt „Einkaufs-App“ (=ursprüngliche gewünschte Änderung des Benutzers) angezeigt. Der Benutzer kann nun den Inhalt des Textfelds nach seinem Ermessen anpassen.

Ist das in Konflikt stehende Attribut kein einfacher Text sondern etwa ein Verweis auf einen Eintrag im Kundenstamm (=Referenz auf einen anderen Datensatz), wird anstatt des Textfelds ein Dropdown-Menü angezeigt.

2. Eingegebene Änderung vom Konflikt-Partner



Neben Version des Benutzers wird auch die aktuellste am Backend erhältliche Version angezeigt. Diese Version hat auch einen bestimmten Verfasser (der Konflikt-Partner).

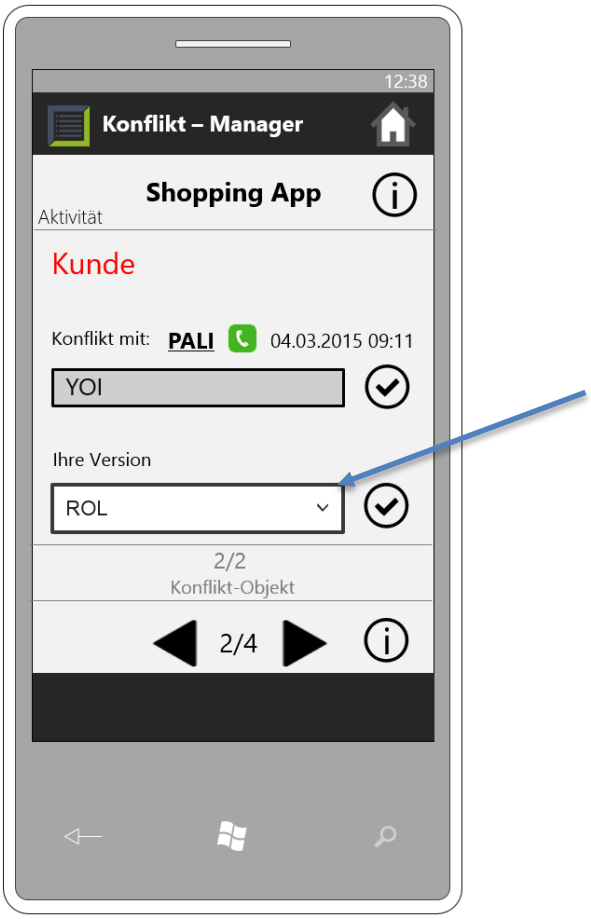
Der Benutzer kann nun entweder seine eigene erzeugte Version des Attributs des Datenobjekts oder jene vom Konflikt-Partner übernehmen.

3. Buttons zur Übernahme einer der beiden Versionen

Neben den beiden in Konflikt stehenden Versionen des betroffenen Attributs ist jeweils ein Button angebracht, mit dem eine der beiden Versionen schlussendlich vom Benutzer übernommen werden kann.

Hat sich der Benutzer für eine passende Version entschieden (evtl. nach Rücksprache mit dem Konflikt-Partner), so wird die gewünschte Version an das Backend gesendet. Falls die Anfrage fehlschlägt bleibt der Konflikt nach wie vor in der Konfliktliste. Erst, wenn erfolgreich Kontakt zum Backend aufgenommen werden kann, erlischt der Konflikt und verschwindet aus der Liste.

Darstellung bei komplexen Datentypen



Eingegebene Änderung vom Benutzer: Hierbei ist das Feld „Kunde“ kein einfacher Datentyp wie ein Text, sondern ist ein komplexer; in diesem Fall ein Verweis auf einen Kunden im Kundenstamm. Daher steht kein Textfeld sondern ein Dropdown-Menü zur Auswahl der Eigenschaft „Kunde“ zur Verfügung.

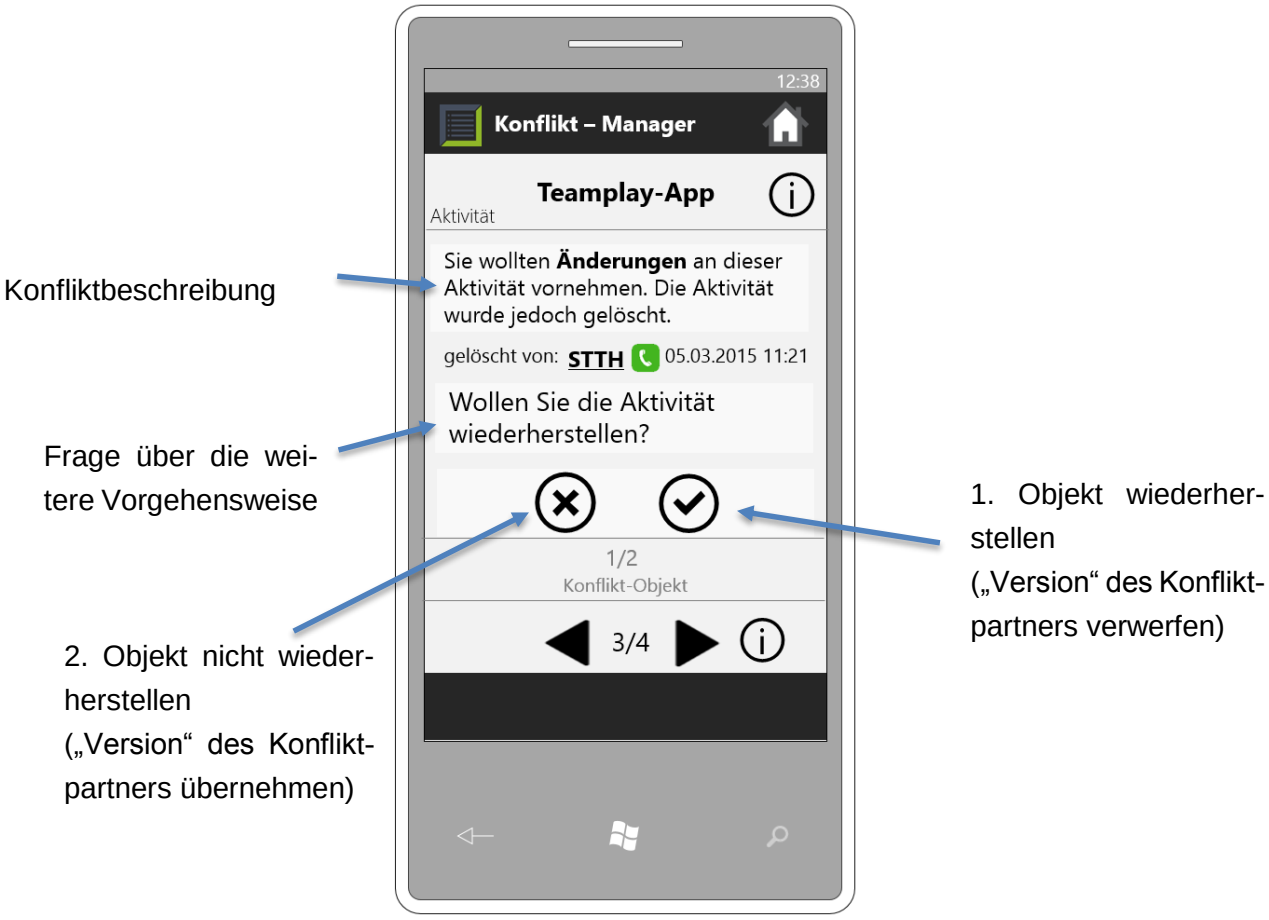
7.4.4.3.2.2UPDATE-DELETE

Gewünschte Benutzer-Aktion: UPDATE

Aktion des Konflikt-Partners: DELETE

Situationsbeschreibung: Der Benutzer führte offline Änderungen an einem oder mehreren Attributen eines Objekts durch. Ein anderer Mitarbeiter (Konflikt-Partner) hat das Objekt in der Zwischenzeit gelöscht.

Für diesen Konfliktfall wird folgende Bildschirmmaske verwendet:



1. Objekt wiederherstellen

Die Aktion „Objekt wiederherstellen“ sorgt dafür, dass die Löschung des in Konflikt stehenden Objekts rückgängig gemacht wird. Mit dem Wiederherstellen wird anschließend auch die vom Benutzer ursprünglich gewünschte Änderung an einem oder mehreren Attributen des betroffenen Datensatzes durchgeführt.

Dabei stammen die Daten des Objekts entweder vom Backend, falls es gelöschte Datensätze weiterhin speichert, oder vom Client, bei dem die Daten lokal gesichert vorliegen. Beide Varianten haben Vor- und Nachteile:

Die Backend-Variante ist einerseits zuverlässiger, andererseits können alle gespeicherten Daten zu einer hohen Wahrscheinlichkeit wiederhergestellt werden. Der Nachteil ist, dass am Backend eine große Menge an Daten anfällt, je mehr Datensätze gelöscht werden. Das würde zusätzliche Lösch-Management-Lösung erfordern, um die Datenmenge möglichst gering zu halten (z. B. zeitliche Steuerung: nach 2 Monaten wird ein gelöschtes Objekt tatsächlich gelöscht).

Vorteilhaft an der Client-Variante ist eine geringe Speicherauslastung am Backend. Jedoch ist der lokale Speicher am Client keine zuverlässige Angelegenheit. Die Chance, dass fehlerhafte oder nur Teile der ursprünglichen Daten bereitgestellt werden, ist daher sehr hoch. Auch der Speicher auf den Client-Geräten ist oftmals sehr beschränkt.

Die Speicherung von gelöschten Datensätzen ist am Backend die zu bevorzugende Variante, auch wenn dadurch die Performance leicht beeinträchtigt werden könnte.

Im aktuellen TeamPlay-System der Fa. Auris IT Consult werden am Backend Datensätze prinzipiell nie gelöscht, sondern als gelöscht markiert.

2. Objekt nicht wiederherstellen

Bei dieser Option wird die vom Konflikt-Partner getätigte Löschung nicht rückgängig gemacht. Der Konflikt ist sofort gelöst und erfordert keine weiteren Operationen.

Falls am Backend jedoch auch gelöschte Datensätze noch gespeichert werden (etwa durch Einführung eines Flags), so könnte nach dem Aufruf dieser Option noch ein Befehl zur endgültigen Löschung an das Backend gesendet werden.

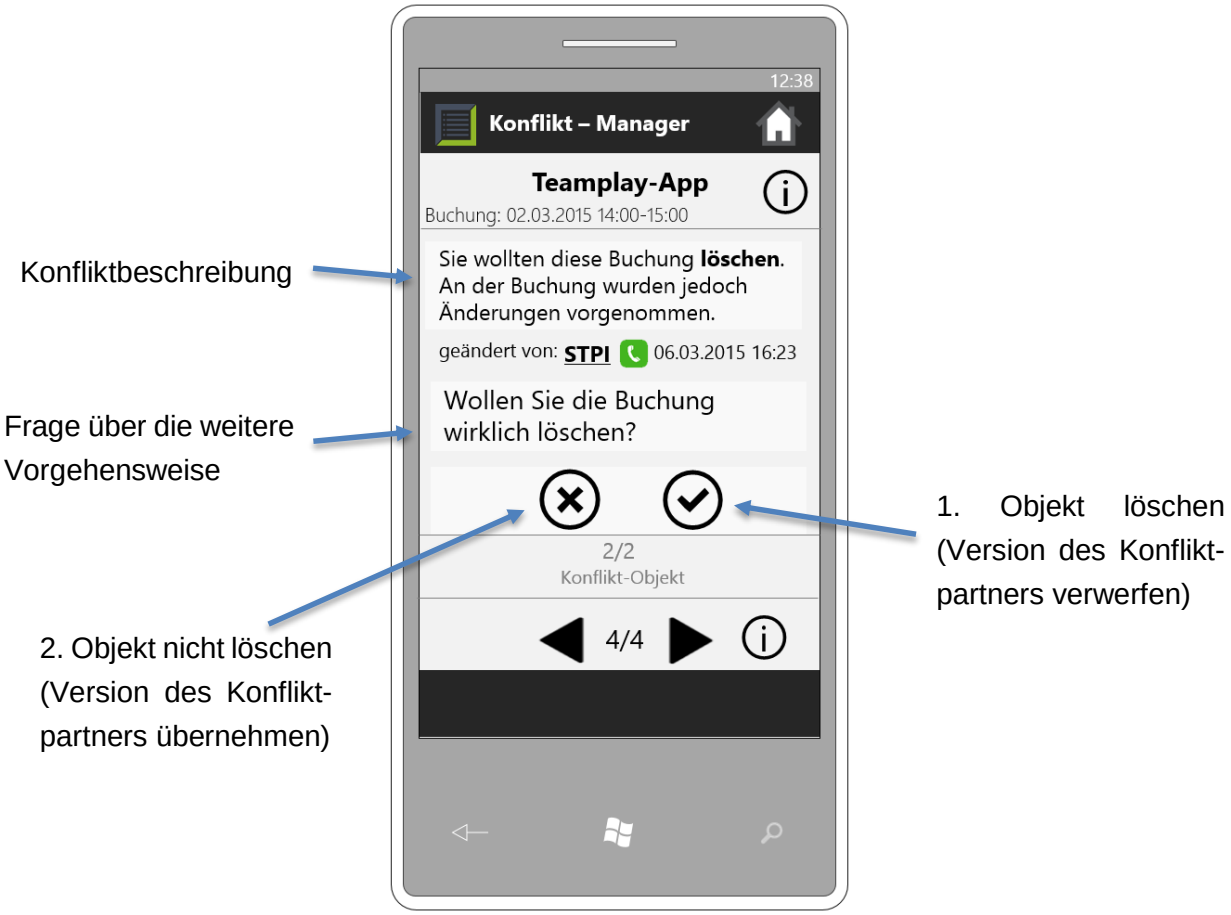
7.4.4.3.2.3 DELETE-UPDATE

Gewünschte Benutzer-Aktion: DELETE

Aktion des Konflikt-Partners: UPDATE

Situationsbeschreibung: Der Benutzer führte offline eine Löschung an einem Objekt durch. Ein anderer Mitarbeiter (Konflikt-Partner) hat in der Zwischenzeit ein oder mehrere Attribute des Objektes verändert.

Dieser Konflikt wird mit folgender Ansicht dargestellt:



1. Objekt löschen

Mit dem Bestätigen der Löschung wird das betroffene Objekt tatsächlich gelöscht und somit sind die Änderungen des Konflikt-Partners unwirksam.

2. Objekt nicht löschen

Die Aktion „Objekt nicht löschen“ sorgt für das Zurückziehen des vom Benutzer stammenden Lösch-Befehls. Dadurch bleibt die Version des Konflikt-Partners unangetastet und steht weiterhin für die Clients am Backend zur Verfügung.

7.4.5 Aufwandsschätzung

7.4.5.1 Vorausgesetzte Technologien

Als Grundlage für die Aufwandsschätzungen wird für die Clientgeräte und das Backend folgender Technologieeinsatz angenommen:

- Client
  - Verwendung einer plattformübergreifenden Entwicklungstechnologie
  - Mögliche Programmiersprache(n):
    - Java, Action Script, C++
- Backend
  - Entwicklung mit der Technologie ASP.NET Web Api
  - Mögliche Programmiersprache(n):
    - C#

7.4.5.2 Übersicht

Modul	Schätzung (h)	Implementierung (Server, Client)
Offlinefähigkeit am Client		
Konzept der Offline-Speicherung (Zeitpunkt, Format, effizientes Speichern, Logik für Workitem-Queue)	25	Client
Umsetzung der Offline-Speicherung	20	Client
Konflikterkennung		
Konzept Konflikterkennung-Logik	25	
Umsetzung Konflikterkennung-Logik	15	Server
Kommunikation Client - Service	8	Client
Konfliktlösung		
Konzept Konfliktlösungs-Logik (GUI + Logik)	25	
GUI Konfliktlösung (Konfliktlösungs-Guide/Manager)	35	Client
Umsetzung Konfliktlösungs-Logik (GUI-Logik + Kommunikation mit Backend mit entsprechender Fehlerbehandlung)	20	Client
Testen & Bug Fix	25	Server/Client
	198	

Tabelle 23: Aufwandsschätzung

7.5 Release-Kompatibilität

Bei Software, die nach der Client-Server-Architektur aufgebaut ist, wird früher oder später für die Client-Seite ein neues Release freigegeben. Meist werden bestimmte Modellklassen erweitert oder manchmal reduziert. Erweiterungen, Anpassungen oder Deaktivieren von Funktionalität sind ebenfalls zu beachten. All diese Änderungen führen theoretisch zu keinen Problemen, wenn alle Clients stets auf das neueste Release updaten. In der Praxis jedoch werden mit einer hohen Wahrscheinlichkeit nicht alle Clients über die aktuellste Version einer Software verfügen. Deswegen ist es wichtig, die Implementierung der Softwarelösung so vorzunehmen, dass Server und Client unabhängig von der Release-Version des Clients immer problemlos miteinander kommunizieren können. Der Server muss also somit in der Lage sein, mit allen bisherigen Releases umzugehen oder wenigstens dem Client Fehlermeldungen bereitstellen können, damit die Benutzerfreundlichkeit nicht beeinträchtigt wird.

In diesem Kapitel werden Überlegungen dargestellt, wie das beschriebene Problem gelöst werden könnte.

7.5.1 Release-Version am Client

Es gibt verschiedene Möglichkeiten, die Release-Version so einzubinden, dass diese vom Client zum Server gelangt.

7.5.1.1 Möglichkeit 1: Versionsnummer im Datenmodell

Eine Herangehensweise ist, die Version des Release als einfache Zahl direkt bei den Klassen des Datenmodells einzufügen. Dabei besteht die Möglichkeit, dass für verschiedene Klassen auch verschiedene Release-Nummern verwendet werden können. Das ist dann der Fall, wenn sich manche Klassen nur bei einem bestimmten Release geändert haben. Ändern sich Klassen von einem Release auf den anderen nicht, so wird auch die Versionsnummer nicht verändert.

Die Implementierung würde etwa so aussehen (in Java):

```
class Activity {
    //Property activityId (added in Release 1)
    private int activityId;
    public int getActivityId() {
        return activityId;
    }
    public void setActivityId(int activityId) {
        this.activityId = activityId;
    }

    //Property customer (added in Release 4)
    private Customer customer;
    public Customer getCustomer() {
        return customer;
    }
    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    //more properties

    //Release version
    public int getReleaseVersion() {
        return 4;
    }
}

class Customer {
    //Property customerId (added in Release 1)
    private int customerId;
    public int getCustomerId() {
        return customerId;
    }
    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    //Property name (added in Release 1)
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    //more properties

    //Release version
    public int getReleaseVersion() {
        return 1;
    }
}
```

Abbildung 68: Versionsnummer im Datenmodell (Beispielcode für Modellklassen)

Wie im Code-Beispiel zu sehen ist, wird für eine Modellklasse die Release-Version auf jenen Wert gesetzt, der dem Release entspricht, bei dem eine oder mehrere Änderungen ab der Klasse vorgenommen wurden. Beispielsweise besitzt die Klasse Activity seit Release 1 ein Property activityId, aber erst seit Release 4 ein Property customer. Daher ist insgesamt die Release-Version auf 4 zu setzen. Bei der Klasse Customer hat sich hingegen seit Release-Version 1 keine Änderung ereignet, weshalb die Release-Version hier 1 lautet.

7.5.1.1.1 Vorteile

- Es ist sofort ersichtlich, bei welcher Klasse und bei welchem Release Änderungen vorgenommen wurden
- Nummer direkt in serialisierbaren Daten enthalten; keine zusätzlichen Datenstrukturen notwendig oder zu beachten

7.5.1.1.2 Nachteile

- Unnötige Vergrößerung der zu übertragenden Daten an das Backend (vor allem, wenn Listen gesendet werden, bei der alle Einträge mit derselben Release-Nummer-Property ausgestattet sind)
- Zusätzliches Attribut in allen Modellklassen
- Fehleranfällig bei Release-Wechsel; Versionsnummern müssen in jeder geänderten Klasse korrigiert werden

7.5.1.1.3 Release-Wechsel

Bei einem Release-Wechsel sind folgende Schritte empfehlenswert:

1. Kopie des Client-Projektes anlegen für das neue Release
2. Modellklassen des neuen Release (optional bzw. nicht bei allen Klassen erforderlich)
    a. Änderungen von Attributen vornehmen
    b. Kommentieren von Änderungen (z. B. hinzugefügt bei Release X, Attribut entfernt bei Release Y, …)
    c. Attribut der Versionsnummer des Releases aktualisieren

7.5.1.2 Möglichkeit 2: Globale Versionsnummer

Als Gegenstück zur vorherigen Möglichkeit, die Versionsnummer des Release am Client praktisch zu handhaben, wird hier die Release-Nummer nicht direkt im Datenmodell abgelegt.

Stattdessen wird die Nummer des aktuellen Release als globales Feld (z. B. in einer Klasse als Konstante) abgelegt.

In einer Implementierung (Java) wird also die Release-Version etwa so gespeichert:

```
class Release {

    public static final int VERSION = 4;

}
```

Abbildung 69: Globale Versionsnummer (Beispielcode für die Release-Klasse)

Das Datenmodell ist hierbei von der Release-Nummer befreit:

```
class Activity {
    //Property activityId (added in Release 1)
    private int activityId;
    public int getActivityId() {
        return activityId;
    }
    public void setActivityId(int activityId) {
        this.activityId = activityId;
    }

    //Property customer (added in Release 4)
    private Customer customer;
    public Customer getCustomer() {
        return customer;
    }
    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    //more properties

}
```

Abbildung 70: Globale Versionsnummer (Beispielcode für eine Modellklasse)

Damit die Release-Nummer nun zum Server übertragen werden kann, muss diese irgendwie in das Datenpaket eingebaut werden. Am besten wird dazu ein neues Feld im HTTP-Header des Datenpakets an den Server definiert:

Header				
Host	Accept	Method	Agent	**Release-Version**
<URL>	application/json	POST	<AGENT>	**3**
Body				
<serlialized data>				

Abbildung 71: HTTP-Header mit Release-Versionsnummer

7.5.1.2.1 Vorteile

- Befreiung der Datenmodellklassen von der Release-Nummer
- Bei Release-Wechsel ist nur die globale Release-Nummer anzupassen
- Keine mehrfache Übertragung der Release-Nummer im Datenpaket (verglichen zu Möglichkeit 1)

7.5.1.2.2 Nachteile

- Neues Feld im HTTP Header muss definiert werden

7.5.1.2.3 Release-Wechsel

Bei einem Release-Wechsel sind folgende Schritte empfehlenswert:

1. Für das neue Release eine Kopie des Client-Projektes anlegen
2. Modellklassen des neuen Release (optional bzw. nicht bei allen Klassen erforderlich)
    a. Änderungen von Attributen vornehmen
    b. Kommentieren von Änderungen (z. B. hinzugefügt bei Release X, Attribut entfernt bei Release Y, …)
3. Globale Release-Version anpassen aufs neue Release

7.5.2 Release-Management am Backend

Nicht nur die Client-Seite muss bei neuen Releases angepasst werden, sondern auch vor allem das Programm am Webservice ist entsprechend zu ändern.

7.5.2.1 Modellklassen

Sowohl die Client-Lösung als auch der Webservice verwendet Modellllassen, um eine sinnvolle Kommunikation und den Datenaustausch zwischen Client und Server zu ermöglichen.

7.5.2.1.1 Möglichkeit 1: Nur aktuellste Version speichern

Die erste Möglichkeit besteht darin, im Backend-Projekt von allen Modellklassen die des aktuellsten Release zu speichern. Es gibt keine Information von vorigen Releases in Form von Klassen. Gleichzeitig bedeutet das auch, dass am Backend möglicherweise ohne IntelliSense auf vom Client übermittelte Objekte von älteren Releases zugegriffen werden müsste (falls Attribute von alten Releases in neuen nicht mehr vorhanden sind oder unbenannt wurden).

7.5.2.1.1.1 Vorteile

- Übersichtlichkeit, da jeweils die aktuellsten Versionen der Modellklassen eingebunden werden
- Geringer Programmieraufwand

7.5.2.1.1.2 Nachteile

- vom Client empfangene serialisierte Objekte alter Releases können nicht 100%ig in die aktuellste Version der Klasse geparst werden, folglich:
- keine Autovervollständigung für von Client übermittelte Objekte älterer Releases
- Fehleranfälligkeit erhöht, falls von Objekten alter Releases Daten ausgelesen werden müssen

7.5.2.1.1.3 Release-Wechsel

Bei einem Release-Wechsel fallen folgende Schritte an:

1. Sicherung der Modellklassen des vorigen Release, da hier die Klassen der vorigen Releases nicht im Projekt eingebunden sind
2. gewünschte Änderungen am Datenmodell werden vorgenommen
3. Datenbanktabellen werden ebenfalls adaptiert

7.5.2.1.2 Möglichkeit 2: Release-Verlauf speichern

Am Backend werden hier prinzipiell von jeder Klasse alle Release-Versionen gespeichert.

7.5.2.1.2.1 Namenskonvention

Dabei wird folgende Namenskonvention angewendet:

Release	Klassenbezeichnung
1	Klasse_v1
2	Klasse_v2
3	Klasse_v3
4 (aktuell)	Klasse

Tabelle 24: Namenskonvention Release-Verlauf

Ist beispielsweise Release 4 aktuell, so existieren für die Abbildung einer Aktivität die Klassen Activity_v1, Activity_v2, Activity_v3 und Activity (aktuellstes Release).

Wenn sich eine Klasse über ein oder mehrere Releases nicht ändert, wird dafür auch keine eigene Klasse angelegt.

Beispielsweise hat sich die Klasse Customer erst bei Release 4 verändert, jedoch ist sie bei Release 1 bis 3 unverändert geblieben:

Release	Klassenbezeichnung
1	Customer_v1
2	Customer_v1
3	Customer_v1
4 (aktuell)	Customer

Tabelle 25: Namenskonvention Release-Verlauf Beispiel

7.5.2.1.2.2 Vorteile

- Vollständigkeit und Korrektheit gegeben, da von allen Modellklassen alle Release-Versionen auf einen Blick einsehbar sind
- vom Client gelieferten Objekte beliebiger Releases können problemlos in die dazugehörige Modellklasse geparst werden, dadurch:
- geringere Fehleranfälligkeit bei der Handhabung von Modell-Objekten

7.5.2.1.2.3 Nachteile

- je mehr Releases desto unübersichtlicher das Datenmodell
- erhöhter Aufwand bei Release-Wechsel
- erhöhter Programmieraufwand beim Parsen von Objekten
    - Erläuterung:

Instanzen von Customer_v1 und Customer_v2 beispielsweise sind zwar logisch gesehen gleichartige Objekte, jedoch sind sie durch die hier gewählte Vorgehensweise durch individuelle, voneinander unabhängige Klassen abgebildet. Somit muss bei empfangenen Objekten vom Client zuerst überprüft werden, um welche Release-Version es sich handelt und danach kann die entsprechende Klasse zum Parsen gewählt werden. Dadurch kann auch kein allgemeingültiger Algorithmus entworfen werden, welcher abhängig von der Version automatisch eine Instanz der Klasse mit der richtigen Release-Version erzeugt.

Anmerkung: Nachfolgender Code ist als vereinfachter Pseudocode zu interpretieren. Der Aufruf request.getData() soll dabei ein serialisiertes Objekt in Textform (z. B. ein JSON-Objekt) liefern.

```
int version = request.getHeader().getField(„releaseVersion“);

if (version == 1) {

   Customer_v1 c = request.getData().parse<Customer_v1>();

   //do something

}

if (version == 2) {

   Customer_v2 c = request.getData().parse<Customer_v2>();

   //do something

}
```

weil beide Klassen voneinander unabhängig sind, ist folgendes nicht zielführend:

```
var c = CustomReleaseParser.parse(version, request.getData());
```

die Variable “c” könnte lediglich mit dem Typ „Object“ versehen werden, da Customer_v1 und Customer_v2 keine gemeinsamen Elternklassen haben (außer eben Object).

7.5.2.1.2.4 Release-Wechsel

Bei einem Release-Wechsel fallen folgende Schritte an:

1. für zu ändernde Klassen wird eine neue Klasse erstellt (Namenskonvention: „Klasse“) und die Änderungen werden hier durchgeführt
2. die jeweiligen dazugehörigen „alten“ Klassen werden unbenannt, also mit einem Kürzel zur alten Version versehen

- Beispiel:

  Aktuelles Release: 4

  Neues Release fällt an, Klasse Customer wird im neuen Release verändert.

  → Neue Klasse „Customer“ definieren und da die gewünschten Änderungen
  vornehmen

  → vorherige Klasse „Customer“ wird auf „Customer_v4“ unbenannt

3. Datenbanktabellen werden ebenfalls adaptiert

7.5.2.1.3 Möglichkeit 3: Modellklassen mit voller Release-Information

7.5.2.1.3.1 Konzept

Nach diesem Konzept werden alle Release-Versionen einer Modellklasse in genau einer Klasse abgebildet. Das ist vor allem dann sinnvoll, wenn bei einem Datenmodell zukünftig nur Erweiterungen um Attribute und keine/wenige Abänderungen bzw. Reduzierungen von Attributen.

Beispiel:

```
class Activity {
    //Property activityId (added in Release 1)
    private int activityId;
    public int getActivityId() {
        return activityId;
    }
    public void setActivityId(int activityId) {
        this.activityId = activityId;
    }

    //Property description (added in Release 2)
    private String description;
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }

    //Property customer (added in Release 4)
    private Customer customer;
    public Customer getCustomer() {
        return customer;
    }
    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    //more properties

}
```

Abbildung 72: Modellklasse mit voller Release-Information (Beispielcode)

Wie im Beispiel zu sehen ist, kann demnach die Zusammensetzung der Klasse von einem bestimmten Release nur über Programmierer-Kommentare oder separate Dokumentationen festgestellt werden.

Das Parsen von empfangenen Objekten vom Client erfolgt mit diesem System problemlos mit einem Kommando, denn es wird für jeden Bestandteil des Modells nur jeweils eine Klasse verwendet. Es muss vor dem Parsen keine Überprüfung auf die Release-Version durchgeführt werden, denn die jeweilige Klasse enthält alle Informationen des gesamten Release-Verlaufs. Damit enthält das geparste Objekt möglicherweise auch einige Attribute mit NULL-Werten, da Clients früherer Releases diese noch nicht bereitstellen können.

7.5.2.1.3.2 Optional: Modellklassen des aktuellen Release separat halten

Nachdem je nach Änderungsausmaß die Modellklassen sehr unübersichtlich werden können, empfiehlt es sich, daneben Modellklassen mit den Attributen des aktuellen Release zu speichern:

Activity	
ActivityHolder Klasse mit Attributen aller Releases	Activity Klasse mit nur den Attributen des aktuellen Releases

Tabelle 26: Bestandteile eines Modellbausteins mit Optionaler Modellklasse des aktuellsten Releases

Das könnte auch dann sinnvoll sein, wenn eine spezielle objektrelationale Datenzugriffstechnologie (z. B. EntityFramework) verwendet wird. Damit ist es sauberer, mit separaten Klassen mit der aktuellsten Release-Version mit einer Datenbank zu kommunizieren.

Zur Überführung von einem Objekt mit Information über alle Releases zu einem Objekt des aktuellsten Releases muss ein eigener Mapper eingeführt werden, der die Modellklassen dementsprechend umwandelt.

7.5.2.1.3.3 Vorteile

- Überschaubarkeit, da für alle Releases eines Modellbestandteils eine Klasse verwendet werden kann
- Geringer Programmieraufwand, Parsen der Objekte erfolgt ohne Überprüfung der Release-Version des gelieferten Objekts
- Release-Wechsel rasch möglich
- optimales Konzept, wenn bei Modellklassen nur Erweiterungen zu erwarten sind

7.5.2.1.3.4 Nachteile

- Gefahr der „Entartung“; Modellklassen können leicht unübersichtlich werden (Abhilfe durch separate Klasse)
- Fehlende Information darüber, welche Attribute zu welcher Release-Version gehören, ist möglich (eigene Dokumentation notwendig; ODER: mit Verwendung der separaten Klassen können diese beim Release-Wechsel archiviert werden)

7.5.2.1.3.5 Release-Wechsel

Bei einem Release-Wechsel fallen folgende Schritte an:

1. Dokumentation der Änderungen (bzw. Archivierung der Modellklassen der aktuellsten Releases)
2. gewünschte Änderungen am Datenmodell werden vorgenommen
3. optional: falls separate Klassen mit den aktuellsten Release-Versionen gehalten werden, werden auch diese angepasst
4. Datenbanktabellen werden ebenfalls adaptiert

7.5.2.2 Datenbank

Neben programmiersprachenspezifischen Modellklassen muss auch darauf geachtet werden, die Datenbank an das neueste Release anzupassen. Die Datenbank soll gleichzeitig auch so gestaltet werden, dass auch mit älteren Releases noch umgegangen werden kann, ohne in der Programmlogik am Backend viele Überprüfungen zwischenschalten zu müssen.

7.5.2.3 Funktionalität

Das Backend stellt dem Client eine Reihe von aufrufbaren Funktionen zur Verfügung (=verschiedene HTTP-Endpunkte). Beim Release-Management sind hierbei einige Dinge zu beachten.

7.5.2.3.1 Hinzufügen von Funktionalität

Soll in der Softwarelösung eine neuartige Funktion eingebunden werden, die nicht in bestehende Funktionen integriert werden kann, wird ein neuer Endpunkt (=ein neuer aufrufbarer HTTP-Endpunkt) am Service angelegt. In ASP.NET würden beispielsweise neue Funktionen in einem Controller definiert werden.

Nur Clients, die ebenfalls über die neueste Version der Software verfügen, können dann die neue Funktionalität ansteuern.

7.5.2.3.2 Änderung von Funktionalität

Wird der Inhalt von Funktionen am Backend angepasst, darf keinesfalls der HTTP-Endpunkt verändert werden. Die URL, mit der die Funktion aufgerufen werden kann, muss dieselbe bleiben, da nicht davon ausgegangen werden darf, dass die Benutzer ihre Client-Software regelmäßig aktualisieren.

Der Inhalt von Funktionen kann verändert werden, sofern die Kompatibilität mit allen Release-Versionen erhalten bleibt.

7.5.2.3.3 Entfernen von Funktionalität

Soll die Unterstützung aufrufbarer HTTP-Endpunkte eingestellt werden, muss beachtet werden, dass Clients mit alten Release-Versionen diesen Endpunkt nicht mehr ordnungsmäßig ansteuern können. Für den Fehlerfall hat der Client eine Implementierung vorzunehmen, die den Benutzer darüber in Kenntnis bringt, dass eine bestimmte Funktionalität am Backend eingestellt wurde.

Zwei grundsätzliche Möglichkeiten sind zu unterscheiden:

1. Der Endpunkt zu der Funktion am Backend bleibt erhalten, jedoch wird der gesamte Inhalt entfernt und durch Rückgabe einer Fehlermeldung in das Response-Paket ersetzt. Die Fehlermeldung kann hierbei also auch detailliert und spezifisch für diese Funktion beschrieben werden. Der Client auf der anderen Seite muss ebenfalls eine Lösung implementieren, diese Fehlermeldung abzufangen und dem Benutzer darzulegen.
2. Die Funktion am Backend wird tatsächlich entfernt. Der Client muss dementsprechend den Fehlercode „not found“ (404) handhaben. Daraufhin wird eine allgemeine Fehlermeldung ausgeben.

Sollen detaillierte Fehlertexte über das Entfernen einer Funktion am Client ausgegeben werden, ist die 1. Variante zu bevorzugen.

8 Evaluierung

8.1 Planung vs. Realisierung

Da zu Beginn für das Projektteam die plattformübergreifenden Technologien noch sehr fremd waren, stellte der Projekteinstieg ein gewisses Risiko dar. Deswegen wurde auch eine angemessene Einarbeitungszeit geplant. Glücklicherweise konnte sich das Projektteam schon nach kurzer Zeit gut orientieren, nachdem die ersten Erfahrungen mit den Technologien gesammelt wurden.

Adobe AIR erwies sich zu Entwicklungsstart nach Recherchearbeiten als Favorit. Deswegen wurde bei der Entwicklung auch damit begonnen. Darüber hinaus ist die Adobe AIR App optisch anspruchsvoll. Allerdings drängte die vergleichsweise schlechte Performance des Prototyps Adobe AIR wieder weg von der Favoritenposition und erforderte mehr Zeit als geplant zur Einbringung von performancesteigernden Maßnahmen.

Unerwartet war auch das Ausscheiden der Technologie Eclipse Scout, die nicht für den Zweck geeignet war, primär deswegen, weil damit keine mobilen Apps erstellt werden können und ein zusätzliches Backend implementiert werden müsste.

Stellvertretend wurde daher Embarcadero als Alternative eingesetzt. Embarcadero machte zu Beginn einen positiven Eindruck und wurde schließlich aufgrund der unübersichtlichen Entwicklungsweise abgebrochen. Somit konnte das Projektteam nur zwei statt der drei geplanten Prototypen erstellen.

Die Programmierung des PhoneGap-Prototyps erwies sich als aufwändiger als erwartet, wodurch mehr Zeit als geplant investiert wurde. Die optische Darstellung erreichte zwar nicht das Level von Adobe AIR, das Gesamtergebnis stellte jedoch letztendlich PhoneGap in die Position des Favoriten.

Zu Projektende stellte sich die Diplomarbeit trotz einiger unerwarteter Ereignisse als Erfolg heraus. Das Projektteam sowie auch der Auftraggeber Auris IT Consult konnten über die plattformübergreifende Entwicklung weitereichende Kenntnisse erlangen und sich eine neue Perspektive diesbezüglich bilden.

8.2 Erfahrungen im Team

Da unsere Diplomarbeit ein eher größeres Projekt ist und alleine nur schwer umzusetzen wäre, haben wir uns entschieden, sie zu zweit zu gestalten. Dies bringt einige Vorteile mit sich, wie zum Beispiel mehr Motivation durch gegenseitiges Anspornen und schnellere Problemlösung durch gemeinsames Lösen von Aufgaben.

Paul Schmutz hat in der Rolle als Projektleiter die E-Mail Kommunikation mit dem Auftraggeber zum Großteil übernommen und sich meistens um die Organisation der Meetings gekümmert. An der Entwicklung der Prototypen und der dazugehörigen Dokumentation haben beide Diplomanden gearbeitet. Da die Entwicklung mehrerer Prototypen beinhaltet, konnten die Programmierarbeiten gut aufgeteilt werden, sodass beide Diplomanden an verschiedenen Prototypen arbeiten konnten. Trotzdem wurden Probleme, die aufgetreten sind, gemeinsam besprochen und gelöst, da diese beim jeweils anderen eventuell schon einmal aufgetreten waren oder früher oder später aufgetreten wären.

8.3 Nutzen

Die Auris IT Consult GmbH verwendet unsere Erfahrungen bei der Entwicklung der Prototypen mit den jeweiligen Technologiekandidaten als Entscheidungsgrundlage, um herauszufinden, auf welche Technologie sie zukünftig bei der App Entwicklung setzen werden.

Neben den Prototypen selbst werden auch die Dokumentation und Erfahrungsberichte, welche von uns erstellt wurden, ausgewertet. Ziel ist es, eine geeignete Cross-Platform-Development Technologie zu finden, die nicht nur für Kundenprojekte verwendet werden kann, sondern auch für das eigene Produkt TeamPlay.

8.4 Stundenverteilung

Während der Diplomarbeit wurden die Tätigkeiten mit Datum und Dauer laufend in einer Excel-Stundenliste dokumentiert. Durch eine passende Verteilung der Aufgaben, arbeiteten die Projektmitglieder in etwa gleich viele Stunden.

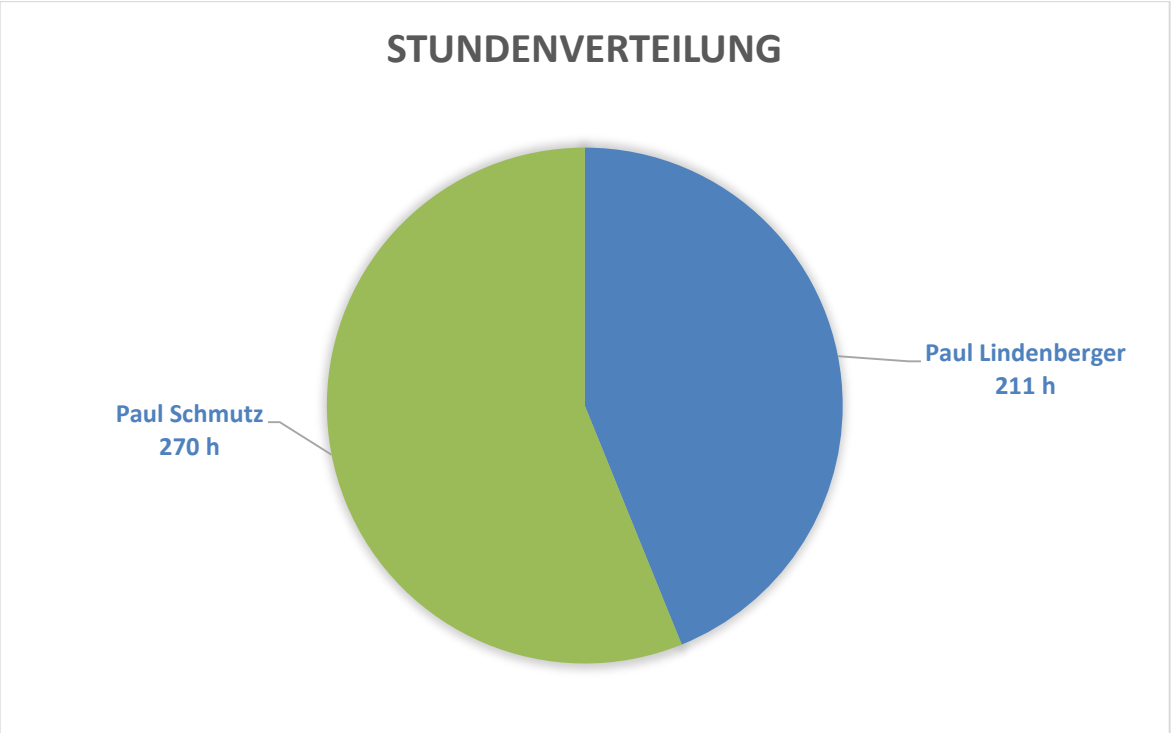


Abbildung 73: Stundenverteilung

A Anhang und Ergänzungen

A1 Glossar

Backend Im Zusammenhang mit dieser Arbeit ist das Backend der TeamPlay-Service, mit dem die Prototypen kommunizieren. Im Allgemeinen ist das Backend eine Schnittstelle in einer Softwareanwendung, die von einer höheren Softwareschicht angesprochen wird. In der Praxis stellt das Backend meist die Serveranwendung dar, die die Aufgabe der Datenverwaltung übernimmt.
Bug Ein Bug ist ein Programmierfehler, der zu Fehlverhalten der Software führen kann.
Client Anstatt Client wird oft der Begriff Frontend verwendet. In einem Client-Server-Modell beansprucht der Client die Dienste vom Server.
CSS CSS (Cascading Style Sheets) bezeichnet eine Designsprache für Elemente eines mit einer Beschreibungssprache erstellten Dokuments. Häufig wird CSS auf XML-Dokumente und vor allem HTML-Dokumente angewendet.
Debugging Debugging ist der Prozess der Auffindung und Behebung von Fehlern (bzw. Bugs) im Code von Softwareprogrammen.
Emulator Ein Emulator ist ein Programm, das ein anderes Softwaresystem hinsichtlich bestimmter Aspekte nachbildet (z. B. Emulator zur Abbildung eines Android-Smartphones auf einem Computer).
Gantt-Plan Ein Gantt-Plan oder –Diagramm ist ein Zeitplanungsinstrument in einem Projekt und bildet die zu erledigenden Aktivitäten als Balken in einem Zeitdiagramm ab.
GitHub GitHub ist ein Internetdienst zur Bereitstellung von Softwareprojekten.

GUI

Die GUI (Graphical User Interface) bezeichnet die grafische Oberfläche einer Software, mit der der Benutzer mithilfe von Eingabegeräten das Programm bedienen kann.

HTML

HTML (Hypertext Markup Language) ist eine Beschreibungssprache für Inhalte in HTML-Dokumenten, welche die Grundlage des Internets darstellen und von einem Browser angezeigt werden können.

Intellisense

Intellisense ist das Autovervollständigungswerkzeug von Microsoft Visual Studio und erleichtert dem Programmierer die Entwicklung des Programms.

Kick-Off-Meeting

Das erste Treffen des Projektteams mit dem Auftraggeber zu Beginn eines Projektes wird Kick-Off-Meeting genannt. Dabei werden die Grundzüge und die Planung des Projekts besprochen.

Mockups

Ein Mockup bezeichnet eine Nachbildung eines Modells, wie z. B. Abbildungen der GUI eines geplanten Softwareprojektes bevor das Projekt tatsächlich umgesetzt wird.

In dieser Arbeit wird der Begriff ausschließlich für die Planungsabbildungen der GUI-Ansichten verwendet.

Nativ

Nativ bedeutet in Bezug auf Softwaretechnologien, dass beispielsweise eine Softwareapplikation auf ein bestimmtes Betriebssystem oder für bestimmte Hardware maßgeschneidert wird. Das Gegenstück dazu ist der Begriff „plattformübergreifend“.

Programmierpattern

Das Programmierpattern oder Design Pattern bezeichnet eine grundlegende Strukturierungsmethode für Programmcode, welche immer wiederkehrende Entwurfsprobleme lösen soll.

Queue

Eine Queue besteht aus einer Reihe von Elementen und ist nach dem Warteschlangenprinzip aufgebaut. Die Elemente werden also hinten an der Reihe angefügt und werden vorne weggenommen (First-In-First-Out-Prinzip).

Release

In der Softwaretechnik ist ein Release die Freigabe und Veröffentlichung einer Software.

Scrum Scrum ist eine agile Methode zur flexiblen Abwicklung von (eher kleineren) Softwareprojekten, bei welchen sich das Projektteam selbst organisiert.
Singleton Singleton ist ein Design Pattern für die Programmierung. Dabei wird sichergestellt, dass von einer Klasse während der gesamten Programmausführungszeit nur ein Objekt instanziert werden kann.
Windows Azure Windwos Azure ist eine kostenpflichtige Cloud-Plattform von Microsoft, die die Veröffentlichung sowie Skalierung von Webdiensten ermöglicht, ohne selbst über leistungsfähige Hardware zu verfügen.

A2 Internetverzeichnis

[1] TeamPlay Logo

http://www.auris-consult.at/gfx/products/teamplay/teamplay_logo_noversion.png

[Zugriff: 21.04.2015]

[2] JobScheduler Logo

http://www.auris-consult.at/gfx/picarchive/JobScheduler/Logobanner_JobScheduler.png

[Zugriff: 21.04.2015]

[3] CashBox Logo

http://www.auris-consult.at/gfx/products/cashbox/logo_small.png

[Zugriff: 21.04.2015]

[4] HTL Perg Logo

http://www.htl-perg.ac.at/templates/htl-perg/images/logo.gif

[Zugriff: 27.05.2015]

[5] DI Michael Stumpfl

http://www.htl-perg.ac.at/images/lehrerfotos/stumpfl_michael.jpg

[Zugriff: 27.05.2015]

[6] Dreyer, Eric (2008): Agiles Projektmanagement mit Scrum. Präsentation zu einer Studienarbeit, Westfälische Wilhelms-Universität Münster

ScrumUniMuenster.pdf (Quelle unbekannt; die Datei wurde vom Projektentwicklungslehrer 2014 an der HTL Perg zur Verfügung gestellt)

[7] Scrum-Prozess

ScrumUniMuenster.pdf

[8] Auris IT Consult GmbH (2015): TeamPlay-Homepage

http://teamplay.at

[Zugriff: 20.04.2015]

[9] Adobe Systems (2014): Flash Builder Premium

https://creative.adobe.com/products/flash-builder

[Zugriff: 23.04.2015]

[10] Beispielapplikation Adobe AIR

http://files.zend.com/help/Flash-Builder-for-PHP/Getting-Started/Mobile/mobile_app.png

[Zugriff: 23.04.2015]

[11] Debug-Menü Adobe Flash Builder

http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/air/articles/ane-android-devices/ane-android-devices-fig17.png

[Zugriff: 23.04.2015]

[12] Adobe Systems (2014): Adobe AIR / Technische Daten

http://www.adobe.com/de/products/air/tech-specs.html

[Zugriff: 25.11.2014]

[13] Adobe Systems (2014): Flash Builder 4.7 Premium / Technische Daten

http://www.adobe.com/products/flash-builder/tech-specs.html

[Zugriff: 26.04.2015]

[14] Eclipse Foundation (2015): Eclipse

http://eclipse.org

[Zugriff: 26.04.2015]

[15] Eclipse Foundation (2015): Mindestanforderungen für die Entwicklung mit PhoneGap (Eclipse)

http://www.eclipse.org/downloads/

[Zugriff: 26.04.2015]

[16] Eclipse Foundation (2015): Eclipse Scout

http://eclipse.org/scout/

[Zugriff: 26.04.2015]

[17] Screenshots einer Eclipse Scout App

http://www.eclipse.org/community/eclipse_newsletter/2013/july/images/Helloworld_running.png

[Zugriff: 26.04.2015]

[18] Eclipse Foundation (2015): Mindestanforderungen bei der Eclipse Scout Entwicklung

https://eclipse.org/scout/downloads/

[Zugriff: 26.04.2015]

[19] Embarcadero Technologies Inc. (o. J.): Rad Studio XE8.
https://www.embarcadero.com/de/products/rad-studio
[Zugriff: 22.04.2015]

[20] Screenshot RAD Studio XE4 mit Designer
http://edn.embarcadero.com/article/images/43099/03000015.png
[Zugriff: 22.04.2015]

[21] Houser, Kris (2010): Mindestanforderungen RAD Studio
http://edn.embarcadero.com/article/40774
[Zugriff: 26.04.2015]

[22] Anonymus (2015): ASP.NET
http://de.wikipedia.org/wiki/ASP.NET
[Zugriff: 21.04.2015]

[23] Grundlegender Aufbau von ASP.NET
http://upload.wikimedia.org/wikipedia/commons/thumb/c/cf/ASP.NET_Stack.svg/348px-ASP.NET_Stack.svg.png
[Zugriff: 21.04.2015]

[24] Anonymus (2015): Microsoft Visual Studio
http://de.wikipedia.org/wiki/Microsoft_Visual_Studio
[Zugriff: 21.04.2015]

[25] Download Eclipse
http://www.eclipse.org/downloads/
[Zugriff: 26.04.2015]

[26] Download Adobe AIR
http://get.adobe.com/air/
[Zugriff: 22.04.2015]

[27] Download Adobe Flash Builder
http://www.adobe.com/at/products/flash-builder.html
[Zugriff: 22.04.2015]

[28] Download Android SDK
http://developer.android.com/sdk/index.html
[Zugriff: 22.04.2015]

[29] Anonymus (2013): Strategje zur Behebung des Android-Problems bei Adobe AIR

https://forums.adobe.com/message/5086305

[Zugriff: 10.11.2014]

[30] Yaiser, Michelle (2011): Garbage collection internals for Flash Player and Adobe AIR

http://www.adobe.com/devnet/actionscript/learning/as3-fundamentals/garbage-collection.html

[Zugriff: 12.11.2014]

[31] Vujovic, Max (2011): Flex mobile performance checklist

http://www.adobe.com/devnet/flex/articles/flex-mobile-performance-check-list.html

[Zugriff: 20.01.2015]

[32] Architektur Eclipse Scout

https://www.bsiag.com/scout/wp-content/uploads/2014/06/scout_cloud_archi-tecture.png

[Zugriff: 26.04.2015]

[33] Zimmermann, Matthias (2014): The Eclipse Scout Book v 3.9 (Kepler).

http://tools.bsiag.com/scoutbook/3.9/latest/pdf/book.pdf

[Zugriff: 15.01.2015]

[34] GUI-Designer RAD Studio

http://www.itwriting.com/blog/wp-content/uploads/2014/09/image3.png

[Zugriff: 22.04.2015]

A3 Abbildungsverzeichnis

A4 Tabellenverzeichnis

A5 Abkürzungsverzeichnis

API	Application Programming Interface
ASP.NET	Active Server Pages .NET
CSS	Cascading Style Sheets
DI	Diplomingenieur
DTO	Data Transfer Object
EDV	Elektronische Datenverarbeitung
ERD	Entity Relationship Diagram
EPET	Diplomarbeitstitel (Evaluierung plattformübergreifender Entwicklungstechnologien für TeamPlay)
GB	Gigabyte
GC	Garbage Collector
GmbH	Gesellschaft mit beschränkter Haftung
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTBLA	Höhere technische Bundeslehranstalt
HTL	Höhere technische Lehranstalt
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifikation(snummer)
IDE	Integrated Development Environment
IIS	Internet Information Service
IT	Informationstechnologie
JSON	JavaScript Object Notation
MB	Megabyte
MVC	Model-View-Controller
MVP	Model-View-Presenter
OS	Operating System (Betriebssystem)
PC	Personal Computer
RAP	Remote Application Platform
REST	Representational State Transfer

SDK	Software Development Kit
SWT	Standard Widget Toolkit (for Java)
UI	User Interface (=GUI)
USB	Universal Serial Bus
VM	Virtual Machine
XML	eXtensible Markup Language