HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

finApp - Mobiles E-Banking für das iPhone

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2010

Autoren:	Matthias Good, Philipp Marugg und Sven Lenz
Betreuer:	Prof. Hans Rudin
Projektpartner:	finnova AG Bankware
Gegenleser:	Wolfgang Giersche

finnova

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Auftraggeber und Betreuer

Diese Studienarbeit findet in Zusammenarbeit mit der *finnova AG Bankware* statt.

1.1.1 Ansprechpartner Auftraggeber:

- Diego Stalder, finnova AG Bankware, Abteilungsleiter Retail

1.1.2 Betreuer HSR:

- Prof. Hans Rudin, Institut für Software, hrudin@hsr.ch
- Michael Graf, Assistent am IFS, als Ansprechpartner für die iPhone-Programmierung

1.2 Ausgangslage

Es befindet sich bereits eine iPhone-Applikation in der Entwicklungsphase (durch die finnova AG Bankware). In dieser Applikation wird dem Endbenutzer jedoch nur ermöglicht, Kontodaten zu begutachten und noch keine Transaktionen durchzuführen. Die Migrosbank und die Graubündner Kantonalbank werden als Kunden der finnova AG Bankware diese Applikation benutzen.

1.3 Ziele und Aufgabenstellung

Die zentrale Frage der Studienarbeit lautet:
"Mit welchem UserInterface kann ein mobiler Internet Banking Kunde einfach, intuitiv und effizient Zahlungen erfassen".
Folgende Ziele werden vorgegeben:

- Anforderungen des Bankkunden an ein Zahlungsverkehr-Modul aufnehmen und daraus verschiedene Lösungsansätze evaluieren.
- Entwurf von User-Interface-Prototypen, welche den Kunden vorgelegt werden um daraus die bestmögliche Lösung für die Applikation zu erhalten.
- Entwicklung der iPhone-Applikation
  Hinweis: Der genaue Umfang der zu entwickelnden Applikation muss nach der Spezifizierung der funktionalen Anforderungen noch abgegrenzt werden (z.B. Beschränkung auf die zwei häufigsten Zahlungstypen / ob "Übersicht der pendenten Zahlungen" auch implementiert wird etc.)
- Parallel dazu muss die Serverarchitektur spezifiziert und entwickelt werden. Gegeben ist, dass mit dem Finnova Core über einen Tomcat kommuniziert wird.
  Nach aussen (zum iPhone-Client) handelt es sich um eine XML-Schnittstelle welche mittels XML-Schema beschrieben wird und nach innen(zum Backend) um eine EDBS-Schnittstelle welche mittels Tasks beschrieben wird. Das EDBS ist bereits vorhanden.
  Der Server muss in der Lage sein, einen XML- auf einen EDBS-Request und eine EDBS- auf einen XML-Response abzubilden.
- Da ohne die entsprechende Security keine sicheren Transaktionen über die iPhone-Applikation abgewickelt werden können, gilt es zudem noch abzuklären, was für Security-Komponenten auf dem Markt verfügbar sind. Dazu muss eine Marktstudie

durchgeführt werden, mit dem Ziel in der Theorie aufzuzeigen, dass entsprechende Sicherheitsmechanismen auch integriert werden könnten. Die Implementation eines solchen Security-Moduls wird von dieser Studienarbeit jedoch ausgeschlossen.

1.4 Zur Durchführung

Mit den HSR-Betreuern finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das den Betreuern, dem technischen Berater und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

1.5 Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Dazu gehört insbesondere auch eine Plagiatserklärung. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 3 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen.

1.6 Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html

20.09.10	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch die Betreuer.
20.12.10	Die Studierenden senden folgende Dokumente der Arbeit per Email zur Prüfung an ihre Betreuer: - Abstract/Kurzfassung - A0-Poster Vorlagen stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
23.12.10	Die Studierenden senden das vom Betreuer abgenommene und freigegebene Abstract/Kurzfassung als Word-Dokument an das Studiengangsekretariat (cfurrer(at)hsr.ch).
23.12.10	Abgabe des Berichtes an den Betreuer bis 17.00 Uhr.

1.7 Beurteilung

Eine erfolgreiche Studienarbeit erhält 8 ECTS-Punkten (1 ECTS Punkt entspricht einer Arbeitsleistung von ca. 25 bis 30 Stunden). Für die Modulbeschreibung der Studienarbeit siehe https://unterricht.hsr.ch/staticWeb/allModules/10938_M_SAI.html

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Mgmt Summary, techn. u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 6. Oktober 2010

Der verantwortliche Dozent

Prof. Hans Rudin
Institut für Software
Hochschule für Technik Rapperswil

2 Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Matthias Good, Rapperswil, den 23. Dezember 2010

Philipp Marugg, Rapperswil, den 23. Dezember 2010

Sven Lenz, Rapperswil, den 23. Dezember 2010

3 Geheimhaltunserklärung

Die Implementierung der Business Logik auf dem Finnova Core ist von der Finnova als vertraulich eingestuft worden, weshalb das betroffene PL/SQL Package auf einer separaten CD an Herrn Rudin übergeben wird. Herr Rudin hat dazu eine Geheimhaltungserklärung [ANH11] unterzeichnet und wird die CD nach dem Review vernichten oder Herrn Lenz zurückgeben.

4 Abstract

4.1 Ausgangslage

Mobiles E-Banking befasst sich mit der Umsetzung der bestehenden E-Banking Geschäfte auf mobile Endgeräte. Bestehende E-Banking Geschäfte beinhalten Kundendaten, welche vom Bankserver abgerufen werden können, sowie Transaktionen, welche dem Server übergeben werden können. Die finnova AG Bankware befindet sich bereits in einer ersten Entwicklungsphase einer mobile E-Banking Applikation, die sich jedoch auf das Anzeigen von Kundendaten beschränkt.

4.2 Vorgehen und Technologien

Mit dieser Studienarbeit ist ein Prototyp entstanden, der den Weg in eine zweite Phase des mobilen E-Bankings ermöglicht. Der Prototyp beinhaltet eine funktionsfähige iPhone Applikation (finApp), welche einem Applikations-Server (eMob-Converter) mittels definierten Services Transaktionen in Auftrag geben kann.

In dem der Endbenutzer eine Codierzeile eines Einzahlungsscheins fotografiert und die finApp mittels einer OCR-Erkennung diese Codierzeile aus dem Foto extrahiert, konnte gezeigt werden, dass ohne grossen Aufwand und mit minimaler Benutzerinteraktion eine Zahlung bei der Bank in Auftrag geben werden kann.

Der Security-Aspekt ist in der zweiten Phase des mobilen E-Bankings ein Kernthema, da keine von Drittpersonen eingeschleusten oder manipulierten Transaktionen auf dem System ausgeführt werden sollen. Eine dazu durchgeführte Studie gibt Aufschluss darüber, wie eine Transaktion geschützt werden könnte. Der eMob-Converter ist so konzipiert, dass sicherheitskritische Transaktionen über das bestehende Security-Interface der finnova AG Bankeware signiert werden müssen. Dies gewährleistet, dass alle in der Studie beschriebenen Massnahmen auch angewendet werden können.

4.3 Ergebnis

Mit dieser Studienarbeit konnte dem Auftraggeber aufgezeigt werden, dass Transaktionen auch auf Mobilegeräten einfach und intuitiv erstellt und der Bank in Auftrag gegeben werden können. Durch das Einbinden des bestehenden Security-Interface der finnova AG Bankeware ist zudem sichergestellt, dass nur vertrauenswürdige Transaktionen entgegen genommen werden.

5 Technischer Bericht

5.1 Einleitung

Das Ziel dieser Studienarbeit war eine iPhone-Applikation zu entwickeln, welche in der Lage ist Bankzahlungen an einen eigens implementierten Banken-Server zu schicken, welcher diese Zahlungen in Auftrag bringt.
Sicherheit, Ausbaufähigkeit, Performance und eine wohldefinierte Schnittstelle sowohl zu der finApp als auch zum Finnova Core waren die wichtigsten Ziele serverseitig.

Auf Client-Seite stellten ein intuitiv bedienbares User Interface sowie eine schnelle und einfache Kommunikation mit dem Server die wichtigsten Anforderungen dar. Das Thema Sicherheit für mobiles E-Banking wurde in einer Studie erarbeitet, wurde aber im Rahmen der Studienarbeit nicht implementiert.

5.2 Ergebnisse

5.2.1 Erreichte Ziele

- Implementierung der Grundfunktionen für die iPhone-Applikation
- Implementierung der Grundfunktionen des eMob-Converters
- OCR Erkennung und parsen der Codierzeile
- Orangen Einzahlungsschein beim Finnova Core in Auftrag geben
- Signieren der Zahlung
- Integration des external Security Interface

5.2.2 Erklärung

Die finApp (iPhone-Applikation) ist in der Lage, Zahlungsdaten, welche von Hand eingegeben wurden, an den eMob-Converter (Server) zu senden. Als einziger Zahlungstyp steht der „Einzahlungsschein orange“ zur Verfügung (weitere Typen können äquivalent implementiert werden). Zusätzlich kann mittels der iPhone-Kamera die Codierzeile fotografiert werden. Die in dieser Zeile enthaltenen Informationen werden nach Austausch mit dem Server dem Benutzer in der finApp angezeigt.
Auch das Signieren einer Zahlung konnte implementiert werden. Falls ein Empfänger einer Zahlung nicht in der Whitelist enthalten ist, muss die Zahlung signiert werden. Der Benutzer wird nach dem Absenden der Zahlung aufgefordert, einen Code einzugeben (dieser Code kann z.B. von einer Streichliste stammen).

Der eMob-Converter bietet ein XML-Interface, welches von der finApp angesprochen werden kann. Das bestehende external Security Interface der Finnova wurde ebenfalls eingebunden, um eine sichere Erfassung von Transaktionen zu ermöglichen.

5.2.3 Nicht erreichte Ziele

- Laden der Konti des Benutzers (sowohl in der finApp als auch auf dem Backend)

5.2.4 Ursachen

Während der Entwicklung der finApp und des eMob-Converters wurde laufend festgestellt, dass duplizierter Code vorhanden ist beziehungsweise auftauchen kann. Da wir grossen Wert auf eine Wiederverwendbarkeit und leichte Wartung des Codes gesetzt haben, musste laufend ein Refactoring des Codes vorgenommen werden. Diesen zusätzlichen Aufwand (siehe Abbildung 50 - Ist Zustand/ Punkt Qualitätssicherung), den wir auf uns genommen haben, hat dazu geführt, dass wir aus Zeitmangel die Implementierung der Konti nicht mehr miteinbeziehen konnten. Da diese Kontoliste bereits in der Phase 1 der finApp implementiert

wird und wir dies somit nur bis zur Integration der Phase 2 benötigen, haben wir uns dazu entschieden, eine statische Liste in die finApp zu laden.

5.3 Schlussfolgerung

Sowohl die finApp als auch der eMob-Converter wurden soweit vorbereitet, dass neue Services schnell in die Architektur aufgenommen werden können. Für die Phase 1 der finApp wurden im Dezember 2010 bereits mehrere neue Services auf dem eMob-Converter implementiert. Somit ist der eMob-Converter ab Mitte des nächsten Jahres produktiv im Einsatz. Die Integration des Zahlungsverkehrs ist per Anfang 2012 vorgesehen.

Die finApp und der eMob-Converter wurden der Finnova präsentiert und das Feedback ist sehr positiv ausgefallen. Wir konnten mit einem einfachen User Interface und einer schnellen Kommunikation mit dem Finnova Core punkten.

6 Projektplan

6.1 Einführung

6.1.1 Zweck

Dieses Dokument beschreibt den Projektplan für das die Studienarbeit " finApp – Mobiles E-Banking für das iPhone".

6.1.2 Gültigkeitsbereich

Dieses Dokument gilt als Grundlage für die ganze Studienarbeit und hat deshalb Gültigkeit über die gesamte Arbeitsdauer.

6.1.3 Übersicht

Der Projektplan gibt eine Übersicht über die Projektorganisation, die Zeitplanung und das Risikomanagement. Zudem werden die verschiedenen Arbeitspakete aufgelistet und zum Schluss werden die Infrastruktur und die Qualitätsmassnahmen beschrieben.

6.2 Projektorganisation

6.2.1 Projektmitarbeiter

Das Team besteht aus drei einander gleichgestellten Mitgliedern, wobei jedes Teammitglied für ein spezielles Aufgabengebiet verantwortlich ist.

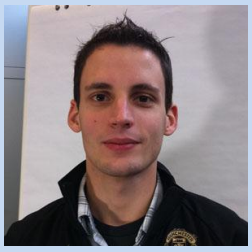
Projektmitglied	Aufgabengebiet	Foto
Matthias Good	Data, Problem Domain	
Philipp Marugg	User Interface, Problem Domain	
Sven Lenz	Server (eMobC)	

Tabelle 1 - Projektmitglieder

Dies ist nur ein kleiner Überblick, für eine genauere Unterteilung dient die Dokumentation im Kapitel 6.4 (Arbeitspakete).

6.2.2 Externe Schnittstellen

Die Studienarbeit wird von Prof. Hans Rudin (Institut für Software, Rapperswil) betreut. Ansprechpartner der finnova AG Bankware ist Diego Stalder.

6.3 Managmentabläufe

6.3.1 Methodik

Die Studienarbeit wird mithilfe von Scrum (Vorgehensmodell für agile Software-Entwicklung) erarbeitet.

Bei Scrum wird grundsätzlich angenommen, dass Produktfertigungs- und Entwicklungsprozesse so komplex sind, dass sie sich im Voraus weder in große abgeschlossene Phasen noch in einzelne Arbeitsschritte mit der Granularität von Tagen oder Stunden pro Mitarbeiter vorher planen lassen. Somit ist es produktiver, wenn sich ein Team in einem festen äusseren Rahmen mit sehr grober Granularität selbst organisiert[01]. Der Scrum-Ablauf sieht im Allgemeinen folgendermassen aus:

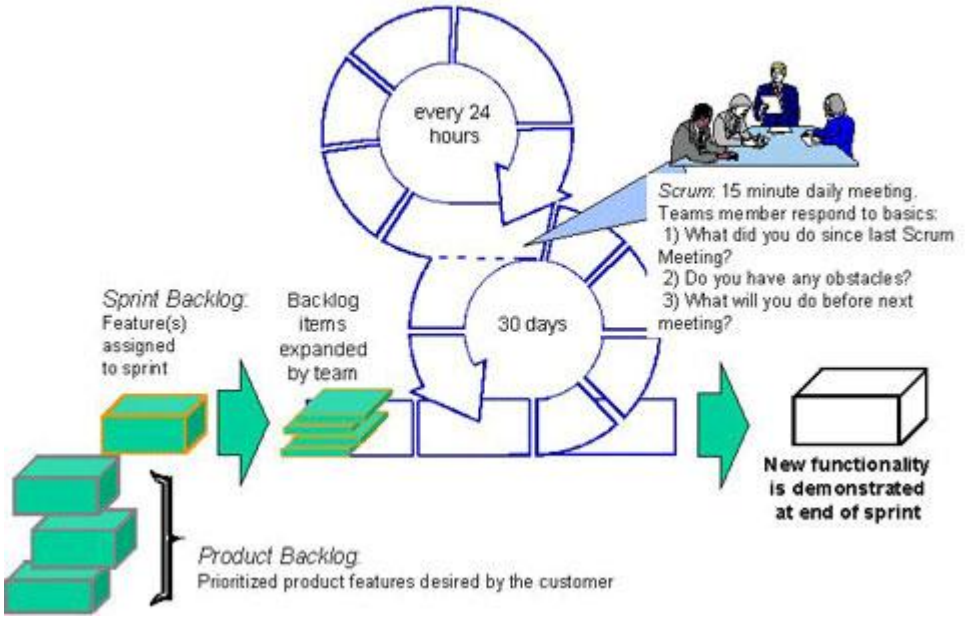


Abbildung 1 - Scrum Ablauf [02]

- Product Backlog:
  Dieses Dokument stellt den Anforderungskatalog des Kunden dar. Vor dem ersten Sprint wird dieser Katalog zusammengestellt und jede Anforderung erhält eine Priorität.
- Sprint Backlog:
  Die Sprint-Backlogs enthalten jeweils die Anforderungen aus dem Product Backlog mit der höchsten Priorität. Vor Sprint-Beginn setzt sich das Team zusammen um die Anforderungen für den Sprint zu deklarieren, d.h. wie viel Anforderungen sollen in diesem Sprint geplant werden.

Das Team hat die Dauer eines Sprints auf zwei Wochen gesetzt, d.h alles zwei Wochen wird ein neuer Sprint Backlog erstellt. Anforderungen, welche in einem Sprint nicht komplett implementiert worden sind, werden in den neuen Sprint miteinbezogen.

Nebst den Dokumenten ist die Kommunikation ein weiterer wichtiger Punkt bei Scrum. Das Team hält jeden Tag ein kleines Meeting ("Daily Scrum", ca 15 Minuten) bei welchem die folgendnen drei Punkte von jedem Teammitglied angesprochen werden:

1. Was habe ich seit dem letzten "Daily Scrum" gemacht?
2. Was werde ich bis zum nächsten "Daily Scrum"-Meeting machen?
3. Was hindert mich an meiner Arbeit (Probleme)?

Zusätzlich zu den täglichen Meetings trifft sich das Team jede Woche mit dem Betreuer zur Besprechung. Dabei präsentiert das Team jeweils die erzielten Resultate und gibt Auskunft über geplante Aufgaben. Ausserdem können dabei Dokumente mit dem Betreuer besprochen werden (diese müssen im Vorfeld an den Betreuer zum Review geschickt werden).

Dieses Treffen findet jeweils am Mittwoch von 15.10 bis 16.10 Uhr im Zimmer 6.111 statt.

6.3.2 Projektplan

Eine übersichtliche Tabelle über die zeitliche Planung des Projekts befindet sich im Anhang [ANH02].

6.3.3 Iterationsplanung

Die Tabelle 2 - Iterationsplanung zeigt einen Überblick über die verschiedenen Iterationsphasen und deren Inhalt.

Iteration	Beschreibung	Ende	Dauer in Wochen
Product Planning	- Projektmanagement (Projektplan, -antrag), - Analyse (Anforderungsspezifikation, - Domainanalyse), - Studien, - Design (Systemarchitekturdokument), - Product Backlog	Woche 06	6
Sprint 1	- Setup der Umgebungen - Implementation der XML-Konverter - OCR-Erkennung	Woche 08	2
Sprint 2	- Setup der Problem Domain - Zahlung übermitteln - Codierzeile parsen - Empfänger in Whitelist suchen - Zahlung validieren und ausführen	Woche 10	2
Sprint 3	- Transaktion signieren - Unit Testing	Woche 12	2
Finishing	- Testing und Korrekturen - Dokumentation	Woche 14	2

Tabelle 2 - Iterationsplanung

6.4 Arbeitspakete

Alle Arbeitspakete werden in Tasks (Projektmanagement, Backlogs, Analyse, Design, Tests, Dokumentation, Studium Technologien und Sitzungen) unterteilt.
Für jedes Paket wird der Inhalt angegeben und wer dafür verantwortlich ist. Zudem wird festgehalten, wie viel Soll-Stunden eingetragen wurden und wie lange
das Team wirklich am Paket gearbeitet hat.

Task	Name	Inhalt	Verant-wortli-cher	Soll in h	Ist in h
Projekt-management	Projektantrag	Projektantrag erstellen und dem Betreuer vorlegen	sl	18	18
	Projektplan	Erstellen des Projektplans	Team	28	31
	Konzept	Erstellung des Konzepts	Team	17	16
	Konfigurations-verwaltung	SVN für die Studienarbeit installie-ren und mittels TortoiseSVN lokal einrichten	mg	1	1
	Review	Review der Dokumente	Team	4	0
Backlogs	Product Backlog Stufe 3	Enthält alle Arten von Anforderungen	sl	7	6.5
	Product Backlog Stufe 2	Detaillierte Auswahl von Anforderungen	sl	3	1
	Product Backlog Stufe 1	Zur Implementierung freigegebene Anforderungen	sl	4	2
	Sprint 1 Backlog	Anforderungen für diesen Sprint	Team	1.5	2
	Sprint 2 Backlog	Anforderungen für diesen Sprint	Team	1.5	1.5
	Sprint 3 Backlog	Anforderungen für diesen Sprint	Team	1.5	1.5
	Review	Review der Dokumente	pm, mg	4	2
Analyse	Anforderungs-spezifikation	Erstellen der Anforderungsspezifi-kation (Funktionale und Nicht-Funktionale Anforde-rungen)	Team	12	14
	Use Cases	Erstellen der Use Cases und des Use Case Model	mg	4	5
	Domain Model	Erstellen des Domain Models	pm	10	5.5
	System Sequenzdiagr.	Erstellen der Systemsequenzdia-gramme	mg	4	2
	Paperprototype	Ausarbeitung der Paperprototypes	Team	18	21

	Studie Security	Erarbeitung einer Studie über Mobile-Banking Security	sl	16	14
	Studie XML	Studie über versch. XML-Parser für das iPhone	mg	6	6
	Studie OCR	Studie über OCR-Verfahren für Texterkennung	pm, mg	24	24
	Review	Review der Dokumente	Team	11	8
Design	Klassen-diagramm	Erstellen des Klassendiagramms	Team	30	22.5
	Systemarchitek-turdokument	Erstellen des SAD (Architekturübersicht, Logische Architektur)	Team	55	33
	Server-architektur	Erstellen des Serverarchitektur-Dokuments	sl	14	14
	Review	Review der Dokumente	Team	2	2
Sprints	Sprint 1	Erste Implementierungsphase	Team	60	75
	Sprint 2	Zweite Implemenentierungsphase	Team	62	71
	Sprint 3	Letzte Implementierungsphase	Team	60	56
Tests	System Tests	Funktionale Systemtests für den Standardablauf definieren	pm, mg	7	7
Dokumentation	Administration		pm	9	5
	Schnittstellen-dokumentation	Definiert die Schnittstellen des Servers	sl	8	7
	Bericht	Finale Dokumentation	Team	82	88
Studium Techn.	Einarbeitung Objective-C	Einarbeitung in die Programmiersprache Objective-C 2.0	pm, mg	38	37
	Einarbeitung iPhone-Prog.	Einarbeitung in die iPhone-Progammierung	mg, mg	12	20
Sitzungen	Daily Scrum	Tägliches Team-Meeting	Team	42	49
	Scrum Meeting		Team	3	3
	Meeting mit Betreuer	Wöchentliches Meeting mit dem Betreuer	Team	42	39
Qualtitäts-sicherung	Code Review	Bugfixing, Refactoring	Team	8	42

Tabelle 3 - Tasks

6.5 Risikomanagement

Risiko Bewertungen								
Risk ID	**Risiko**	**Auswirkung**	**Massnahme**	**Kosten der Massnahmen in h**	**Max. Schaden in h**	**Wahrschein-lichkeit des Eintreffens**	**Gewichteter Schaden in h**	**Priorität**
R01	Probleme mit verwendeten Technologien	Verzögerung im Projektverlauf	Gute Einarbeitung in Objective C / iPhone-Programmierung	**5**	20	25%	**5**	**hoch**
R02	Unzureichende Dokumentation der Technologien	Längere Einarbeitungszeit	Weitere Dokumentationen/ Beispiele suchen und studieren. Bücher bestellen.		10	20%	**2**	**mittel**
R03	Ausfall eines Teammitglieds infolge Krankheit/Unfall	Verzögerung im Projektverlauf	Zeitreserve einplanen		20	10%	**2**	**mittel**
R04	Wissen konzentriert auf einzelne Person	Einzelne Aufgaben können nicht erfüllt werden	Wissen weitergeben in Sitzungen	**4**	10	10%	**1**	**niedrig**
R05	Probleme mit Konfigurationsverwaltung	Teile der Arbeit gehen verloren.	Einsatz von Versionierungssystem	**2**	10	10%	**1**	**niedrig**
R06	Arbeitgeber zieht Projekt zurück	Fehlende Unterstützung (Core-DB, Server)	Gute Kommunikation mit dem Arbeitgeber		100	3%	**3**	**niedrig**
R067	Anforderungen des Projekts ändern sich	Überarbeitung von Projektdokumenten	Regelmässige Sitzungen / Zeitreserve einplanen		10	20%	**2**	**hoch**
	Total Kosten in Arbeitspaketen enthalten			**11**				
	Total Rückstellungen				**180**		**16**	

finnova

6.6 Infrastruktur

6.6.1 Hardware

Die HSR stellt dem Projektteam während der Dauer der Studienarbeit freundlicherweise folgende Geräte zur Verfügung:

- 2x iMac 24"
- 1x iPhone 3G
- 1x iPhone 4G
- 2x iPod Touch

Ausserdem besitzt jedes Teammitglied ein privates MacBook. Zusätzlich kann auch auf die HSR-Arbeitsplatzrechner zurückgegriffen werden.

Die Netzwerkinfrastruktur sowie der SVN-Server werden ebenfalls von der HSR bereitgestellt.

6.6.2 Software

- Betriebssystem:
  - Mac OS X "Snow Leopard" (Entwicklung)
  - Microsoft Windows 7 Enterprise (Dokumentation)
- Programmiersprache:
  - Objective-C 2.0 (iPhone-App)
  - Java (Server)
- Entwicklungsumgebung:
  - XCode 3.2.4
  - Eclipse Helios
- Versionsverwaltungssoftware:
  - SVN
- Dokumentation:
  - Atlassian Confluence 3.3.3

6.7 Qualitätsmassnahmen

6.7.1 Dokumentation

Das Team sorgt dafür, dass die Dokumente stets auf dem aktuellen Stand sind.
Ausserdem werden diese sowohl von einem Teammitglied als auch vom Betreuer gegengelesen.

6.7.2 Sitzungsprotokolle

Alle Meetings mit dem Betreuer werden protokolliert. Somit ist garantiert, dass Kritik, Anregungen und Abmachungen schriftlich festgehalten sind und Missverständnisse minimiert werden.

6.7.3 Daily Scrum Meeting

Durch den täglichen Austausch unter den Teammitgliedern ist jedes Mitglied zu jeder Zeit über den aktuellen Projektstand informiert.

6.7.4 Zeitplanung

Die Zeitplanung wird im Projektplan nachgeführt. Darin ist der Vergleich von Soll- und Ist-Zustand ersichtlich, welcher das Team analysiert und gegebenenfalls Massnahmen durchführt.

6.7.5 Versionsverwaltungssystem

Als Versionsverwaltungssystem wird der von der HSR zur Verfügung gestellte SVN-Server verwendet.
Das Repository befindet sich in svn://svns.hsr.ch/finApp, die Zugriffsrechte werden durch Matthias Good verwaltet.

6.7.6 Codierrichtlinien

Bei der Programmierung der iPhone-Applikation wird sich das Team an die von Apple vorgegebenen Codierrichtlinien halten.
Eigene Richtlinien sind im Kapitel 9 (Software Architektur Design) definiert.

6.7.7 Unit-Tests

Während der Implementierung werden wo möglich und sinnvoll Unit-Tests durchgeführt. Diese Tests müssen erfolgreich durchlaufen, bevor der Quellcode wieder eingecheckt werden darf.
Die Tests werden mithilfe des GHUnit-Framework [03] erstellt, welches das Testen direkt auf dem iPhone zulässt.

Abbildung 2 - Unit Tests

6.7.8 Usability-Tests

Externe Benutzer testen die Prototypen und geben ihr Feedback dazu ab. Dieses Feedback dient dem Team als Input für weitere Projektiterationen.

Im Rahmen dieser Arbeit wurde ein Interview mit den Bankkunden der Finnova durchgeführt, welches im Kapitel 7.4.1 (Interview) zu finden ist.

7 Analyse

7.1 Anforderungsspezifikation

7.1.1 Einführung

7.1.1.1 Zweck

Dieses Dokument beschreibt die Anforderungen für das Projekt "finApp – Mobiles E-Banking für das iPhone".

7.1.1.2 Gültigkeitsbereich

Das Dokument ist für die komplette Studienarbeitdauer gültig.

7.1.1.3 Übersicht

Die Anforderungsspezifikation beinhaltet die funktionalen und nicht-funktionalen Anforderungen.

7.1.2 Spezifische Anforderungen

Im Folgenden werden die funktionalen und nichtfunktionalen Anforderungen, welche aus der Befragung von Finnova [04] und den Kunden der Finnova resultieren [05], aufgelistet.

7.1.2.1 Funktionale Anforderungen

Anforderungen an die finApp

Grundlegend wird von der Bank gewünscht, dass Zahlungsvorlagen sowohl auf dem iPhone als auch im Internetbanking (IB) erfasst und ausgeführt werden können.
Im Internetbanking hat der User bereits heute die Möglichkeit, beim Erfassen einer Zahlung diese als Vorlage in einem selbstgewählten Ordner abzuspeichern. Auf dem iPhone soll der User auch manuell eine Zahlungsvorlage erfassen können.
Eine Zahlungsvorlage beinhaltet folgende Attribute

- Den Zahlungstyp. Mögliche Typen sind:
  - Oranger Einzahlungsschein
  - Roter Einzahlungsschein
  - Bankzahlung Inland
  - Bankzahlung Ausland
  - IPI-Beleg Inland
  - IPI-Beleg Ausland
- Das Belastungskonto
- Das Begünstigtenkonto
- Die Adresse des Begünstigten
- Betrag der Zahlung
- Referenznummer (Oranger Einzahlungsschein)
- Mitteilungen

Aus einer Zahlungsvorlage kann der User eine Zahlung generieren. Die Zahlung enthält als zusätzliches Attribut das Ausführungsdatum.

Eine weitere Anforderung ist, dass orange und rote Einzahlungsscheine mit der im iPhone integrierten Kamera fotografiert werden können und die Codierzeile mittels optischer Zeichenerkennung (OCR) ausgelesen werden kann. Aus der Codierzeile können alle Angaben ermittelt werden, um eine Zahlung zu generieren. In untenstehender Abbildung ist ersichtlich, wo sich die Codierzeile für den Zahlungstyp befindet. Beim Ausführen dieser Zahlung soll der User zudem noch die Option erhalten, diese Zahlung als Zahlungsvorlage abzuspeichern.

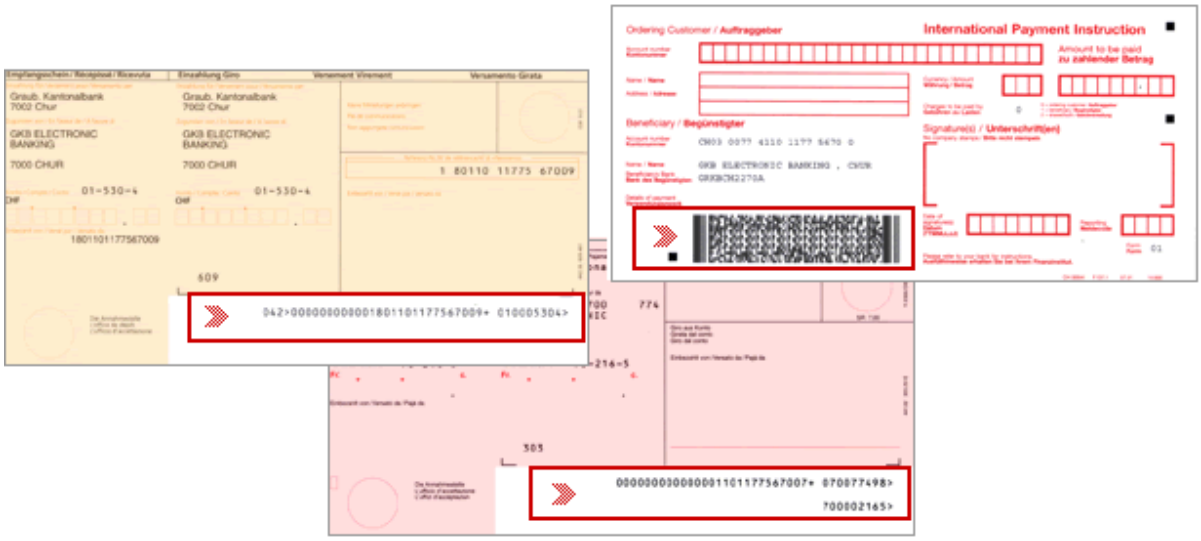


Abbildung 3 - Einzahlungsscheine

Bei der Ausführung einer Zahlungen ist gefordert, dass der Zahlungsempfänger in der bestehenden lokalen oder globalen Whitelist enthalten ist, ansonsten darf die Zahlung nicht übermittelt werden. Diese beiden Whitelists werden bereits heute von der Finnova und den Banken gepflegt und müssen nicht neu implementiert werden.

Der Kontoübertrag wird einfacher behandelt. Dies bedeutet, dass der User für einen Übertrag keine Zahlungsvorlagen auswählen kann, sondern dass direkt eine separate Maske angeboten wird, in der Belastungs- und Begünstigtenkonto ausgewählt, Betrag sowie Ausführungsdatum gesetzt werden können.

Pendente Zahlungen müssen in der Applikation aufgelistet werden, um eine bestehende Zahlung zu bearbeiten oder zu löschen. Verarbeitete Zahlungen müssen in einer Übersicht und als Detail angezeigt werden

Die finApp soll nun um eine Komponente Zahlungsverkehr (ZV) erweitert werden, welche folgende Module enthält:

- Zahlungsvorlagen
- Zahlungen
- Kontoübertrag
- Belegleser (OCR oranger und roter Einzahlungsscheine)

Erweiterbarkeit

Die Grundfunktionen der finApp bestehen aus der Navigation der Views, der Erfassung einer neuen Zahlung (nur mit orangem oder rotem Einzahlungsschein), sowie einer Übersicht über pendente und abgeschlossene Zahlungen. Die Applikation bietet dazu auch die nötigen Schnittstellen für den Server. Zukünftig sollte es möglich sein, die finApp mit weiteren Funktionen zu ergänzen (z.B. Zahlungen mit anderen Zahlungstypen, Kontoübersicht, Zahlungen per Vorlage).

Anforderungen an den eMob-Converter

Um eine Kommunikation zwischen Core-Datenbank der finnova und der finApp zu ermöglichen, übernimmt ein Tomcat die Aufgabe eines eMob-Konverters. Der Konverter muss folgende zwei Schnittstellen implementieren:

- XML-Schnittstelle, welche mittels XML-Schema beschrieben wird , als Interface zur finApp
- EDBS-Schnittstelle, welche mittels Tasks beschrieben wird, als Interface zur Core Datenbank

Der Konverter muss in der Lage sein, einen XML- auf einen EDBS-Request und eine EDBS- auf einen XML-Response abzubilden.

Der EDBS-Server und die Core-Datenbank sind bereits bestehende Komponenten und müssen nur um neuen EDBS-Services erweitert werden.

Technische Anforderungen

- Die einzelnen Services müssen über Servlets aufgerufen werden können
- Automatischer Build in der Entwicklungsumgebung mittels Ant-Tasks
- Automatische Distributionserstellung für die Auslieferung an die Kunden (Zip-File, welches das war-Archive, Installations-Skripte und Templates für die Konfiguration enthält)
- Die Distribution muss von den Kunden parametrierbar und installierbar sein
- Anbindung des bestehenden external Security Interface

7.1.3 Funktionalitätsübersicht

7.1.3.1 Muss-Anforderungen

- Implementierung der Grundfunktionen für die iPhone-Applikation
- Implementierung der Grundfunktionen des eMob-Converters
- OCR Erkennung und parsen der Codierzeile
- Orangen Einzahlungsschein beim Finnova Core in Auftrag geben

7.1.3.2 Optionale Anforderungen

- Signieren der Zahlung
- Integrations des external Security Interface

7.1.3.3 Nichtfunktionale Anforderungen

Zuverlässigkeit

Die finApp ist eine Applikation, in der hohe Geldbeträge überwiesen und sensible Daten ausgetauscht werden. Bei der Ausführung von Transaktionen und der Anzeige von sensiblen Daten ist keine Fehlertoleranz erlaubt.

Um die Wiederherstellbarkeit zu gewährleisten, muss darauf geachtet werden, dass jegliche Interaktionen eines Users mit der Core-Datenbank geloggt werden um jederzeit Nachvollziehen zu können, welche Transaktionen ein User in einer Session erledigt hat.

Benutzbarkeit

Die finApp läuft auf einem Mobilgerät und bietet nicht dieselbe Konformität die sich ein User einer Internet- oder Desktopapplikation gewöhnt ist. Deshalb gilt es, dass Userinterface einfach und die Bedienung intuitiv zu halten.

Leistung und Effizienz

Da auf Mobilgeräten wie dem iPhone Anfrage/Antwort-Verzögerungen (Round-Trip-Time, RTT) relativ hoch sind, ist die Granularität der Anfragen grundsätzlich so zu wählen, dass möglichst viele Daten in eine Antwort gepackt werden und ein Nachladen nur für Detail-Daten notwendig ist.

Portierbarkeit und Übertragbarkeit

Die finApp wird von mehreren Banken betrieben. Den Banken muss somit eine Möglichkeit geboten werden, die Applikation dem Corporate Design(CD) anzupassen. Die finApp wird erst von den Banken dem CD angepasst und als separate Applikation in dem App-Store veröffentlicht

Ebenfalls soll gewährleistet werden, dass in Zukunft auch Applikationen, welche unter einem anderen Betriebssystem entwickelt worden sind, Zugriff auf die Services haben. Da als Übertragungs-Protokoll XML verwendet wird, können alle Geräte, die XML unterstützten, auf die Services zugreifen.

Sicherheitsanforderungen

Die Sicherheitsanforderungen an eine iPhone Applikation für Internet-Banking sind sehr hoch, da sensitive Daten über das Wide Area Network (WAN) oder das Datennetz des Telefonbetreiber übertragen werden oder das Mobilgerät eines User in falsche Hände gerät.

Um sicherzustellen, dass keine sensitiven Daten abgegriffen werden können, werden die Daten nie in Klartext übermittelt, sondern jeweils die ganze Kommunikation verschlüsselt. Ebenfalls werden keine sensitiven Daten auf dem Mobilgerät gespeichert.

Korrektheit

Ergebnisse müssen jederzeit fehlerfrei sein. Deshalb dürfen in der finApp die Daten welche vom Server geliefert werden nicht mehr modifiziert werden.

7.2 Use Cases

7.2.1 Use Case Model

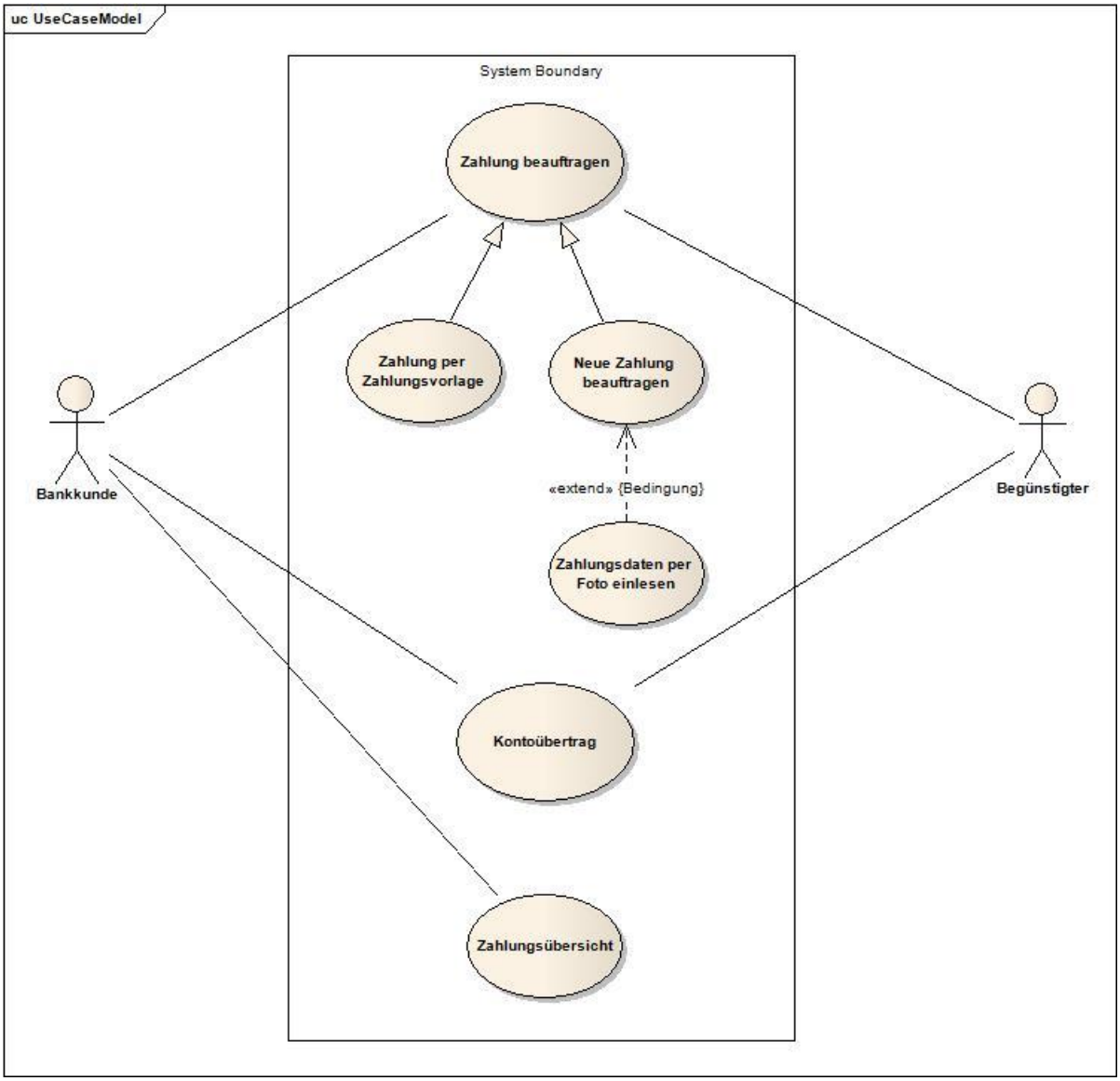


Abbildung 4 - Use Case Model

Bankkunde:	Dieser entspricht dem iPhone-Benutzer, welcher die finApp und darauf einer der Use Cases ausführt.
Begünstigter:	Der Begünstigte ist diejenige Person, Firma, welche die Zahlung des Bankkunden erhält.
System Boundary:	Die Systemgrenzen beinhalten sowohl die iPhone-App (finApp), als auch den Finnova Core.

Das Use Case Model enthält folgende sechs Use Cases:

- **UC01_Zahlung beauftragen**: Der Bankkunde beauftragt eine Zahlung.
- **UC02_Zahlung per Zahlungsvorlage:** Der Bankkunde beauftragt eine Zahlung mittels Vorlage, welche bereits über ausgefüllte Felder verfügt.
- **UC03_Neue Zahlung beauftragen**: Der Bankkunde wählt eine leere Zahlung aus und füllt die benötigten Felder selber aus.
- **UC04_Zahlungsdaten per Foto einlesen**: Extend vom UC03, der Bankkunde schiesst ein Foto der Codierzeile der Zahlung, wodurch die Felder automatisch ausgefüllt werden
- **UC05_Kontoübertrag**: Der Bankkunde überträgt einen Geldbetrag von einem seiner Konten auf ein anderes eigenes Konto.
- **UC06_Zahlungsübersicht**: Der Bankkunde hat die Übersicht über pendente und bereits verarbeitete Zahlungen.

7.2.2 UC01_Zahlung beauftragen

7.2.2.1 Brief

Ein Bankkunde erhält per Post einen roten/orangen Einzahlungsschein und möchte die Zahlung per iPhone abwickeln.

7.2.2.2 Fully Dressed

Primary Actor

Bankkunde

Stakeholders and Interests

Bankkunde (iPhone-Besitzer):

- Bequeme, sichere und schnelle Ausführung der Zahlung.
- Speicherung der Zahlung als Vorlage

Backend:

- Daten des Einzahlungsscheins
- Zahlung, welche vom Bankkunden übermittelt wird

Preconditions

- Der Bankkunde hat einen gültigen Account für die finApp.

Postconditions

- Die Zahlung wurde erfolgreich in Auftrag gegeben.
- Die Zahlung wurde, falls vom Kunden gewünscht, als Vorlage abgespeichert.

Main Success Scenario

1. Der Bankkunde gibt die Daten des Einzahlungsscheins auf dem iPhone ein.
2. Der Bankkunde sendet die ausgefüllte Zahlung an den Finnova Core.
3. Der Finnova Core validiert die Zahlung und gibt diese in Auftrag
4. Der Bankkunde speichert die Zahlung zusätzlich als Vorlage ab.

Extensions

3a. Der Finnova Core kann die Zahlung nicht validieren.

1. Der Finnova Core gibt das falsch ausgefüllte Feld zurück.
2. Der Bankkunde korrigiert das angegebene Feld.
3. 1-2 wiederholt, bis alle Felder korrekt ausgefüllt sind, dann wird die Zahlung in Auftrag gegeben

3b. Der Finnova Core kann die Zahlung nicht validieren.

1. Der Finnova Core gibt an, dass die Zahlung noch signiert werden muss
2. Der Bankkunde gibt den geheimen Code ein.
3. Der Finnova Core validiert die Signierung.
4. 3-4 wiederholt, bis Signierung korrekt, dann wird die Zahlung ausgeführt.

4a. Der Bankkunde will die Zahlung nicht als Vorlage speichern:

1. Der Finnova Core verwirft die Zahlung und wechselt zurück auf die Hauptseite.

Technology and Data Variations List

iPhone (inkl. Kamera für das Fotografieren des Einzahlungsscheins)

7.2.3 UC02_Zahlung per Zahlungsvorlage

7.2.3.1 Brief

Der Bankkunde wählt eine Zahlungsvorlage aus und erfasst eine Zahlung anhand von dieser Vorlage.

7.2.3.2 Fully Dressed

Primary Actor

Bankkunde

Stakeholders and Interests

Bankkunde (iPhone-Besitzer):

- Schnelles Auffinden der gewünschten Zahlungsvorlage.
- Sichere Ausführung der Zahlung.

Preconditions

- Der Bankkunde hat einen gültigen Account für die finApp.

Postconditions

- Die Zahlung wurde erfolgreich in Auftrag gegeben.

Main Success Scenario

1. Der Bankkunde öffnet die Übersicht aller Zahlungsvorlagen und filtert diese nach dem gewünschten Ordner.
2. Der Bankkunde öffnet den gefilterten Zahlungsvorlagen-Ordner und filtert diesen nach der gewünschten Zahlungsvorlage.
3. Der Bankkunde wählt die gefilterte Zahlungsvorlage aus.
4. Der Bankkunde schickt die ausgefüllte Zahlung an den Finnova Core.
5. Der Finnova Core validiert die Zahlung und gibt diese in Auftrag.

Extensions

1a. Der gewünschte Ordner ist nicht vorhanden:

1. Der Bankkunde erstellt einen neuen Ordner mit diesem Namen.
   1a. Der Bankkunde filtert nach einem anderen Ordner

2a. Die gewünschte Zahlungsvorlage ist im ausgewählten Ordner nicht vorhanden:

1. Der Bankkunde erstellt eine neue Zahlungsvorlage.
   1a. Der Bankkunde filtert nach einer anderen Zahlungsvorlage.

5a. Der Finnova Core kann die Zahlung nicht validieren.

1. Die Finnova Core gibt das falsch ausgefüllte Feld zurück.
2. Der Bankkunde korrigiert das angegebene Feld.

3. 1-2 wiederholt, bis alle Felder korrekt ausgefüllt sind, dann wird die Zahlung in Auftrag gegeben

5b. Der Finnova Core kann die Zahlung nicht validieren.

1. Der Finnova Core gibt an, dass die Zahlung noch signiert werden muss
2. Der Bankkunde gibt den geheimen Code ein.
3. Der Finnova Core validiert die Signierung.
4. 3-4 wiederholt, bis Signierung korrekt, dann wird die Zahlung in Auftrag gegeben

7.2.4 UC03_Neue Zahlung beauftragen

7.2.4.1 Brief

Ein Bankkunde erhält per Post einen roten/orangen Einzahlungsschein und möchte die Zahlung per iPhone abwickeln.

Dabei entscheidet er sich, einen leeren Einzahlungsschein von Hand auszufüllen

7.2.4.2 Fully Dressed

Primary Actor

Bankkunde

Stakeholders and Interests

Bankkunde(iPhone-Besitzer):

- Bequeme, sichere und schnelle Ausführung der Zahlung.

Backend:

- Daten des Einzahlungsscheins
- Zahlung, welche vom Bankkunden übermittelt wird

Preconditions

- Der Bankkunde hat einen gültigen Account für die finApp.

Postconditions

- Die Zahlung wurde erfolgreich in Auftrag gegeben.

Main Success Scenario

1. Der Bankkunde füllt die Angaben per Hand in eine leere Einzahlungsschein-Vorlage ein.
2. Der Bankkunde schickt die ausgefüllte Zahlung an den Finnova Core
3. Danach validiert der Finnova Core die Zahlung und gibt diese in Auftrag.

Extensions

3a. Der Finnova Core kann die Zahlung nicht validieren.

1. Die Finnova Core gibt das falsch ausgefüllte Feld zurück.
2. Der Bankkunde korrigiert das angegebene Feld.
3. 1-2 wiederholt, bis alle Felder korrekt ausgefüllt sind, dann wird die Zahlung in Auftrag gegeben

3b. Der Finnova Core kann die Zahlung nicht validieren.

1. Der Finnova Core gibt an, dass die Zahlung noch signiert werden muss

2. Der Bankkunde gibt den geheimen Code ein.
3. Der Finnova Core validiert die Signierung.
4. 3-4 wiederholt, bis Signierung korrekt, dann wird die Zahlung in Auftrag gegeben

Technology and Data Variations List

iPhone (inkl. Kamera für das Fotografieren des Einzahlungsscheins)

7.2.5 UC04_Zahlungsdaten per Foto einlesen

7.2.5.1 Brief

Ein Bankkunde erhält per Post einen roten/orangen Einzahlungsschein und möchte die Zahlung per iPhone abwickeln.

Dabei entscheidet er sich, den Einzahlungsschein zu fotografieren, sodass die Daten per OCR eingelesen werden.

7.2.5.2 Fully Dressed

Primary Actor

Bankkunde

Stakeholders and Interests

Bankkunde (iPhone-Besitzer):

- Bequeme, sichere und schnelle Ausführung der Zahlung.

Backend:

- Daten des Einzahlungsscheins
- Zahlung, welche vom Bankkunden übermittelt wird

Preconditions

- Der Bankkunde hat einen gültigen Account für die finApp.

Postconditions

- Die Zahlung wurde erfolgreich in Auftrag gegeben.

Main Success Scenario

1. Der Bankkunde fotografiert mit seiner iPhone-Kamera die Codierzeile des Einzahlungsscheins
2. Der Finnova Core validiert die erhaltene Codierzeile und gibt die daraus generierten Daten zurück
3. Der Bankkunde erhält eine Übersicht über alle Angaben und füllt gegebenenfalls noch die leeren Felder.
4. Der Bankkunde schickt die ausgefüllte Zahlung an den Finnova Core.
5. Der Finnova Core validiert die Zahlung und gibt diese in Auftrag.

Extensions

2a. Der Finnova Core kann die Codierzeile nicht validieren.

1. Die Finnova Core gibt einen Fehler an den Bankkunden zurück.
2. Der Bankkunde fotografiert die Codierzeile ein weiteres Mal.
3. 1-2 wiederholt, bis die Finnova Core die Codierzeile validieren kann.

2b. Der Finnova Core kann die Codierzeile nicht validieren.

1. Der Finnova Core gibt einen Fehler an den Bankkunden zurück.
2. Der Bankkunde gibt die Daten per Hand ein.

5a. Der Finnova Core kann die Zahlung nicht validieren.

1. Der Finnova Core gibt das falsch ausgefüllte Feld zurück.
2. Der Bankkunde korrigiert das angegebene Feld.
3. 1-2 wiederholt, bis alle Felder korrekt ausgefüllt sind, dann wird die Zahlung in Auftrag gegeben.

5b. Der Finnova Core kann die Zahlung nicht validieren.

1. Der Finnova Core gibt an, dass die Zahlung noch signiert werden muss
2. Der Bankkunde gibt den geheimen Code ein.
3. Der Finnova Core validiert die Signierung.
4. 3-4 wiederholt, bis Signierung korrekt, dann wird die Zahlung in Auftrag gegeben.

Technology and Data Variations List

iPhone (inkl. Kamera für das Fotografieren des Einzahlungsscheins)

7.2.6 UC05_Kontoübertrag

7.2.6.1 Brief

Der Bankkunde überweist einen Betrag von einem seiner Konten auf ein anderes Konto, welches ebenfalls ihm gehört.

7.2.6.2 Fully Dressed

Primary Actor

Bankkunde

Stakeholders and Interests

Bankkunde (iPhone-Besitzer):

- Schnelle und sichere Übeweisung

Preconditions

- Der Bankkunde hat einen gültigen Account für die finApp.

Postconditions

- Die Übertragung wurde erfolgreich in Auftrag gegeben

Main Success Scenario

1. Der Bankkunde wählt die Ansicht für die Kontoübertagung.
2. Der Bankkunde wählt das Belastungskonto aus.
3. Der Bankkunde wählt das Begünstigtenkonto aus.
4. Der Bankkunde gibt den Betrag an und wählt das Ausführungsdatum aus.
5. Der Finnova Core überprüft die Übertragung und gibt diese in Auftrag.

Extensions

5a. Der Finnova Core führt die Übertragung nicht aus, weil der Betrag nicht gültig ist:

1. Die Finnova Core zeigt den ungültigen Betrag an.
2. Der Bankkunde korrigiert diese Angabe.

Weiter mit 5. aus dem Main Success Scenario

7.2.7 UC06_Zahlungsübersicht

7.2.7.1 Brief

Der Bankkunde will eine Übersicht über alle seine pendenten Zahlungen, resp. über die bereits ausgeführten Zahlungen.

7.2.7.2 Fully Dressed

Primary Actor

Bankkunde

Stakeholders and Interests

Bankkunde (iPhone-Besitzer):

- Schnelle und sichere Übeweisung

Preconditions

- Der Bankkunde hat einen gültigen Account für die finApp.

Postconditions

- Der Bankkunde konnte alle Zahlungen (pendent/ausgeführt) einsehen

Main Success Scenario

1. Der Bankkunde wählt die Ansicht für die Zahlungsübersicht.
2. Der Bankkunde wählt eine Zahlung aus und gelangt in die Detailview.

Extensions

2a. Der Bankkunde möchte eine pendente Zahlung bearbeiten

1. Der Bankkunde korrigiert/ergänzt die Angaben der ausgewählten Zahlung
2. Der Bankkunde schickt die ausgefüllte Zahlung an den Finnova Core
3. Der Finnova Core validiert die Zahlung und gibt diese in Auftrag.

7.3 Domainanalyse

7.3.1 Einführung

7.3.1.1 Zweck

Dieses Dokument beschreibt die Analyse der Domain für das Projekt "finApp – Mobiles E-Banking für das iPhone".

7.3.1.2 Gültigkeitsbereich

Das Dokument ist für die komplette Projektdauer gültig.

7.3.1.3 Übersicht

Die Domainanalyse beschreibt im ersten Teil die Domain anhand des Domain Models. Danach folgen die einzelnen System Sequenzdiagramme und das User Environment Model

7.3.2 Domain Model

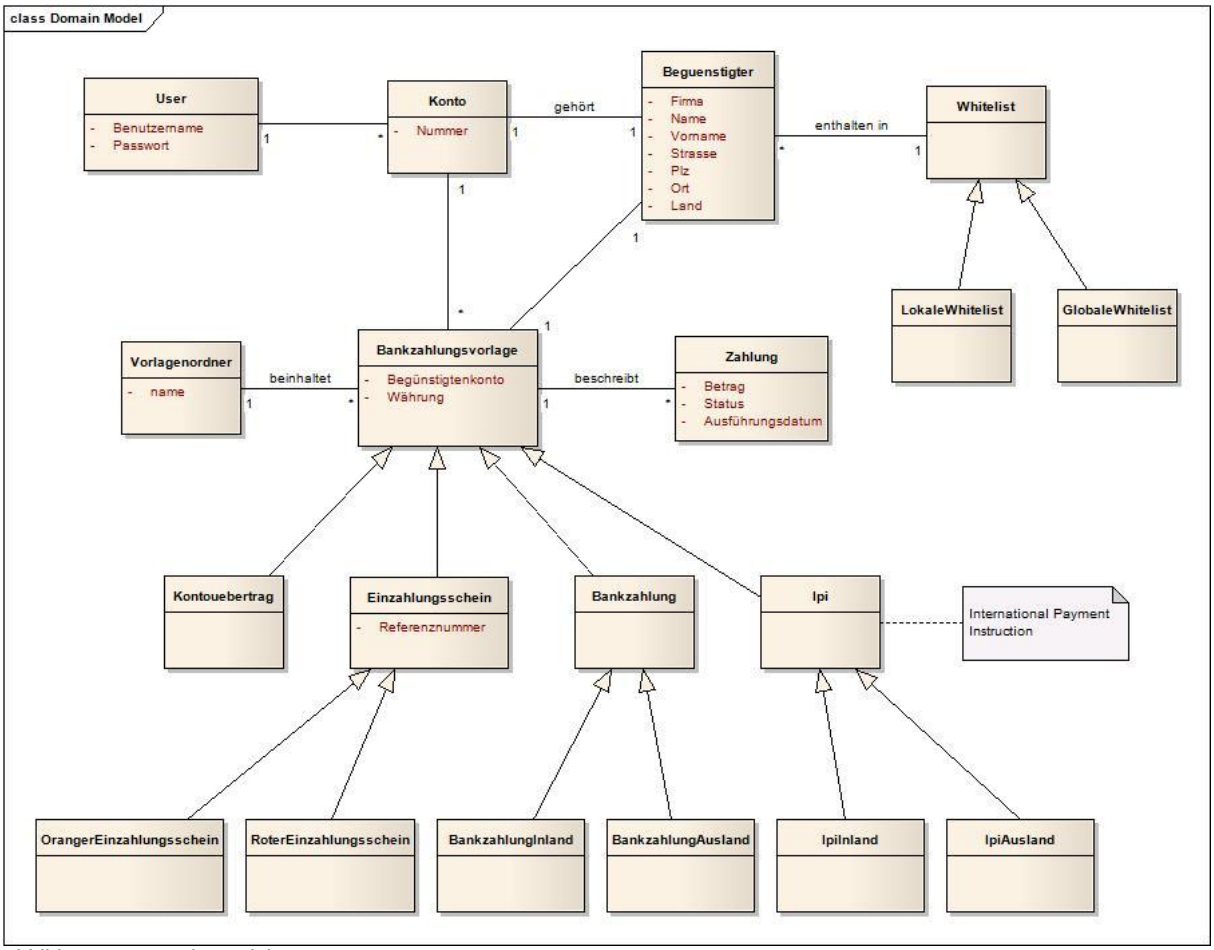


Abbildung 5 - Domain Model

Das DomainModel stellt die gesamte Funktionalität der iPhone-Applikation dar. In der finApp wurde jedoch nur die Zahlung mittels orangen Einzahlungsscheins implementiert.

7.3.2.1 User

Der User meldet sich mittels Benutzernamen und Passwort am System an. Dem Benutzernamen können vom System mehrere Konten zugeordnet werden.

7.3.2.2 Begünstigter

Der Begünstigte ist entweder eine Bank/Firma oder eine Person mit einer zugehörigen Adresse. Jeder Begünstigte hat genau ein Konto und für jeden Begünstigten kann es genau eine Zahlungsvorlage geben.

7.3.2.3 Konto

Ein Konto wird mittels einer Nummer eindeutig identifiziert und kann über Zahlungsvorlagen enthalten. Optional gibt es für jedes Konto mehrere Meldungen über Neuigkeiten.

7.3.2.4 Zahlung

Eine Zahlung hat einen Betrag und einen Status (pendent / erledigt), eine Zahlung kann als Zahlungsvorlage gespeichert werden.

7.3.2.5 Bankzahlungsvorlage

Eine Zahlungsvorlage bildet ein Template für alle verfügbaren Zahlungstypen wie Einzahlungsschein, IPI, Bankzahlung sowie für den Kontoübertrag.

7.3.2.6 Vorlageordner

Ein Vorlageordner beinhaltet Bankzahlungsvorlagen. Unterordner sind nicht möglich.

7.3.2.7 Whitelist

In der Whitelist sind Begünstigte enthalten, welche als vertrauenswürdig gelten. Dabei unterscheidet man zwischen einer globalen und einer lokalen Whitelist. Die globale Whitelist wird von der Bank für alle Kunden geführt, währendem die globale Whitelist vertragsbezogen für einen einzelnen Bankkunden geführt wird.

7.3.3 Systemsequenzdiagramme

7.3.3.1 SSD Zahlung beauftragen

Das Interaktionsdiagramm Abbildung 6 visualisiert den Vorgang, wenn vom User eine Zahlung beim Finnova Core in Auftrag gegeben wird.

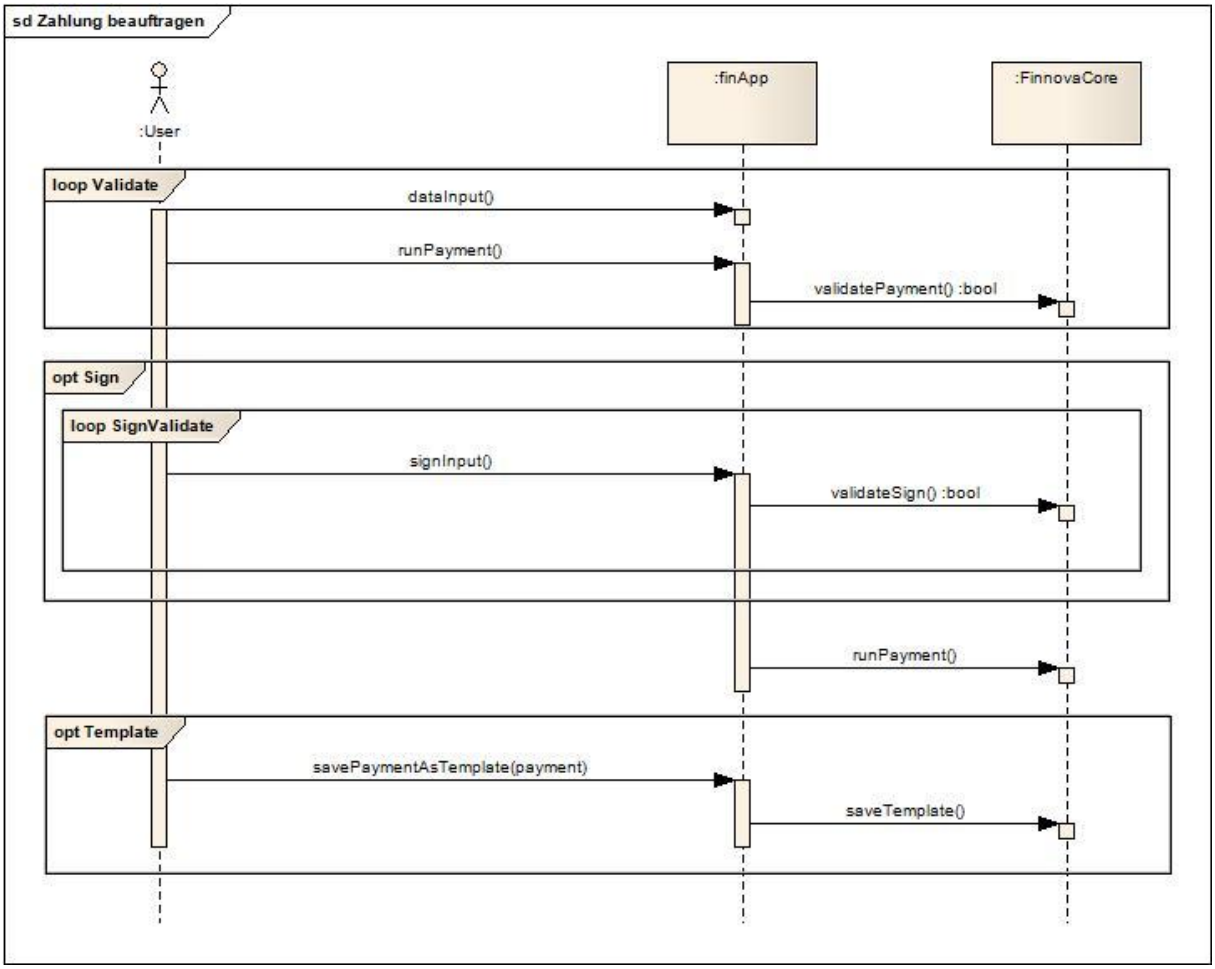


Abbildung 6 - SSD_Zahlung beauftragen

7.3.3.2 SSD Zahlung per Zahlungsvorlage

Das Interaktionsdiagramm Abbildung 7 visualisiert den Vorgang, wenn vom User aus einer Zahlungsvorlage eine Zahlung generiert wird. Optional können Ordner und Vorlagen neu erstellt werden.

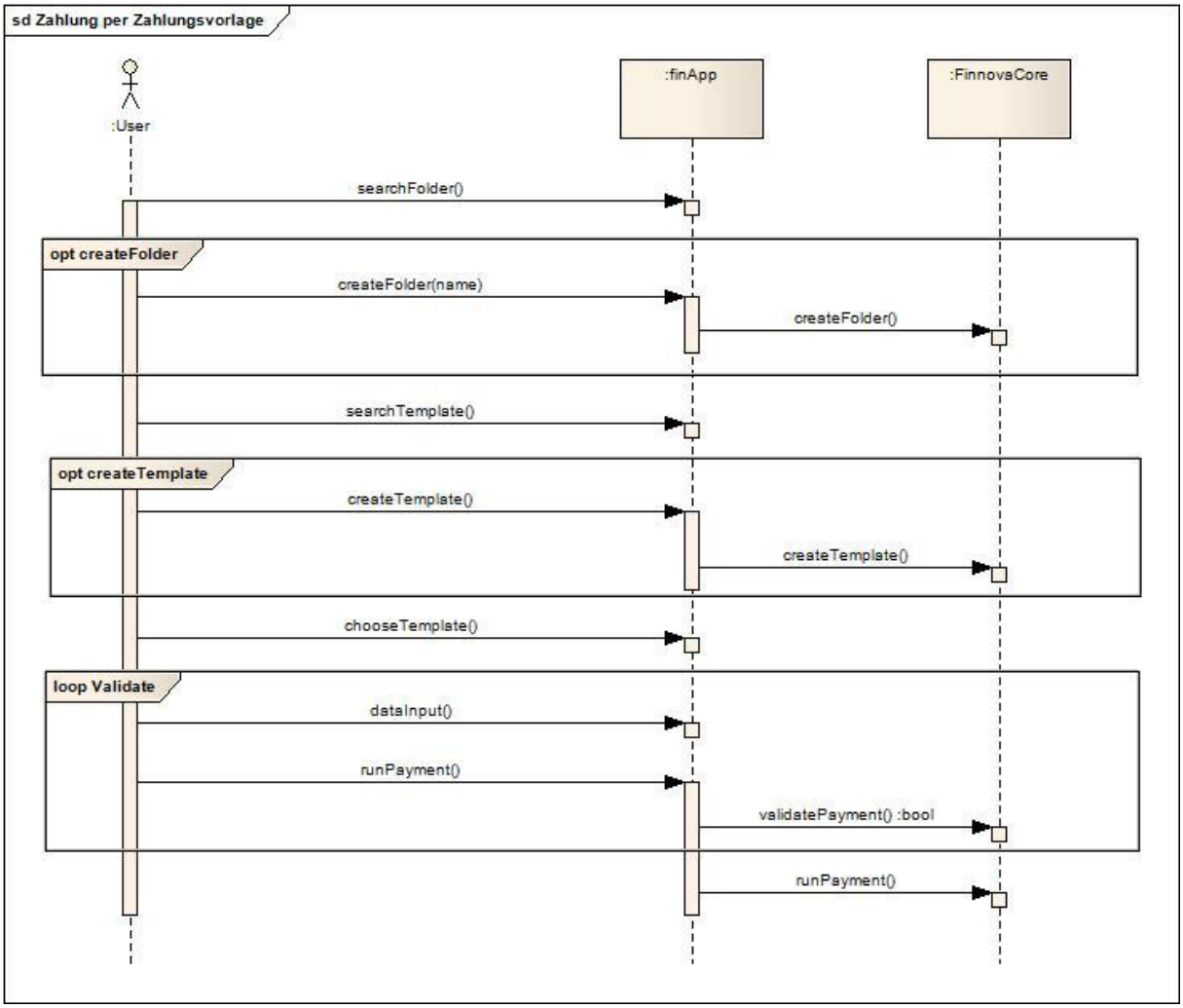


Abbildung 7 - SSD_Zahlung per Zahlungsvorlage

7.3.3.3 SSD Zahlungsdaten per Foto einlesen

Das Interaktionsdiagramm Abbildung 8 visualisiert den Vorgang, wenn vom User die Codierzeile eines Einzahlungsscheins fotografiert und dem Finnova Core zum Parsen übergeben wird. Ist das Parsen erfolgreich und die fehlenden Daten wurden vom Benutzer eingegeben, wird die Zahlung dem Finnova Core in Auftrag gegeben.

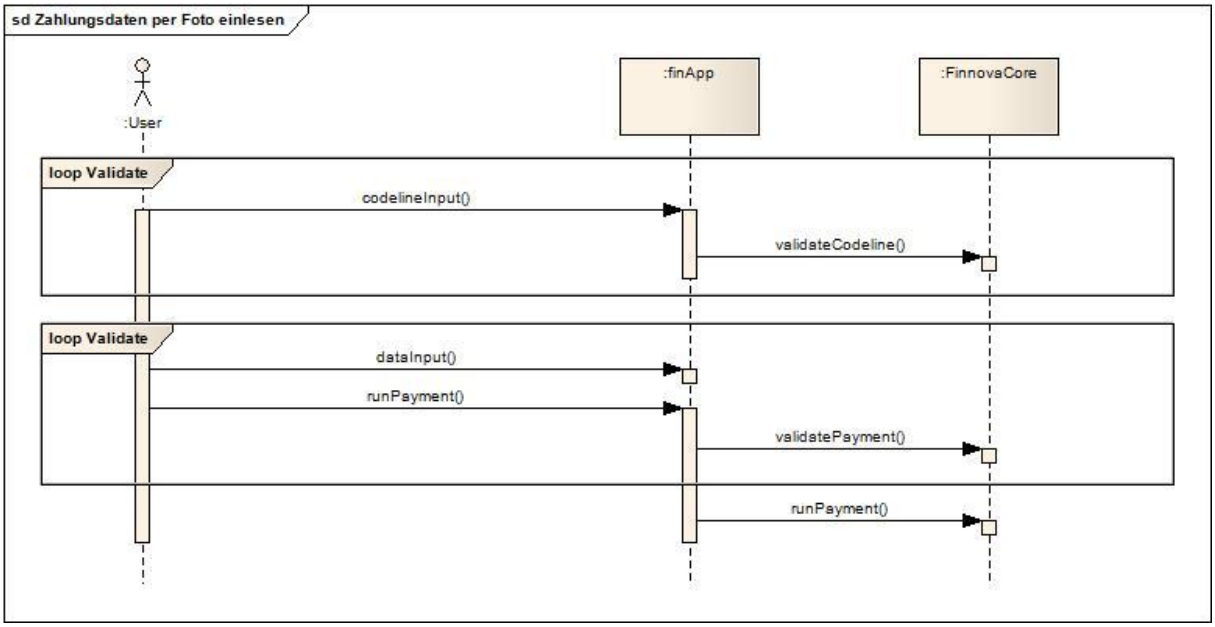
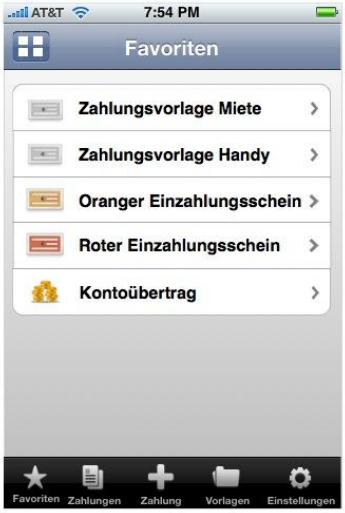
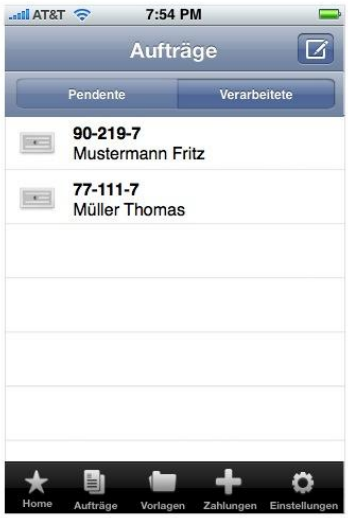
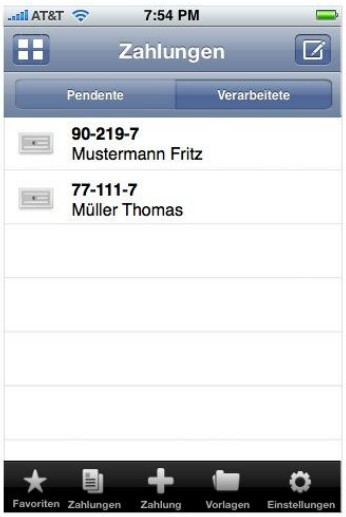
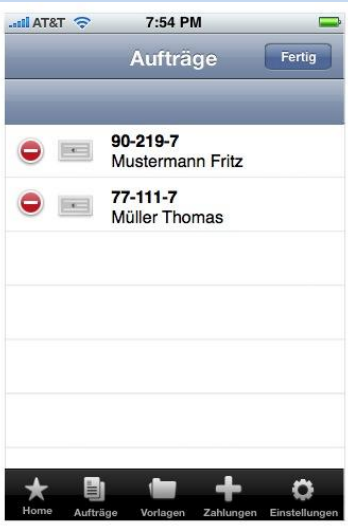
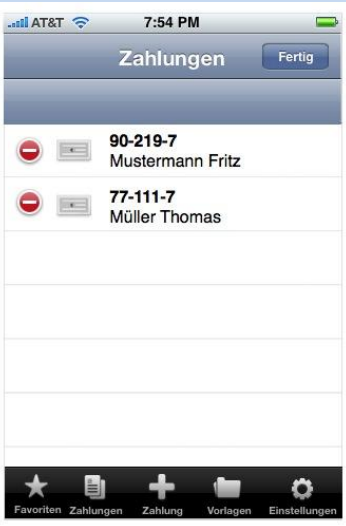
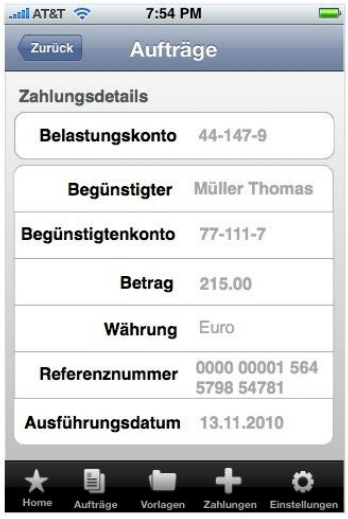


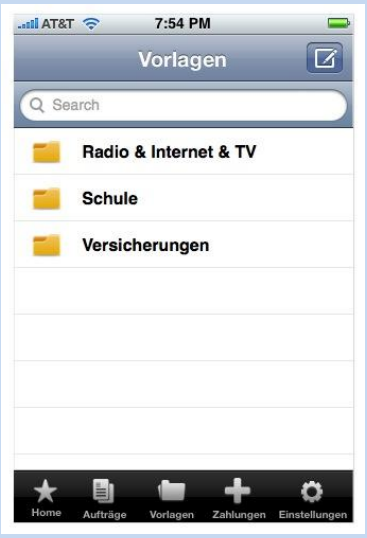
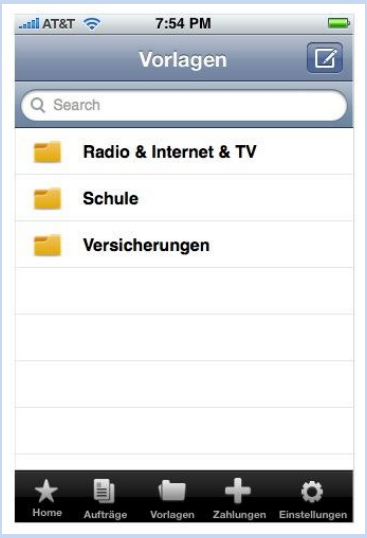
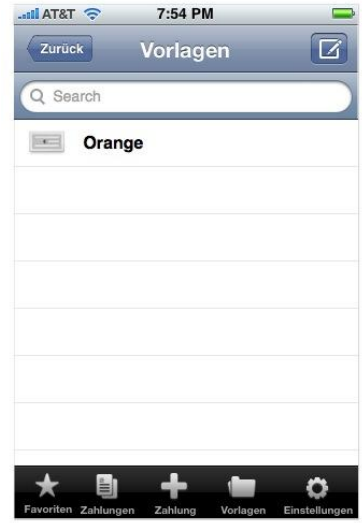
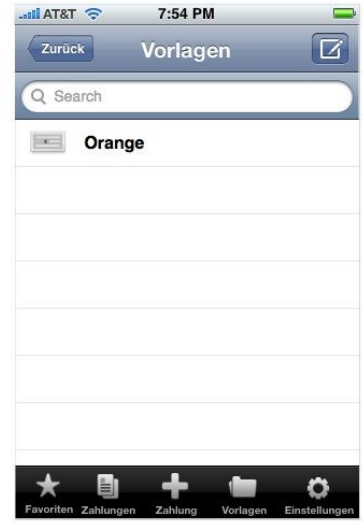
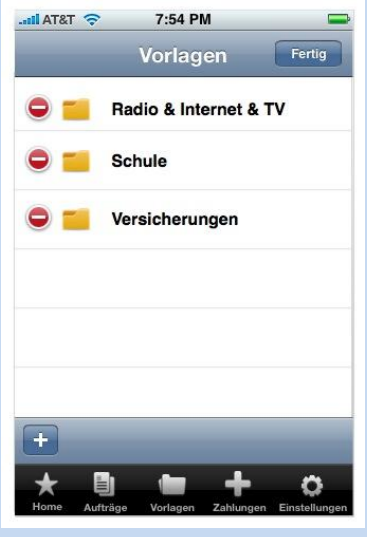
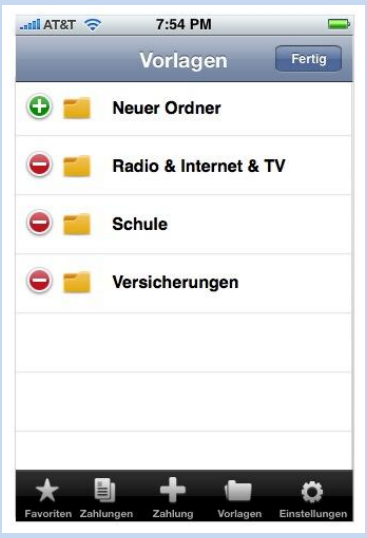
Abbildung 8 - SSD_Zahlungsdaten per Foto einlesen

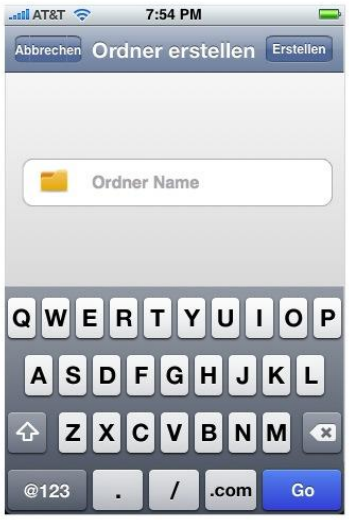
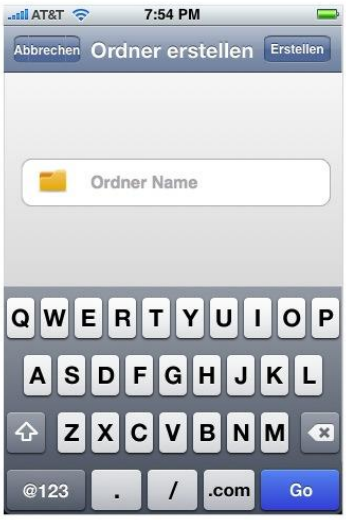
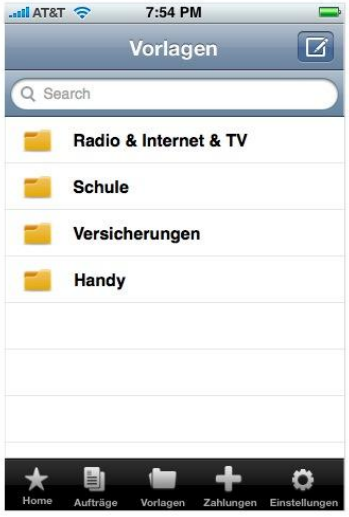
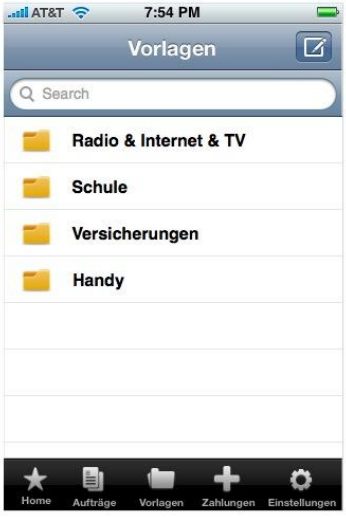
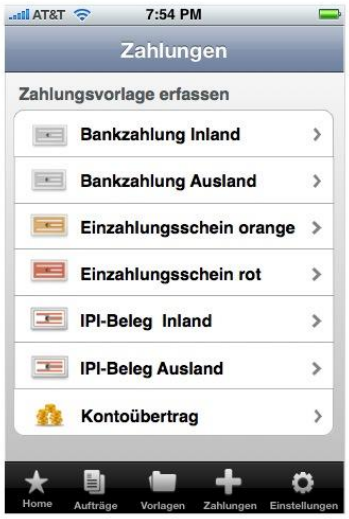
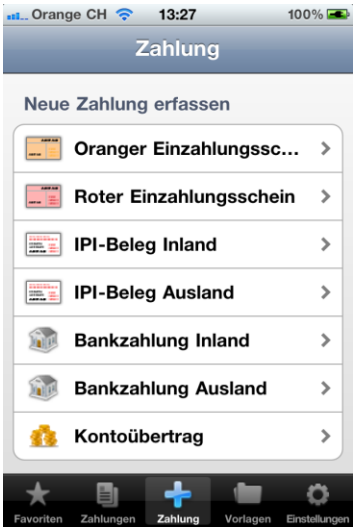
7.4 Paperprototyping

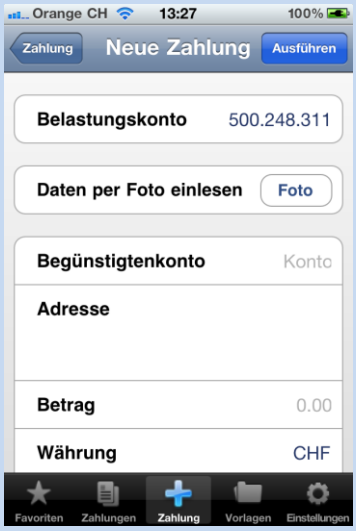
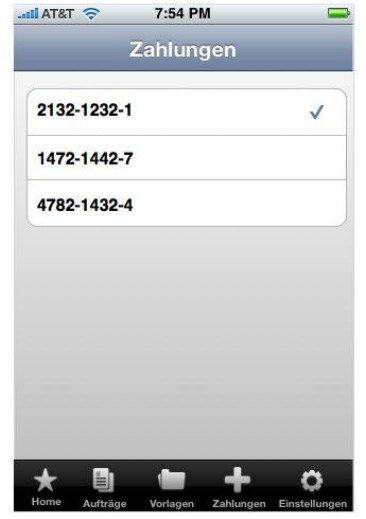
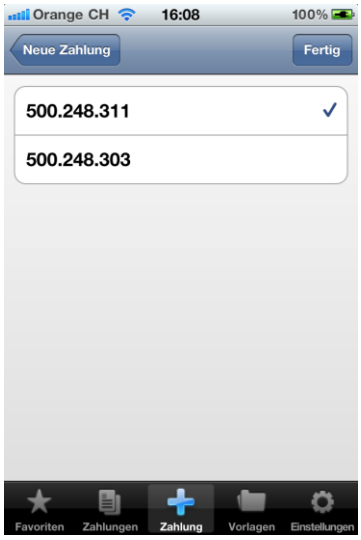
Die folgende Tabelle gibt einen Überblick über die Entwicklung der finApp. In der linken Spalte befinden sich die Paperprototypes. Diese wurden zwei Kunden vorgelegt, eine kurze Zusammenfassung ihres Feedbacks befindet sich in der rechten Spalte. In der mittleren Spalte werden die finalen Screenshots der finApp angezeigt.

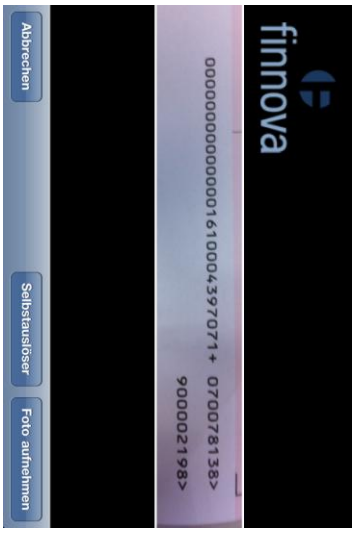
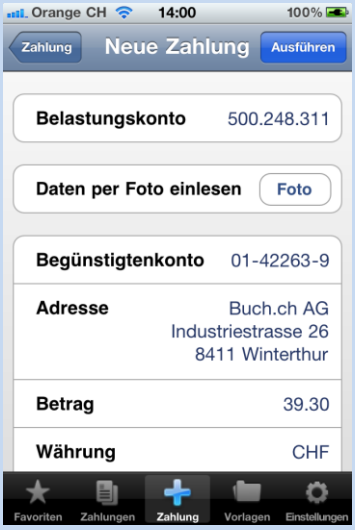
Paperprototypes	Finales Layout	Feedback
		Der Homescreen mit Favoriten ist ok, die Meldungen werden jedoch nicht benötigt.
	View entfällt	Meldungen sind in diesem Stil nicht erwünscht

		Die Begriffe sollten sich an das bestehende e-banking anlehnen (bspw. Aufträge in Zahlungen ändern).
		
		Begünstigter allein reicht noch nicht. Es muss ein Feld für die komplette Adresse des Begünstigten vorhanden sein.

		
		
		Die Position des + Buttons (Vorlagenordner hinzufügen) sollte überdacht werden.

		
		
		Die Namen der Zahlungsvorlagen stimmen nicht mit den Namen bei der finnova AG überein.

		Begünstigter allein reicht noch nicht. Es muss ein Feld für die komplette Adresse des Begünstigten vorhanden sein.
		
	View entfällt	Die Option Bilder aus dem Foto-Archiv zu verwenden ist überflüssig, da dies nicht genutzt wird und zudem ein Sicherheitsrisiko darstellt, wenn Daten auf dem iPhone gespeichert werden.

		Das Einblenden eines Logos wäre von Vorteil.
		Begünstigter allein reicht noch nicht. Es muss ein Feld für die komplette Adresse des Begünstigten vorhanden sein.
		

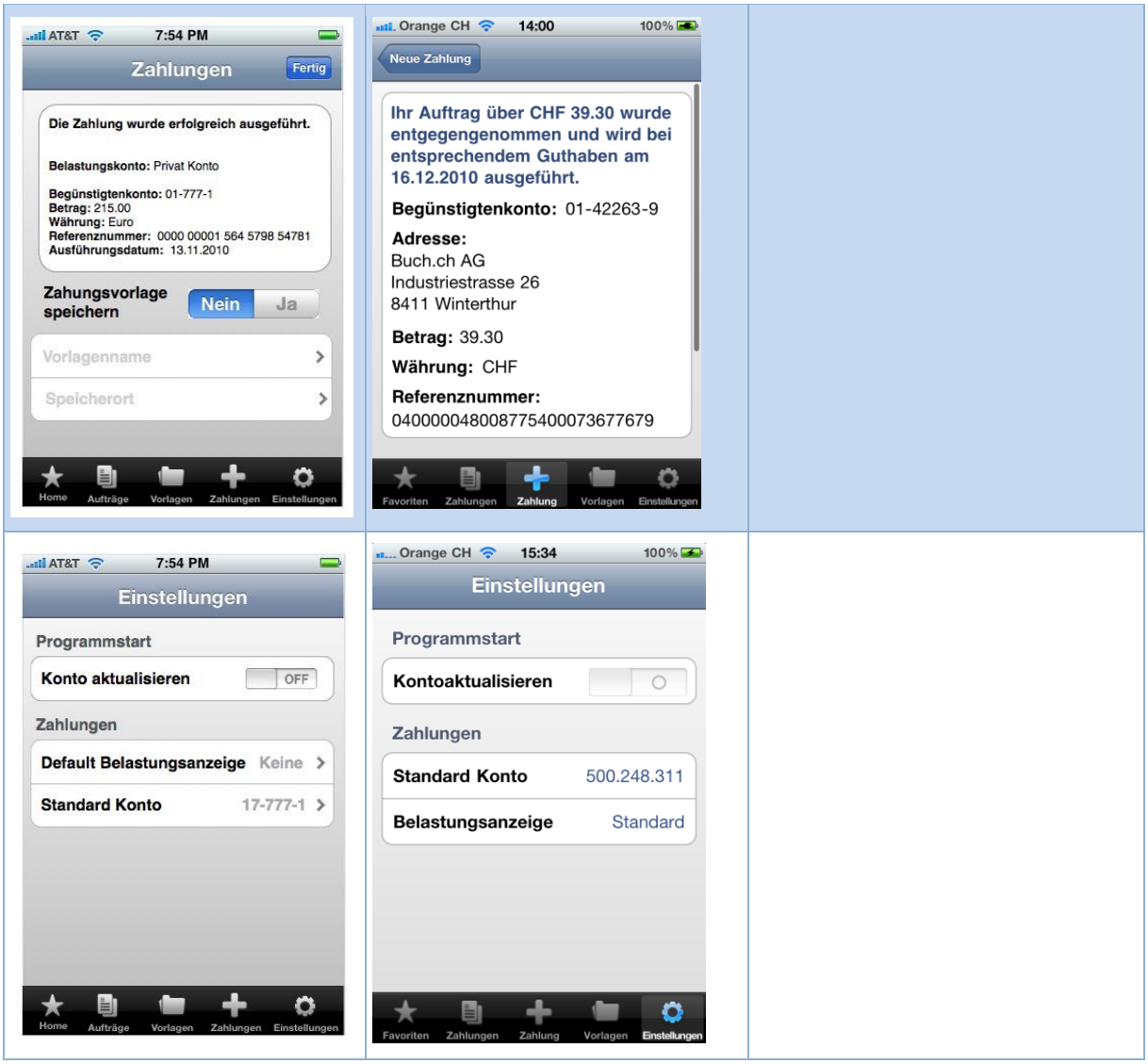


Tabelle 4 - Paperprototyping

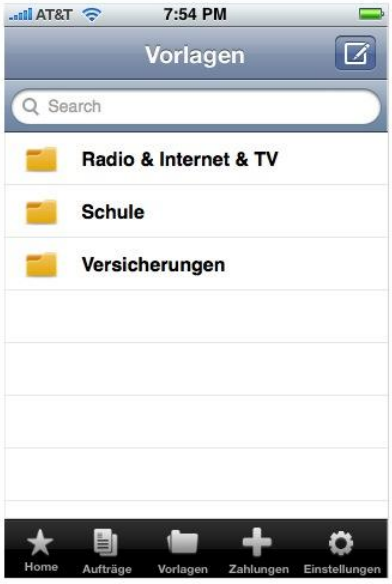
7.4.1 Interview

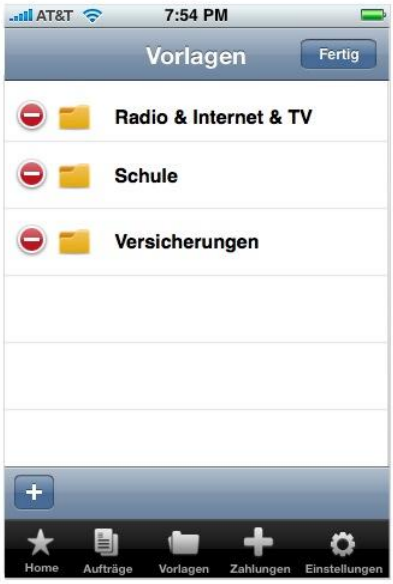
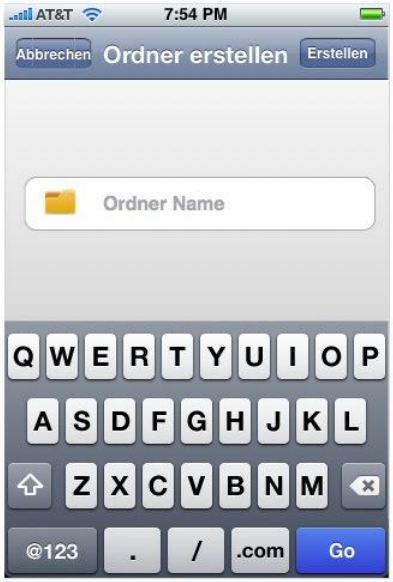
Den Bankkunden wurden die nachfolgenden Szenarien und den Interviewbogen vorgelegt. Das Feedback ist jeweils direkt in den Kapiteln 7.4.1.1 (Szenarien) und 7.4.1.2 (Fragebogen zu den Paperprototypes und den Szenarien) ersichtlich.

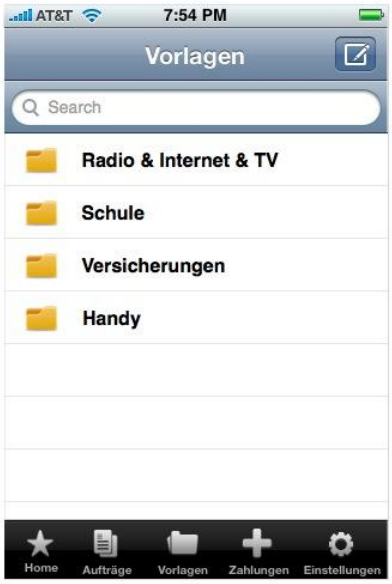
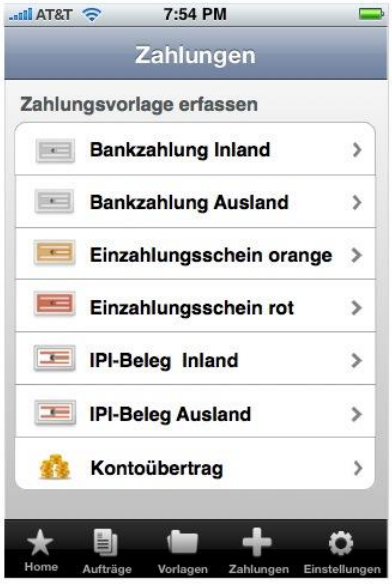
7.4.1.1 Szenarien

Szenario 1

Norbert Natel hat soeben seine monatliche Handy-Rechnung von seinem Anbieter bekommen. Da er nicht mehr jedes Mal dieselben Eingaben für diese Rechnung tätigen will, erstellt er eine Vorlage mithilfe der finApp.

	Dazu startet Norbert die finApp auf seinem iPhone und bekommt als Erstes den Homescreen zu sehen. Da er für diese Zahlungsvorlage einen neuen Vorlagen-Ordner erstellen will, wählt er den Menüpunkt „Vorlagen“ aus der Tab-Bar aus.
	Mit einem Klick auf den Edit-Button gelangt er in den Bearbeitungs-Modus der Ordner.

	Hier kann er nun einen neuen Ordner hinzufügen (durch das Anklicken des „+“-Buttons).
	Im „Ordner erstellen“-Screen gibt er den gewünschten Namen ein (z.B. Handy) und erstellt den Ordner mittels „Erstellen“-Button.

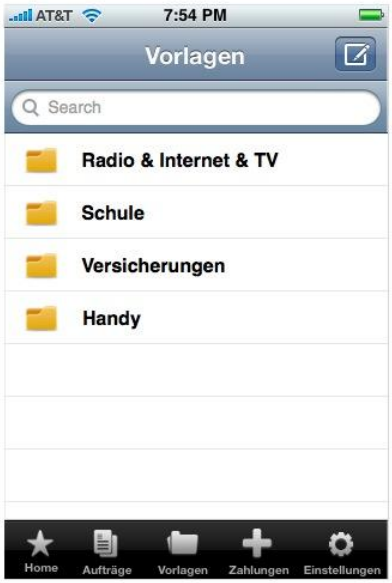
	Nun ist der Ordner erstellt. Norbert kann nun eine Zahlungsvorlage erstellen. Dazu wählt er den Menüpunkt „Zahlungen“ aus.
	Hier wählt er den gewünschten Zahlungstyp aus. In seinem Fall handelt es sich um einen „Einzahlungsschein orange“.

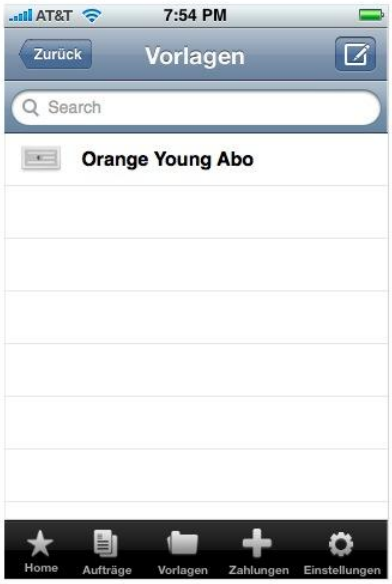
	Es erscheint der Zahlungen-Screen, wo er die Daten des Einzahlungsscheins eingeben kann. Als Erstes gibt Norbert das Belastungskonto an.
	

	Die restlichen Angaben, wie Begünstigtenkonto, Betrag, Währung und Referenznummer füllt er von Hand aus. Zum Schluss gibt er noch das Ausführungsdatum aus. Nun muss Norbert die Zahlung nur noch ausführen (mittels Klick auf den „Ausführen“-Button).
	Nach der Ausführung der Zahlung erhält Norbert einen Überblick über die soeben getätigte Zahlung. Zudem hat er nun die Möglichkeit die Zahlung als Zahlungsvorlage zu speichern, er klickt dafür auf den „Ja“-Button und wählt als Speicherort den von ihm erstellten Ordner „Handy“ aus. Als Vorlagenname gibt er „Orange Young Abo“ an. Die Zahlung ist nun ausgeführt und als Zahlungsvorlage in dem von ihm erstellten Ordner gespeichert.

Szenario 2

Einen Monat später erhält Norbert Natel wieder eine Handy-Rechnung. Da er bereits eine Zahlungsvorlage dafür erstellt hat, kann er die Rechnung in wenigen Schritten ausführen.

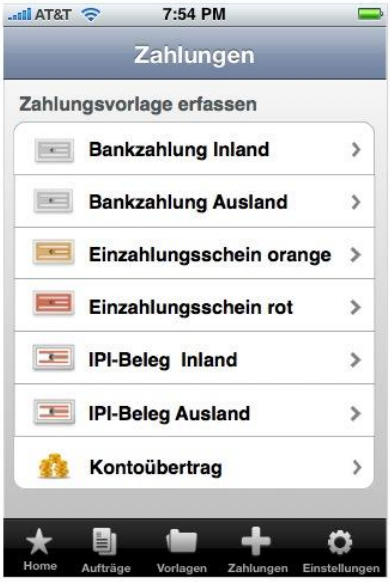
	Die finApp gestartet, befindet er sich auf dem Home-Screen. Hier wählt er den Menüpunkt „Vorlagen“ aus.
	Aus den Vorlagen-Ordner wählt er den gewünschten Ordner aus, in seinem Fall den „Handy“-Ordner.

	Darin befindet sich die von ihm erstellte Zahlungsvorlage „Orange Young Abo“. Norbert wählt diese aus.
	Die Zahlung beinhaltet bereits alle Angaben, welche er in der Vorlage gespeichert hat. Norbert muss nun nur noch den Betrag ändern und das Ausführungsdatum auswählen und kann dann die Zahlung ausführen.

Er erhält wiederum eine Zusammenfassung der getätigten Zahlung und klickt nun einfach auf „Fertig“, da er die Zahlung bereits als Vorlage gespeichert hat.

Szenario 3

Bernhard Buch hat sich bei einer Online-Bücherei ein Buch bestellt. Zusammen mit dem Paket erhält er einen roten Einzahlungsschein. Da ihm das Abtippen der Angaben mit seinem iPhone zu mühsam ist und ausserdem zu lange dauert, will er den Einzahlungsschein fotografieren.

	Bernhard startet die finApp auf seinem iPhone und wählt auf dem Home-Screen den Menüpunkt „Zahlungen“ aus.
	Es erscheint ein Überblick über alle möglichen Zahlungstypen. Er wählt den „Einzahlungsschein rot“ aus.

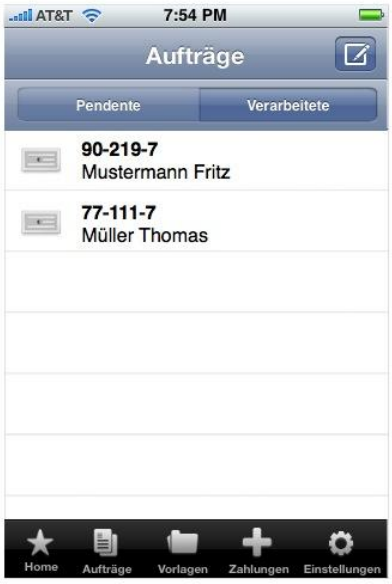
	Nachdem Bernhard das Belastungskonto ausgewählt hat, entscheidet er sich für die Möglichkeit durch einen Klick auf den „Foto“-Button die Daten des Einzahlungsscheins per Foto einzulesen. Die andere Möglichkeit wäre die Daten von Hand einzugeben.
	Bernhard kann jetzt auswählen, ob er ein neues Foto schiessen will oder ein bestehendes Foto aus dem Archiv auswählen will. Er entscheidet sich, ein neues Foto zu schiessen.

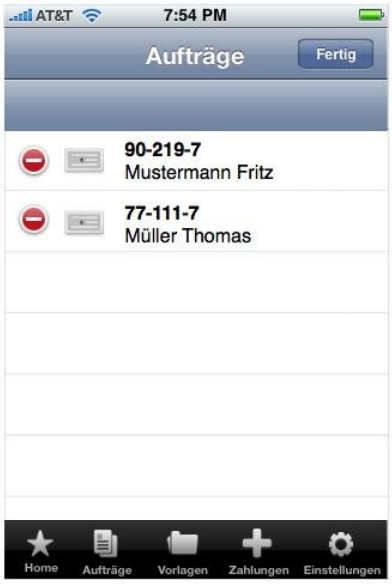
	Mithilfe seiner iPhone-Kamera schiesst er nun ein Foto der Codierzeile des Einzahlungsscheins. (Hinweis: Die Kamera verwendet dabei eine Maske, die genau der Grösse der Codierzeile entspricht.)
	Die Daten werden automatisch in die Zahlung eingefügt. Bernhard überprüft kurz die Richtigkeit der Angaben und führt darauf die Zahlung aus.

Nach der Ausführung der Zahlung erhält Bernhard eine kurze Zusammenfassung der getätigten Zahlung. Da es sich um eine einmalige Einzahlung handelt, speichert er diese nicht als Vorlage und klickt auf den „Fertig“-Button.

Szenario 4

Lothar Löscher hat vor zwei Tagen eine Zahlung ausgeführt, als Ausführungsdatum hat er das heutige Datum ausgewählt. Da er unterdessen den Betrag bar bezahlt hat, will er die pendente Zahlung löschen.

	Lothar startet die finApp und wählt auf dem Home-Screen den Menüpunkt „Aufträge“ aus.
	Er erhält eine Übersicht über alle pendenten Aufträge. Mit einem Klick auf „Verarbeitete“ könnte er sich alle bereits verarbeiteten Zahlungen anzeigen lassen. Lothar klickt nun auf den „Edit“-Button und gelangt so in den Bearbeitungs-Modus.

	Hier hat er einzig die Möglichkeit einen pendenten Auftrag zu löschen. Dazu klickt er auf das „Löschen“-Symbol vor dem Auftrag an Müller Thomas. Dadurch wird diese Zahlung aus der Pendenten-Liste entfernt.

7.4.1.2 Fragebogen zu den Paperprototypes und den Szenarien

Nachfolgend der Fragebogen, der an die Bankkunden der finnova AG verteilt und von Ihnen ausgefüllt worden ist.

Allgemein

Bedienung des User Interface (UI)

☒ UI ist intuitiv bedienbar

☐ UI ist übersichtlich. Jederzeit ist man sich bewusst, wo man sich befindet. Falls nein, wo nicht:

Die Begriffe sollten sich an das bestehende e-banking anlehnen (bspw. Aufträge).

☒ Jederzeit ist ersichtlich, wohin man weiternavigieren kann. Falls nein, wo nicht:

[Platzhalter]

☐ UI enthält Lücken. Falls ja, welche:

Die Position des + Buttons (Vorlagenordner hinzufügen) sollte überdacht werden – ist er am richtigen Ort? Andere Apps machen es anders.

Hauptnavigation (Tab-Bar)

Die Hauptnavigation sieht nur die Auswahl von ZV-Funktionen vor, und geht davon aus, dass ein Dashboard vorgeschalten wird, in dem verschiedene Module wie z.B. Konto/Depot, ZV oder ausgewählt werden können.

☐ Idee dieser strikten Trennung ok

☒ Nicht ok. Die Tab-Bar sollte um alle aufrufbaren Module erweitert werden (z.B. Tab „Mehr“)

Wir wissen nicht wo sich die Applikation weiterentwickelt. Die Navigation sollte offen gestaltet werden. Vermögensübersicht ist die meistgenutzte Funktion im e-banking und nicht der ZV – darum ist eine Trennung wichtig.

Über die Hauptnavigation erreichbare ZV-Bereiche

☒ Ok

☐ Zuviel

☒ Zuwenig

Falls zu viele oder zu wenige, was wäre eine alternative Tab-Bar:

Eventuell muss zu einem späteren Zeitpunkt auch der Ausland-ZV abgebildet werden.

Homescreen

Auf dem Homescreen werden dem Benutzer die sechs meistbenutzen Zahlungsvorlagen und Zahlungserfassungstypen in Form von Favoriten angezeigt. Durch das Erfassen einer neuen Zahlung oder dem Ausführen einer Zahlungsvorlage steigt die Gewichtung des Favoriten und wird in der Liste höher eingestuft.

☒ Ok

☐ Benutzer soll selber Favoriten setzten und Gewichten(sortieren) können

☒ Es sollen mehr als sechs Favoriten angezeigt werden können (Scrollen)

☐ Andere Idee für den Homescreen

Achtung – Vermögensinformation ist wichtig – wenn mit Homescreen der Startbildschirm des ZV-Teils gemeint ist: OK.

Einstellungen

☒ Belastungsanzeige nur unter Einstellungen auswählbar

☐ Belastungsanzeige muss auch pro Zahlung gewählt werden können

☒ Standard Konto soll in den Einstellungen wählbar sein

Weitere Ideen:

[Bemerkungen]

Fotografieren der Codierzeile

☒ Bereich der Codierzeile und Zoom der Kamera sollen von der finApp vorgegeben werden (wie im Paperprototype)

☐ Benutzer soll den Bereich der Codierzeile nach dem Fotografieren mittels einer Zuschneidefunktion selber auswählen können

☒ Bereits fotografierte Einzahlungsscheine sollen auch aus dem Standard-Fotoarchiv des Mobilgerätes ausgewählt werden können.

Bemerkungen

[Bemerkungen]

Szenario 1

☒ Szenario komplett

Bemerkungen zu Szenario 1

[Bemerkungen]

Szenario 2

☒ Szenario komplett

Bemerkungen zu Szenario 2

Oft muss beim erstellen einer Zahlung aus dem Vorlagenordner auch die Referenznummer angepasst werden. Ist man eventuell schneller mit einem erneuten einlesen per Foto?

Szenario 3

☒ Szenario komplett

Bemerkungen zu Szenario 3

[Bemerkungen]

Szenario 4

☒ Szenario komplett

Bemerkungen zu Szenario 4

[Bemerkungen]

8 Studien

8.1 Securityanalyse

8.1.1 Ausgangslage

Dieses Dokument dient dazu, eine Übersicht der bestehenden Security-Verfahren zu haben, welche auch auf einem Mobilgerät eingesetzt werden können. Ziel ist es, der Bank aufzuzeigen, dass ein sicheres Mobile-Banking möglich ist und die entsprechenden Verfahren auf dem Markt verfügbar sind. Der finApp-Server muss ein Security-Interface bereitstellen, mittels dem die Verfahren gemäss dieser Analyse integriert werden können.

8.1.2 Gefahren beim Mobile-Banking

Die nachfolgenden Gefahren müssen beim Mobile-Banking in Betracht gezogen werden:

- Die Informationen, welche vom Mobilgerät an den Server gesendet worden sind, entsprechen nicht jenen, die der Benutzer im Gerät eingegeben hat
- Die Informationen, welche dem Benutzer im Mobilgerät angezeigt werden, entsprechen nicht jenen, die der Server gesendet hat

Ermöglicht werden diese Gefahren durch die Infektion des Mobilgeräts mittels Viren, Malware und Trojaner.

8.1.3 Basic Authentication

Die grundlegende Authentifizierung beinhaltet, dass jedem Benutzer ein eindeutiger Benutzername und ein Passwort zugewiesen wird. Für eine erfolgreiche Autorisierung ist es notwendig, dass sich der Benutzer mit diesen zwei Credentials dem System gegenüber authentisiert.

Die in diesem Dokument beschriebenen Security-Verfahren bauen auf dieser Basisauthentifizierung auf und beschreiben, wie weitere Sicherheitsmerkmale in ein System intgeriert werden können.

8.1.4 Datenintegrität

Zusätzlich zur Authentifizierung wird immer häufiger gefordert, dass die Integrität der Daten festgestellt werden kann. Um dies zu gewährleisten, muss ein bestehendes Securitykonzept um eine Transaktionssignierung erweitert werden, welche dem Benutzer die Möglichkeit bietet, die Integrität der übertragenen Daten zu bestätigen.

Die Transaktionssignierung beinhaltet, dass die Transaktionsdaten, welche der Server empfängt, dem Benutzer auf einem zweiten Kanal nochmals angezeigt werden können. Der Benutzer überprüft die Daten und signiert diese, falls sie seiner Transaktion entsprechen. Erst wenn der Server diese Signatur erhält, wird die Transaktion dem Finnova Core übergeben.

Die Vorteile liegen dabei auf der Hand. Durch den Einsatz eines zweiten Kanals, kann eine Man-In-The-Middle-Attacke, welche die Transaktionsdaten manipuliert hat, zwar nicht umgangen werden, jedoch kann der Benutzer so erkennen, dass die Integrität der Daten nicht mehr gewährleistet ist. Anzumerken ist, dass nicht ausgeschlossen werden kann, dass auch der zweite Kanal von einem Hacker infiltriert worden ist.

8.1.5 Security-Verfahren

Nachfolgend eine Beschreibung der verschiedenen Verfahren, welche auch im Internetbanking der Finnova eingesetzt werden. Für jedes Verfahren werden folgende Punkte aufgeführt:

- Beschreibung
- Authentifizierung - wie kommt es zu einer erfolgreichen Autorisierung des Benutzers
- Kann eine Transaktionssignierung durchgeführt werden, und falls ja wie
- Vor- und Nachteile

8.1.5.1 Streichliste

Nr / Code	Nr / Code	Nr / Code	Nr / Code	Nr / Code
00: XVD4	**10:** XL49	**20:** ZNLL	**30:** LUPG	**40:** 854Y
01: AA8K	**11:** UGZL	**21:** BM8B	**31:** PV4S	**41:** 66GR
02: U35U	**12:** ELFQ	**22:** 4THH	**32:** QSD8	**42:** 75HD
03: PGQX	**13:** 63U3	**23:** 6ZZB	**33:** 2J3K	**43:** KHX4
04: GENJ	**14:** DHUP	**24:** CHG5	**34:** 5AF7	**44:** R8EP
05: 24P3	**15:** K2QR	**25:** 7VFU	**35:** D75S	**45:** 3A8J
06: EQG4	**16:** P4NJ	**26:** SR27	**36:** Q6T9	**46:** DEG9
07: NL3D	**17:** KY64	**27:** 6YB6	**37:** CEJQ	**47:** D8J2
08: 6V86	**18:** PJJK	**28:** Y8YN	**38:** 3VAK	**48:** 78V7
09: 6EJJ	**19:** 9FBJ	**29:** CLQG	**39:** J7DB	**49:** Z98N

Abbildung 9 - Streichliste

Beschreibung

Die Streichliste ist im Moment wohl noch das bekannteste Security-Verfahren, da sie auch heute noch in vielen Internetbanking Plattformen eingesetzt wird. Der Benutzer erhält diese Streichliste nicht in elektronischer sondern in gedruckter Form. Darauf enthalten ist ein Raster von zufällig generierten Codes, die einer bestimmten Position zugeordnet sind. [06]

Authentifizierung

Als zusätzliches Sicherheitsmerkmal wird der Benutzer aufgefordert, den Code auf einer bestimmte Position der Streichliste dem Server zu übermitteln. Der Server überprüft den erhaltenen Code und authentifiziert den Benutzer, falls die geforderte Position geliefert worden ist.

Transaktionssignierung

Eine Transaktionssignierung ist mit der Streichliste nicht möglich. Denkbar ist, dass der Benutzer vor der Ausführung einer Zahlung nochmals aufgefordert wird, einen bestimmten Code zu übermitteln. Dies gewährleistet jedoch nicht die Integrität der Daten.

Vorteile

- günstig in der Anschaffung
- unabhängig vom Mobilgerät einsetzbar

Nachteile

- Datenintegrität kann nicht gewährleistet werden

8.1.5.2 Smartcard

Abbildung 10 - Smartcard

Beschreibung

Die Smartcard im Kreditkartenformat haben einen Chip, auf dem Zertifikate und kryptographische Schlüssel abgespeichert sind. Der Chip bietet einem Kartenleser ein Interface, über welches Befehle (Schreiben, Lesen, Signieren, Ver- und Entschlüsseln) vom Leser auf dem Chip der Smartcard ausgeführt werden können. Die Smartcard ermöglicht somit dem Benutzer, auf dem Kartenleser eingegebene Daten zu verschlüsseln. [07]
Der Server kennt diesen Schlüssel und ist in der Lage, diese Daten wieder zu entschlüsseln.

Authentifizierung

Der Server übermittelt dem Benutzer einen Code, der mittels der Smartcard verschlüsselt zurückgeschickt werden muss.

Transaktionssignierung

Mittels der Smartcard ist es möglich, eine Transaktionssignierung durchzuführen. Der Benutzer kann beispielsweise dazu aufgefordert werden, den Betrag und bestimmte Stellen des Begünstigtenkontos mit dem Key zu verschlüsseln, worauf der Server diese Daten mit den gespeicherten Daten überprüfen kann.

Vorteile

- Verschlüsselung des Codes
- Transaktionssignierung wird unterstützt
- unabhängig vom Mobilgerät einsetzbar

Nachteile

- Smartcard und Reader müssen dem Benutzer zur Verfügung gestellt werden
- Benutzer muss die Transaktionsdaten von Hand im Reader erfassen

8.1.5.3 Public Key Infrastruktur

Beschreibung

Sensible Daten wie öffentliche Schlüssel und Zertifikate können direkt auf einem USB-Stick oder einem anderen Hardware-Device, mit Display und ohne USB-Interface, gespeichert werden.

Authentifizierung

Der Server fordert vom Benutzer ein zusätzliches Sicherheitsmerkmal, welches vor der Eingabe in das System mit dem Public Key verschlüsselt werden muss.
Wird ein Hardware-Device ohne USB-Anschluss eingesetzt, muss der für ein bestimmtes Zeitintervall gültige Code abgetippt werden.

Transaktionssignierung

Eine Transaktionssignierung ist mittels dem PKI Verfahren ist nicht möglich.Wie bei der Streichliste könnte ein weiterer Code angefordert werden, was jedoch nicht die Integrität der Daten gewährleistet.

Vorteile

- Verschlüsselung des Codes mit dem Public Key
- unabhängig vom Mobilgerät einsetzbar

Nachteile

- USB-Sticks müssen dem Benutzer zur Verfügung gestellt werden
- Anschluss des USB-Sticks an das Mobilgerät

8.1.5.4 Mobile Transaction Number (mTan)

Beschreibung

Ein zusätzliches Sicherheitselement wird per SMS an den Benutzer gesendet. [08]

Authentifizierung

Der Server sendet eine Transaktionsnummer dem Benutzer als SMS zu, welcher seinerseits diese Nummer als weiteres Sicherheitselement dem Server überträgt.

Transaktionssignierung

Mittels mTan ist es möglich, eine Transaktionssignierung durchzuführen. Der Benutzer erhält vom Server eine SMS in der die Transaktionsdaten und eine Transaktionsnummer enthalten sind. Entsprechen die Daten im SMS den eingegebenen Daten, wird die Transaktions mittels der Transaktionsnummer signiert.

Vorteile

- kein Rollout eines zusätzlichen Gerätes notwendig
- Transaktionssignierung wird untertstützt

Nachteile

- Um dieses Verfahren sicher für mobile Banking zu benutzen, müsste der Benutzer die mTan auf einem zweiten Gerät erhalten.
- hohe wiederkehrende Kosten bei sehr aktiven Benutzern (SMS-Versand)
- Phishing Attacken über SMS werden ermöglicht [09]

8.1.5.5 zTic

Abbildung 11 - zTic [10]

Beschreibung

zTic ist ein USB-Stick, Hergestellt von der Firma IBM, auf welchem ein TSL- oder SSL-Zertifikat gespeichert ist. Im Unterschied zu einem PKI-Verfahren, wird eine End-Zu-End-Verbindung zwischen zTic und Server hergestellt, die mittels dem TSL- oder SSL-Protokoll gesichert ist. [10]

Authentifizierung

Erhält der Server einen Authentisierungsrequest vom Benutzer, wird dieser gestoppt und die Authentifizierung wird direkt über die gesicherte Verbindung zwischen zTic und dem Server abgewickelt. Durch dieses Verfahren wird ein möglicherweise kompromittierter Computer umgangen.
Der User kann über Steuerungstasten auf dem zTic eine Authentifizierungsanfrage vom Server bestätigen und sich so authentisieren. Hat das System den User authentifiziert, wird der ursprüngliche Browser-Request weitergeleitet.

Transaktionssignierung

Ähnlich mTan kann auch mit dem zTic eine Transaktionssignierung durchgeführt werden. Beim zTic wird jedoch auf den zweiten sicheren Kanal gesetzt und die Transaktionsdaten sowie die Signatur des Benutzers werden ausschliesslich über diesen Kanal ausgetauscht. Wie bei der Authentifizierung muss der Server auf den Callback des zTic warten, bevor die Transaktion ausgeführt werden kann.

Vorteile

- wenige Interaktionen des Benutzers ("OK" oder "NOK" drücken, optionale Eingabe des PIN Codes)

Nachteile

- Anschluss des USB-Sticks an das Mobilgerät.
- Pilotierung eines Mobilgerät tauglichen zTic ist erst im dritten Quartal 2011 vorgesehen

8.1.6 Schlussfolgerung

Der Securitybereich für Mobile-Banking ist ein Markt, der ständigem Wandel unterzogen ist. Es existieren bereits einige Verfahren, die auch im Bereich Mobile-Banking eingesetzt werden können und es werden in Zukunft sicherlich noch neue Verfahren entwickelt und veröffentlicht werden. Ziel der finApp soll es somit sein, ein Security-Interface anzubieten, über welches der Server verschiedene Security-Verfahren ansprechen kann.

Die Abbildung 12 - Anbindung des external Security Interface dient der Illustration, wie ein ausgewähltes Security-Verfahren angebunden wird.

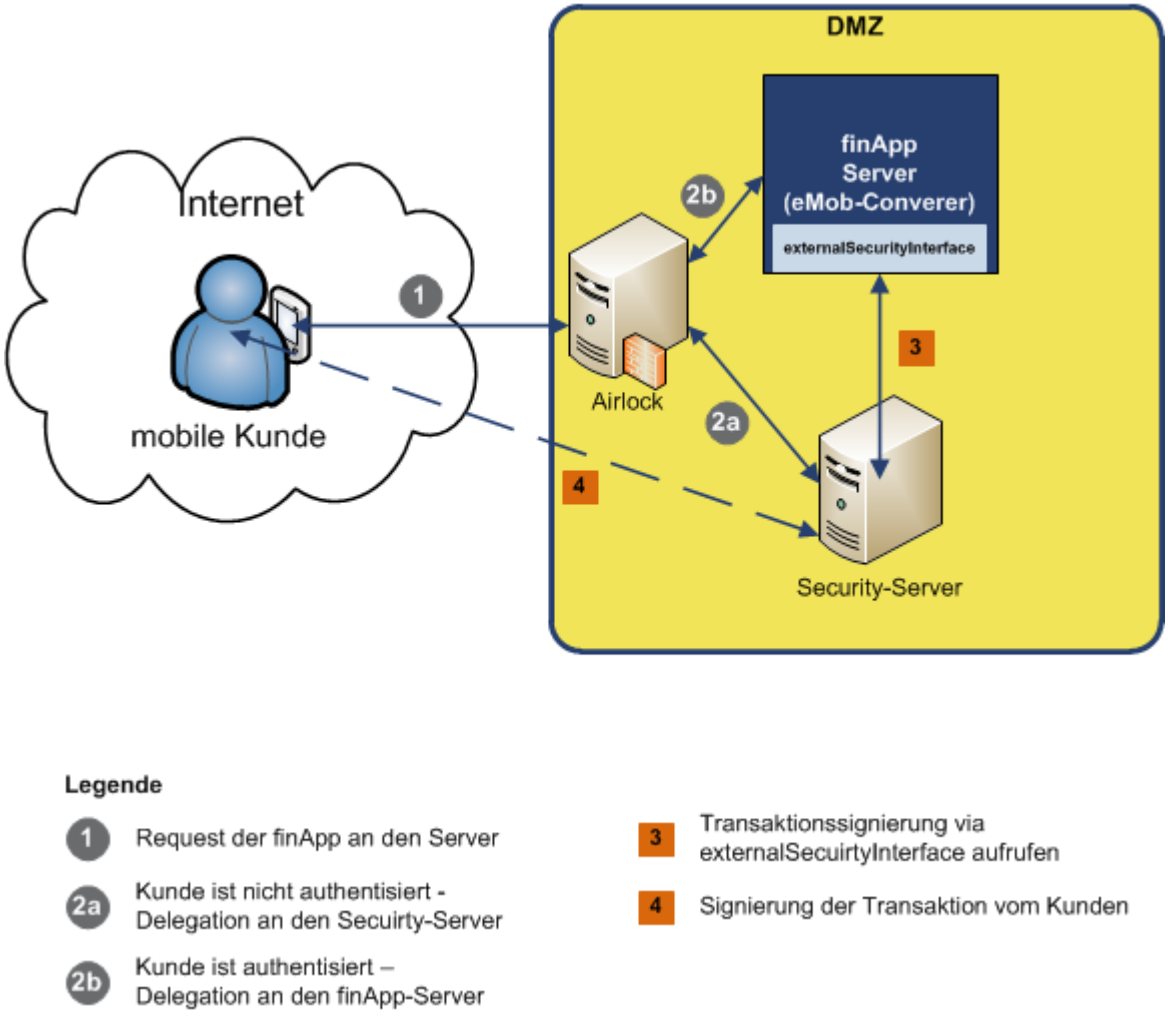


Abbildung 12 - Anbindung des external Security Interface

Die Business-Logik der Authentisierung und Transaktionssignierung sind auf dem Security-Server hinterlegt. Der Zugriff vom finApp- auf den Security-Server wird mittels einem Plugin, welches die Hersteller des Security-Verfahrens implementieren müssen und das der finApp-Server beim Start ladet, gewährleistet. Das Interface des Plugins (externalSecurityInterface) wird dabei von der Finnova vorgegeben.

8.2 OCR

8.2.1 Ausgangslage

Dieses Dokument beinhaltet eine Übersicht über verschiedene OCR-Software. Um die Codierzeile eines Einzahlungsschein erkennen zu können, haben wir uns sowohl mit kommerziellen sowie auch Open Source Lösungen auseinandergesetzt.

8.2.2 Einführung

Optical Character Recognition beschreibt die automatische Erkennung von Textinformationen aus Bilder. Die Technik beruht auf einem Mustervergleich von bekannten Buchstaben, Zahlen und Sonderzeichen.

8.2.3 ABBYY Mobile OCR Engine

Abbildung 13 - Logo ABBYY [11]

Mobile OCR Engine ist ein SDK, welches vom Softwarehersteller ABBYY [11] im Jahr 2009 auf den Markt gebracht wurde. ABBYY Mobile OCR erkennt Texte in 62 Sprachen und wurde mehrfach ausgezeichnet.

Vorteile:

- Unterstützung für das iPhone
- Text im Spalten-Layout werden korrekt ausgelesen
- Automatische Erkennung der Dokument/Foto Ausrichtung
- Automatische Korrektur (z.B. bei Zeilenumbrüchen „Mon- day“ → „Monday“)

Nachteil:

- Hohe Kosten, da kommerziell

8.2.4 GOCR

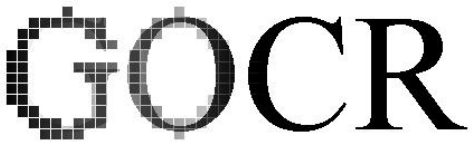


Abbildung 14 - GOCR Logo [12]

GOCR [12] (oder auch JOCR) ist ein freies OCR-Programm, welches ursprünglich von Jörg Schulenburg stammt. Es kann dazu benutzt werden gescannte Bild-Dateien in Text-Dateien umzuwandeln.

GOCR gibt an, dass es mit einer einzeiligen nicht-serifen Schrift mit einer Grösse von 20-60 Pixel umgehen kann. Das Verfahren bekundet bei folgenden Gegebenheiten Mühe:

- serife Schriften
- überlappende Buchstaben
- handgeschriebener Text
- verschiedenartige Schriften
- verrauschte Bilder
- grosse Winkel in der Schräge
- und Text in einem anderen Format als dem lateinischen Alphabet

8.2.4.1 Unterstützte Formate

Die folgenden Bild-Formate werden von GOCR unterstützt:

- pnm, pbm, pgm, ppm
- einige pcx
- tga

Die folgenden Formate werden automatisch konvertiert (mittels netbpm-progs, gzip und bzip2):

- pnm.gz, pnm.bz2
- png, jpeg
- tiff, gif, bmp

Ausserdem ist GOCR in der Lage Barcodes zu übersetzen

8.2.5 Tesseract

Tesseract [13] ist ein in C++ geschriebenes frei verfügbares Texterkennungsprogamm und wurde zwischen 1985 - 1995 von Hewlett-Packard entwickelt. Als Hewlett-Packard aus dem OCR-Markt ausstieg lag das Projekt über 10 Jahre lang brach. Erst seit dem Jahr 2006 wird Tesseract von Google weiterentwickelt.

Vorteile:

- Sehr gute Erkennungsrate
- Wird von Google weiterentwickelt

Nachteile:

- Keine Layout-Analyse

8.2.6 Schlussfolgerung

Wir suchten eine möglichst einfache Art die Codierzeile auf einem Foto zu erkennen. Features wie Autokorrektur von Wörtern, Spaltenunterstützung, etc. war für uns unwichtig. Wir legten Wert auf eine gute Erkennungsrate sowie auf eine möglichst einfache Integration der Library in unsere iPhone-Applikation.

Nachdem wir uns mehrere OCR-Librarys/Engines angeschaut und verglichen haben, entschieden wir uns für Tesseract. Da Tesseract frei zur Verfügung steht, eine sehr gute Erkennungsrate aufweist und von Google weiterentwickelt wird, gibt es auch eine vergleichsweise grosse Community im Internet. Dadurch war es für uns einfacher die OCR-Software in unserer Applikation zu verwenden.

8.3 XML-Parser

8.3.1 Einleitung

Applikationen für das iPhone sind zum grössten Teil in der Programmiersprache Objective-C geschrieben, so auch die finApp. Will der Benutzer per finApp eine Zahlung beauftragen, so gibt er die Zahlungsdetails in die Applikation ein. Die Details werden darauf in einer entsprechenden Objective-C-Klasse gespeichert und müssen nun über eine gesicherte Verbindung an den Server (eMob-Converter) geschickt werden. Der Server arbeitet jeweils mit XML-Requests und –Responses, diese sind durch die finnova AG vorgegeben. Aus diesem Grund bedarf es sowohl eines XML-Parsers, der die XML-Responses des Servers in eine Objective-C-Klasse schreibt, als auch eines XML-Writers, der eine Objective-C-Klasse (mit den Zahlungsdetails) in einen XML-Request schreibt.

Die finApp und der eMob-Converter stellen die folgenden Merkmale an den XML-Parser, resp. –Writer:

- Implementation in Objective-C:
  Da die Applikation bereits in dieser Programmiersprache geschrieben ist, stellt ein Parser/Writer in einer anderen Sprache (z.B. C) ein unnötiger Mehraufwand dar.
- Schnelles Parsen:
  Je länger der Benutzer auf das Resultat warten muss, umso mühsamer wird die Zahlungserfassung mittels iPhone. Es handelt sich jedoch bei der finApp um kleine XML-Files, sodass dieses Merkmal nur bedingt berücksichtigt wird.
- Einfache Umsetzung von Read und Write-Methoden:
  Es existieren drei verschiedene XML-Requests mit einer einfach Struktur. Ein Beispiel einer solchen Struktur befindet sich im Kapitel 9.6 (XML-Struktur).
  Daher braucht es keine komplexe Library, welche fähig ist, alle möglichen Strukturen zu lesen/schreiben, sondern lediglich eine einfache write- resp. read-Methode.
- XPath-Unterstützung:
  XPath ist eine Abfragesprache, um Teile eines XML-Dokuments zu adressieren. Diese erleichtert das Lesen von XML-Dokumenten enorm.

8.3.2 XML-Parser

Im Folgenden werden verschiedene XML-Parser kurz vorgestellt. Alle Parser wurden in der Anfangsphase der Studienarbeit getestet und miteinander verglichen.

8.3.2.1 NSXMLParser

Der NSXML-Parser [14] ist der Standard XML-Parser in der Mac OS X / iPhone-Programmierung. Der SAX-Parser ist bereits in der iPhone-SDK integriert. NSXML-Parsing funktionert nach dem event-driven Prinzip, d.h. die Parser-Klasse übergibt einem Delegate die Daten, welche sie im XML-Dokument gefunden hat und es liegt dann in der Hand des Delegates etwas mit diesen Daten anzufangen.

Vorteile:

- bereits in der iPhone SDK enthalten
- verbraucht wenig Speicher

Nachteile:

- im Vergleich langsam
- komplizierte Anwendung

8.3.2.2 Libxml2

Der Libxml2-Parser [15] ist ebenfalls bereits für das iPhone verfügbar. Der Libxml2 ist eigentlich ein C-Parser und wurde ursprünglich für das Gnome-Project entwickelt, kann aber auch ausserhalb der Gnome-Plattform genutzt werden. Den Parser gibt es sowohl in der SAX- als auch in der DOM-Variante.

Vorteile:

- bereits in der iPhone SDK enthalten
- schnelles Parsen

Nachteile:

- in C implementiert
- komplizierte Anwendung

8.3.2.3 TBXML

TBXML [16] ist ein leichtgewichtiger Parser für den Gebrauch von iPhones. Der DOM-Parser hat das Ziel, der schnellste XML-Parser mit dem geringsten Speicherverbrauch zu sein, dies jedoch auf Kosten von nicht implementierten Funktionen, wie z.B. dem Bearbeiten eines XML-Dokuments oder der Unterstützung von XPath.

Vorteile:

- in Objective-C implementiert
- mit Abstand schnellster Parser

Nachteile:

- Unterstützt zu wenig Methoden (XPath)

8.3.2.4 TouchXML

Der TouchXML-Parser [17] hat das Ziel eine leichtgewichtige NSXML style API zu kreiern. Der DOM-Parser bietet nur eine Read-Funktion für XML-Dokumente an und unterstützt XPath.

Vorteile:

- Schneller Parser
- Einfache Anwendung

Nachteile:

- Keine Write-Methode

8.3.2.5 KissXML

Der KissXML-Parser [18] ist ebenfalls ein Parser in NSXML Style. Der DOM-Parser basiert auf dem oben vorgestellten TouchXML-Parser, zusätzlich bietet er das Editieren von XML-Dokumenten an.

Vorteile:

- Einfache Anwendung
- In Objective-C implementiert

Nachteile:

- XML-Writing sehr umständlich

8.3.2.6 TinyXML

Der TinyXML-Parser [19] ist ein kleiner C++-Parser, welcher aber auch in andere Programme integriert werden kann. Der DOM-Parser bietet keine direkte Unterstützung für Xpath an (über eine weitere Library kann jedoch der sog. „TinyXPath“ eingebunden werden).

Vorteile:

- Schneller Parser

Nachteile:

- In C++ implementiert
- Keine direkte XPath Unterstützung

8.3.2.7 GDataXML

Der GDataXML-Parser [20] wird als Teil der Objective-C client library von Google angeboten. Der DOM-Parser enthält nur ein M-File und eine Header-Datei.

Vorteile:

- Schneller Parser
- In Objective-C implementiert

Nachteile:

- Probleme beim Lesen/schreiben von Attributen

- http://developer.apple.com/library/ios/#samplecode/XMLPerformance/Introduction/Intro.html

8.3.3 Vergleich

Die verschiedenen XML-Parser wurden unter anderem mit einer kleinen XML-Performance-Applikation getestet. Diese von Apple bereitgestellte Applikation [21] lässt die verschiedenen Parser ein ca. 900KB XML Dokument (mit den Top 300 iTunes Songs) parsen. Die folgenden Bilder stellen kurz die Testergebnisse dar.

Abbildung 15 - XML-Performance

Abbildung 16 - NSXML und libxml2

Abbildung 17 - TBMXL und TouchXML

Abbildung 18 - KissXML und TinyXML

Abbildung 19 - GDataXML

Es ist ersichtlich, dass der NSXML- und der TinyXML-Parser mit Abstand am meisten Zeit für das Parsen aufwenden mussten. Der libxml2- und TBXML-Parser waren die beiden schnellsten Parser im Test. Die anderen drei Parser waren in etwa gleich schnell und befinden sich damit im Mittelfeld.

8.3.4 Schlussfolgerung

Keiner der Parser kann uns vollends überzeugen. Der integrierte NSXMLParser ist sehr mächtig, aber für unseren Verwendungszweck zu aufgebläht und daher zu kompliziert für eine einfache XML-Struktur. Der Libxml2-Parser ist mit Abstand der Schnellste der getesteten Parser, jedoch ist er in C implementiert, sodass ein unnötiger Mehraufwand daraus resultiert. Der TBXML-Parser hat eigentlich optimale Voraussetzungen für unsere iPhone-Applikation, leider unterstützt er kein XPath. Der XML-Parser von Google (GDataXML) hat Probleme beim Lesen und Schreiben von XML-Attributen. Da unsere XML-Struktur viele Attribute enthält, ist auch dies nicht der optimale Parser für uns. Der TinyXML-Parser zeigt sich als zu winzig für unsere Anwendungszwecke (keine XPath-Unterstützung), zudem ist er nicht in Objective-C implementiert.
Der TouchXML-Parser gefällt uns am besten zum Lesen von XML-Dokumenten, er ist sehr schnell und auch einfach zu implementieren. Er unterstützt alle Methoden zur Umwandlung eines XML-Attributes in ein Objective-C Property. Leider ist keine Write-Methode implementiert. Das bringt uns zum letzten Parser, dem KissXML-Parser. Die Read-Methoden sind ebenso, wie beim TouchXML optimal für unsere Anwendungszwecke, die zusätzlichen Write-Methoden funktionieren jedoch nur über Umwege und sind sehr umständlich zu implementieren.

Schlussendlich hat uns keiner der getesteten Parser vollends überzeugt, sodass wir uns für eine Dual-Lösung entschieden haben:
Für das Parsen kommt der TouchXML-Parser zum Einsatz, er bietet ein paar wenige Klassen für das Lesen von XML-Dokumenten an. Für das Schreiben verwenden wir nun keinen Parser, sondern einen simplen XML Stream Writer.
Dazu eignet sich der XSWI [22] am besten. XSWI ist ein simpler XML stream writer, welcher in Objective-C geschrieben ist. Das Umwandeln von Objective-C Properties in XML-Attribute funktioniert sehr schnell und ist sehr simple zu implementieren. Ausserdem besteht das XSWI-Projekt nur aus einer .m-Datei und einem zugehörigen Header-File.

Mithilfe dieser beiden Libraries, ist es uns nun möglich, Einzahlungen schnell und ohne grossen Aufwand an den Server zu schicken und die Antworten des Servers auf dem iPhone zu verwenden.

9 Software Architektur Design

9.1 Einführung

9.1.1 Zweck

Dieses Dokument beschreibt die Software Architektur des Grundsystems

9.1.2 Gültigkeitsbereich

Das Dokument ist für die komplette Studienarbeitdauer gültig.

9.1.3 Übersicht

Zu Beginn des Dokumentes wird die Architekturübersicht beschrieben. Danach werden die einzelnen Packages genauer betrachtet.

Der Server ist in einem separaten Kapitel 9.7 Serverarchitektur – Spezifikation des eMob-Converter beschrieben.

9.2 Architekturübersicht

Das folgende Diagramm bieten einen Überblick der verwendeten Komponenten und Interaktionen. Nicht enthalten in diesem Diagramm ist die Authentisierungskomponente, da diese nicht von der Finnova entwickelt wird.Genauere Informationen, wie die Authentisierung abläuft, sind in der Beschreibung der Komponente Airlock zu finden.

In den nachfolgenden Abschnitten werden die einzelnen Komponenten und die Schnittstellen genauer beschrieben.

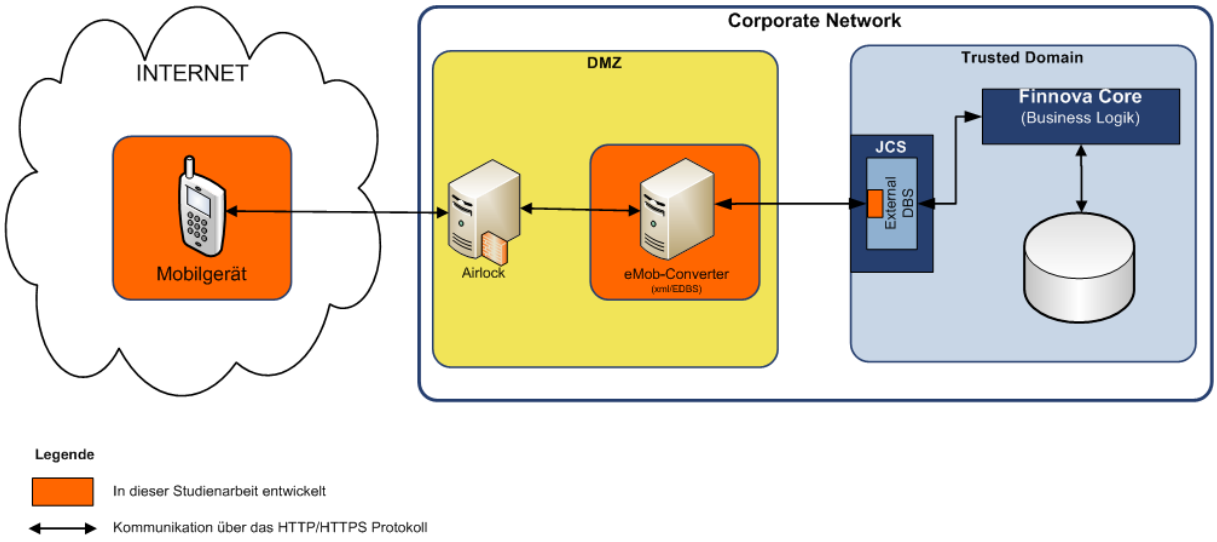


Abbildung 20 - Überblick der verwendeten Komponenten

9.2.1 Komponenten

9.2.1.1 Mobilgerät

Mithilfe der finApp, welche auf einem Mobilgerät (iPhone oder iPod Touch) läuft, kann der Bankkunde Zahlungen erfassen und diese ausführen. Das Mobilgerät sendet mithilfe des HTTPS Protokolls Requests an Airlock.

Die finApp wird im Kapitel 9.4 (Logische Architektur) noch genauer beschrieben.

9.2.1.2 Airlock

Airlock ist eine Web-Application-Firewall (WAF) die von der Firma Ergon [23]entwickelt und von den Kunden der Finnova eingesetzt wird.
Zusammen mit dem Authentisierungsserver, welche in der Übersicht nicht enthalten ist, kontrolliert und authentifiziert der Airlock Zugriffe auf die Komponenten und Ressourcen innerhalb der DMZ.
Eine erfolgreiche Authentisierung eines Benutzers auf dem Authentisierungsserver bedeutet, dass dieser ein auf dem Airlock parametrierbare Rolle mittels dem Airlock-Control-Cookie setzt. Nur wenn dieses Cookie gesetzt ist, lässt Airlock Zugriffe auf die anderen Systemkomponenten, welche sich in der DMZ befinden, zu.

9.2.1.3 eMob-Converter (eMobC)

Der eMobC ist dafür verantwortlich, dass die vom Mobilgerät eingehenden Requests den EDBS-Tasks übergeben werden und die Daten, welche die EDBS-Tasks liefern, dem Mobilgerät zurückgeliefert werden. Auf dem eMob-Converter befindet sich keine Business Logik, sonder nur das Mapping der Services zwischen Mobilgerät und Finnova Core.

9.2.1.4 Job Control System (JCS)

Mittels dem Job Control System können einzelne Jobs definiert und gesteuert werden. Das external DBS läuft als ein solcher Job innerhalb der Trusted Domain.

9.2.1.5 External DBS (EDBS)

External DBS ist der Service, mittels dem die Finnova ihr DBS anderen Applikationen zur Verfügung stellen.
Das EDBS ermöglicht mittels einer Task Definition das Datenmodell zu verstecken. Eine Task Definition bedeutet, dass beschrieben werden kann, auf welches PL/SQL Package (bzw. welche Prozedur/Funktion) des Finnova Core Systems zugegriffen werden soll und welche Input- und Output-Parameter von diesem Aufruf erwartet werden.

Das EDBS stellt in Form einer Finnova-Library (edbs.jar) Java-Stubs zur Verfügung, welche von Applikationen verwendet werden können, um auf die Finnova Databank via Tasks zuzugreifen.

9.2.1.6 Finnova Core

Der Finnova Core beinhaltet die gesamte Business Logik. Sämtliche Datenbankzugriffe werden in PL/SQL Packages gschrieben und mittels den EDBS-Tasks den Umsystemen verfügbar gemacht.

9.2.2 Schnittstellen

9.2.2.1 Übertragungsprotokoll

Zwischen allen beschriebenen Komponenten wird das sichere Hypertext-Übertragungsprotokoll HTTPS für die Kommunikation verwendet.

9.2.2.2 Mobilgerät-eMobC

Die Schnittstelle zwischen dem Mobilgerät und dem eMobC wird mittels XML Schemas (XSD) beschrieben. Sowohl das Mobilgerät als auch der eMobC stellen einen Mechanismus zur Verfügung, welcher die Services gegen die Schemas validiert und das Marshalling/Unmarschalling der HTTP-Requests und -Respones vornimmt. Die Nutzdaten befinden sich dabei jeweils im Body des HTTP-Requests beziehungsweise der HTTP-Response. Die Beschreibung der Schnittstelle befindet sich im Kapitel 9.7.6 (Spezifikation der Schnittstelle "Mobilgerät-eMobC").

9.2.2.3 eMobC-Finnova Core

Die Schnittstelle zwische dem eMobC und dem Finnova Core wird mittels den EDBS-Tasks beschrieben. Die Beschreibung der Schnittstelle befindet sich im Kapitel 9.7.3.5 (EDBS).

9.3 iOS & Cocoa Framework

9.3.1 Schichtenmodell

Die Schichten des iPhone OS können in zwei Kategorien aufgeteilt werden. Die öffentlichen Schichten, auf die der Entwickler zugreifen kann sowie die privaten, welche keinen Zugriff erlauben.

Die folgende Grafik zeigt, wie unsere Applikation die verschiedenen Schichten des iPhone OS verwendet.

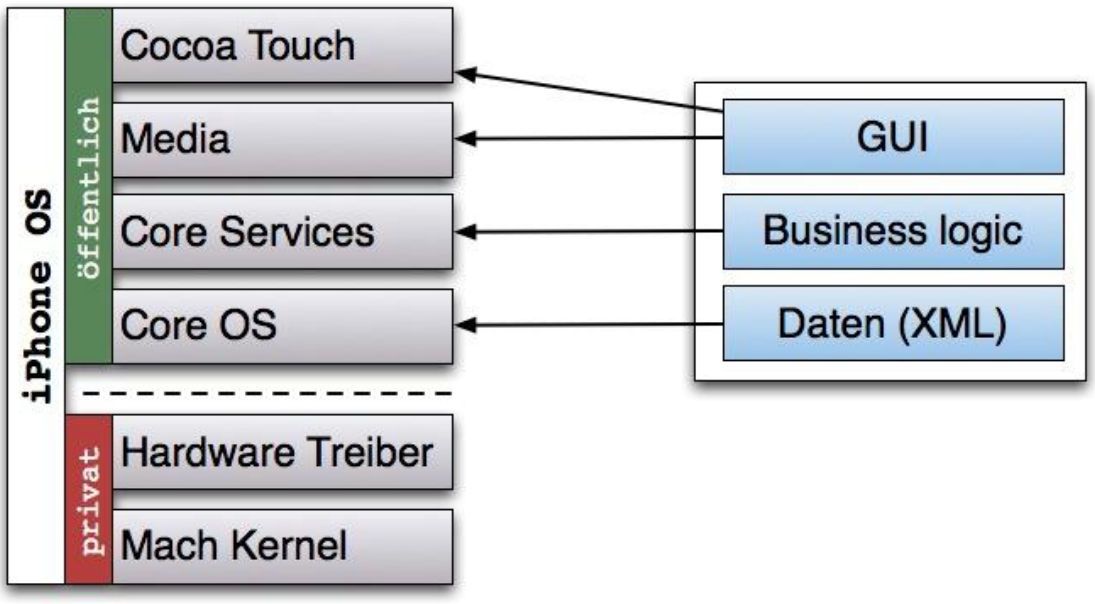


Abbildung 21 - Schichtem im iPhone OS

9.3.1.1 Cocoa Touch

Cocoa Touch beinhaltet das Basisframework Foundation (z.B. für Datentypen) sowie das Framework UIKit, welches für die grafische Oberfläche zuständig ist. Zusätzlich wird das Eventhandling sämtlicher UI-Komponenten übernommen.

9.3.1.2 Media

Die Media-Schicht stellt Dienste für Grafik, Audio und Video zur Verfügung.

9.3.1.3 Core Services

Die Core-Services-Schicht bietet folgende Dienste an:

- Adress Book
- Core Data
- Core Foundation
- Core Location
- Foundation Framework
- Mobilder Core Services
- In App Purchase
- SQLite
- Unterstützung für die Arbeit mit XML

9.3.1.4 Core OS

Aus der Core-OS Schicht benutzen wir die Basisdienste für die Netzwerkkommunikation.

iPhone OS [24]

9.3.2 Model-View-Controller

Das Cocoa Framework beinhaltet das MVC-Pattern. Die Abbildung 22 - MVC zeigt die Kommunikation zwischen UIView, UIViewController und den Model.

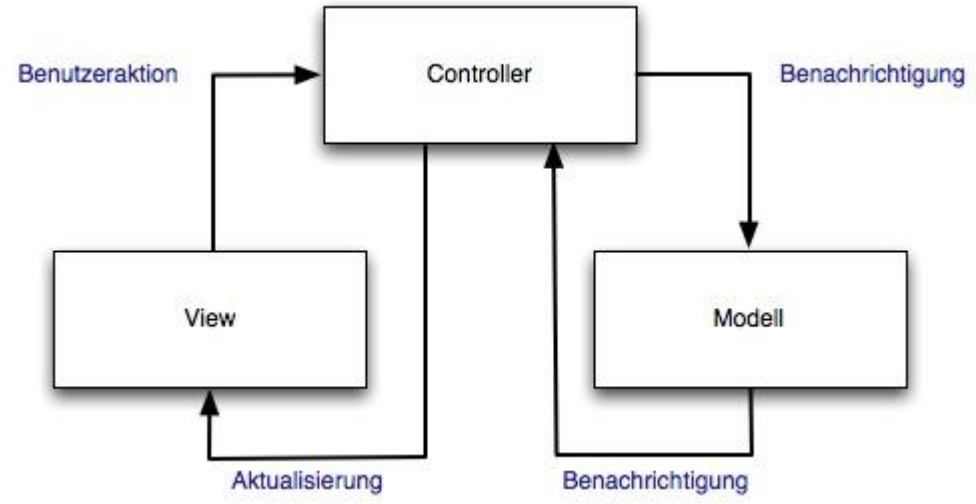


Abbildung 22 - MVC

9.4 Logische Architektur

Die finApp wird clientseitig in drei Schichten aufgeteilt, User Interface, Problem Domain und Data Converter.
In der unteren Abbildung sind jeweils die Abhängigkeiten zwischen den Packages angegeben, die Abhängigkeiten zwischen den Subpackages wurden aus Gründen der Übersichtlichkeit weggelassen.

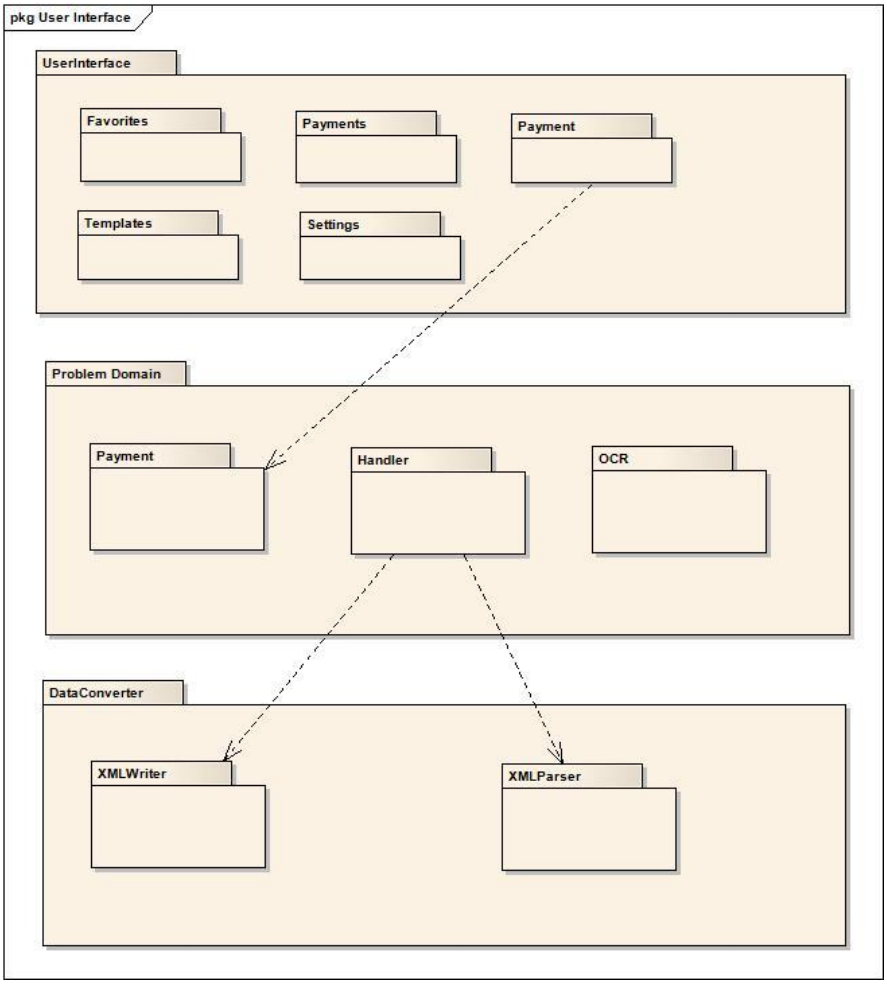


Abbildung 23 - Logische Architektur der finApp

9.4.1 User Interface

Im User Interface befinden sich die verschiedenen View-Controller der iPhone-App.

9.4.1.1 Subpackages

Die Subpackages entsprechen den fünf Tabbar-Menüpunkten der finApp. Jedes Subpackage enthält den für den jeweiligen Menüpunkt benötigten NavigationController sowie ein oder mehrere ViewController.

9.4.2 Problem Domain

In der Problem Domain werden die Komponenten des Domainmodells und somit die Logik der Problem Domain modelliert.

9.4.2.1 Payment

Im Sub-Package Payment sind die drei verschiedenen Paymenttypen Orange-, OCR- und SignPayment enthalten.
Diese Klassen bestehen jeweils aus mehreren Properties, wie z.B. Konto, Betrag, Währung.

9.4.2.2 Handler

Im Sub-Package Handler befinden sich die drei Handler für die unterschiedlichen Zahlungstypen.
Ausserdem beinhalte das Handler-Package auch den XMLServiceHandler, welche die Zahlungen an den eMob-Converter (Server) schickt und auch die darauf folgende Response handelt.

9.4.2.3 OCR

Das OCR-Package beinhaltet die Klasse OCR-Engine sowie zwei Header-Dateien, welche für die Tesseract-Library benötigt werden. Die Klasse OCR-Engine ist für die Texterkennung zuständig.

9.4.3 Data Converter

In der Data Converter-Schicht befinden sich die Klassen für die Umwandlung der Objective-C Klassen in die entsprechende XML-Form und umgekehrt.

9.4.3.1 XMLWriter

Das XMLWriter-Subpackage enthält eine PaymentWriter-Klasse, welche den jeweiligen Zahlungstyp in die entsprechende XML-Form umwandelt.

9.4.3.2 XMLParser

Die Klassen im XML-Parser-Subpackage erhalten vom XMLServiceHandler (aus dem Package ProblemDomain/Handler) die Response in XML-Form und wandeln diese nun wieder zurück in die entsprechende Payment-Klasse.

9.5 Design Pakete

9.5.1 Package User Interface

9.5.1.1 Beschreibung der Pakete

Package	Beschreibung
userinterface	Das Package userinterface ist für die grafische Benutzeroberfläche der finApp verantwortlich.
userinterface.favorites	Die Klassen im Package favorites sind für die Darstellung des Favoriten-Views zuständig.
userinterface.payments	Die Klassen im Package payments sind für die Darstellung der ausgeführten sowie der pendenten Zahlungen zuständig.
userinterface.payment	Die Klassen im Package payment sind für die Darstellung der Zahlungseingabe zuständig.
userinterface.templates	Die Klassen im Package favorites sind für die Darstellung der Zahlungsvorlagen zuständig.
userinterface.settings	Die Klassen im Package settings sind für die Darstellung des App-Einstellungen zuständig.

Tabelle 5 - Beschreibung der Pakete UI

9.5.1.2 Package Übersicht

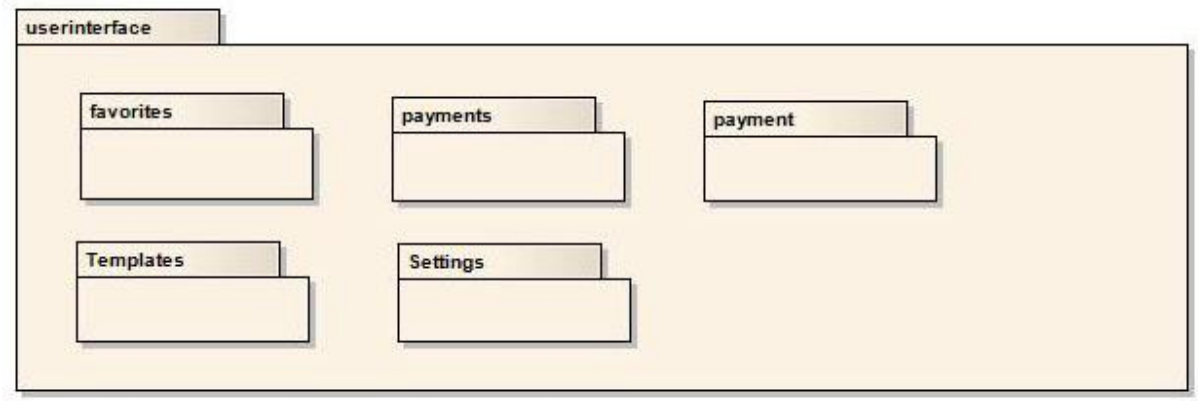


Abbildung 24 - Package Übersicht des UI

9.5.1.3 Klassendiagramm des User Interface

Das unten angefügte Klassendiagramm zeigt die Hauptnavigation, welche mittels einer Tab-bar realisiert wurde (jeder Tab verfügt über einen Navigations-Controller).

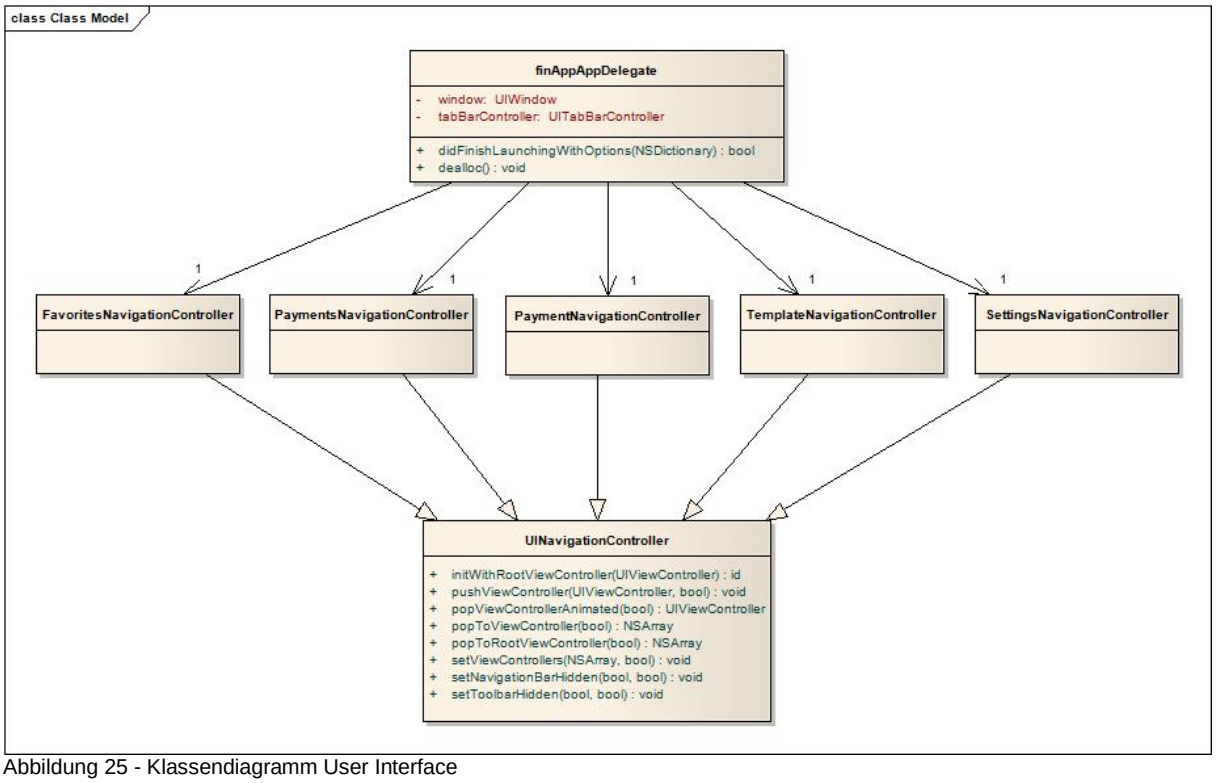


Abbildung 25 - Klassendiagramm User Interface

Klasse	Beschreibung
finAppAppDelegate	Delegate des UIApplication-Objektes. Informiert über Events, welche beim Starten oder Beenden der App auftreten.
Template-, Favorites-, Payments-, Payment-, SettingsNavigationController	Implementiert einen View-Controller, welcher für das Management von hierarchischen Inhalten verantwortlich ist. Für jeden Tab-Menüpunkt gibt es jeweils solch einen Controller.

Tabelle 6 - Klassendiagramm UI

9.5.1.4 Klassendiagramme Sub Packages

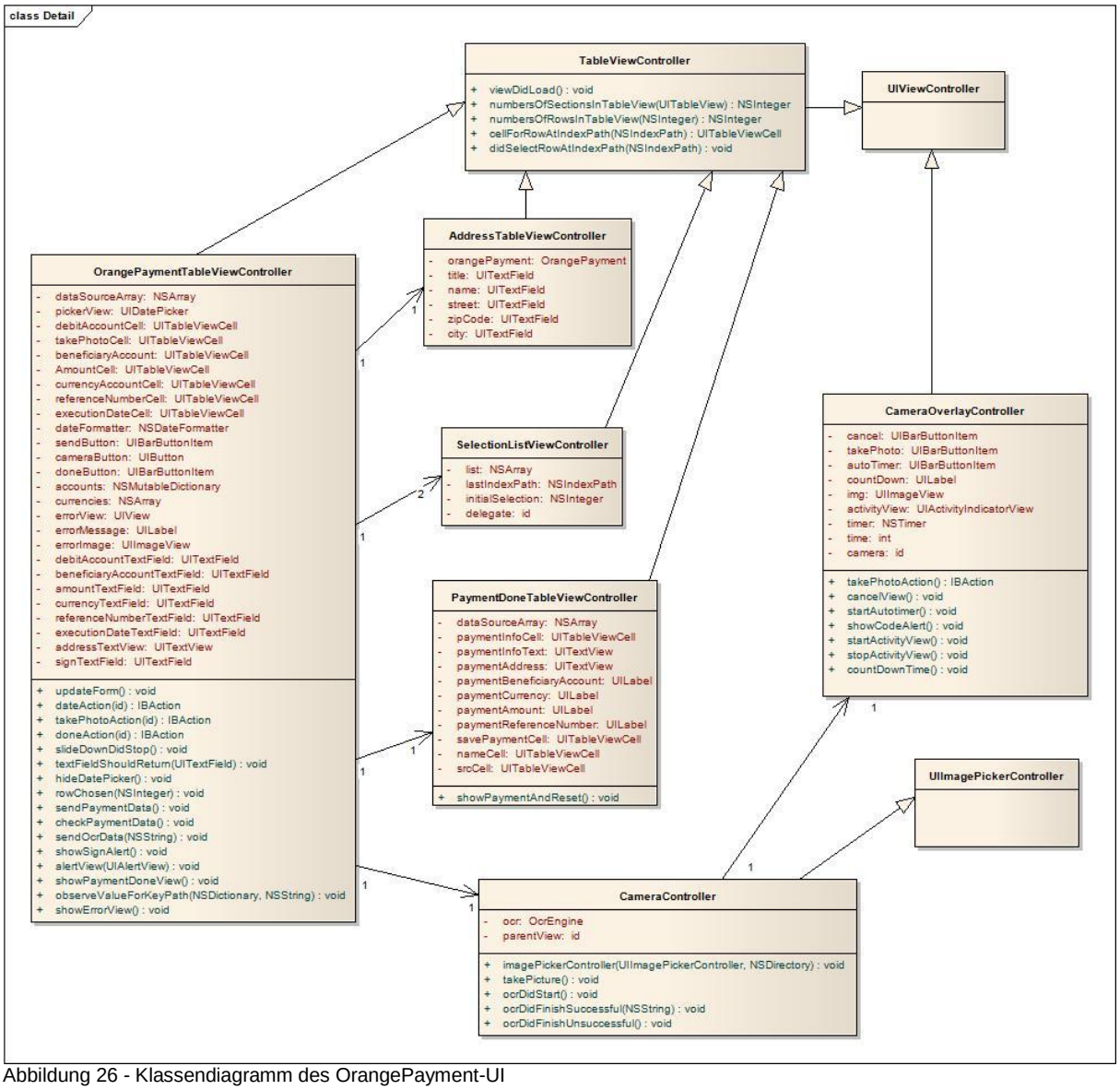


Abbildung 26 - Klassendiagramm des OrangePayment-UI

Klasse	Beschreibung
OrangePaymentTableViewController	Ist für die Verwaltung eines TableViews zuständig. Der TableView beinhaltet das Eingabeformular für die Zahlung eines orangen Einzahlungsscheins.
SelectionListViewController	Ist für die Darstellung einer SelectionsListe verantwortlich. Beispiel: Auswahl der Währung
CameraController	Managed User-Interaktionen welche beim Fotografieren entstehen.
CameraOverlayController	Ist für die Darstellung der Maske, welche dem Benutzer beim Fotografieren unterstützt zuständig.

AddressTableViewController	Ist für die Darstellung des Adresseingabe-Formulars zuständig
PaymentDoneTableViewController	Ist für die Darstellung einer abgeschlossenen Zahlung zuständig

Tabelle 7 - Klassendiagramm OrangePayment

9.5.1.5 Operationen

Das erste Sequenzdiagramm stellt den Ablauf dar, bei dem der Benutzer einen Einzahlungsschein fotografiert, um eine umständliche manuelle Formulareingabe zu vermeiden.

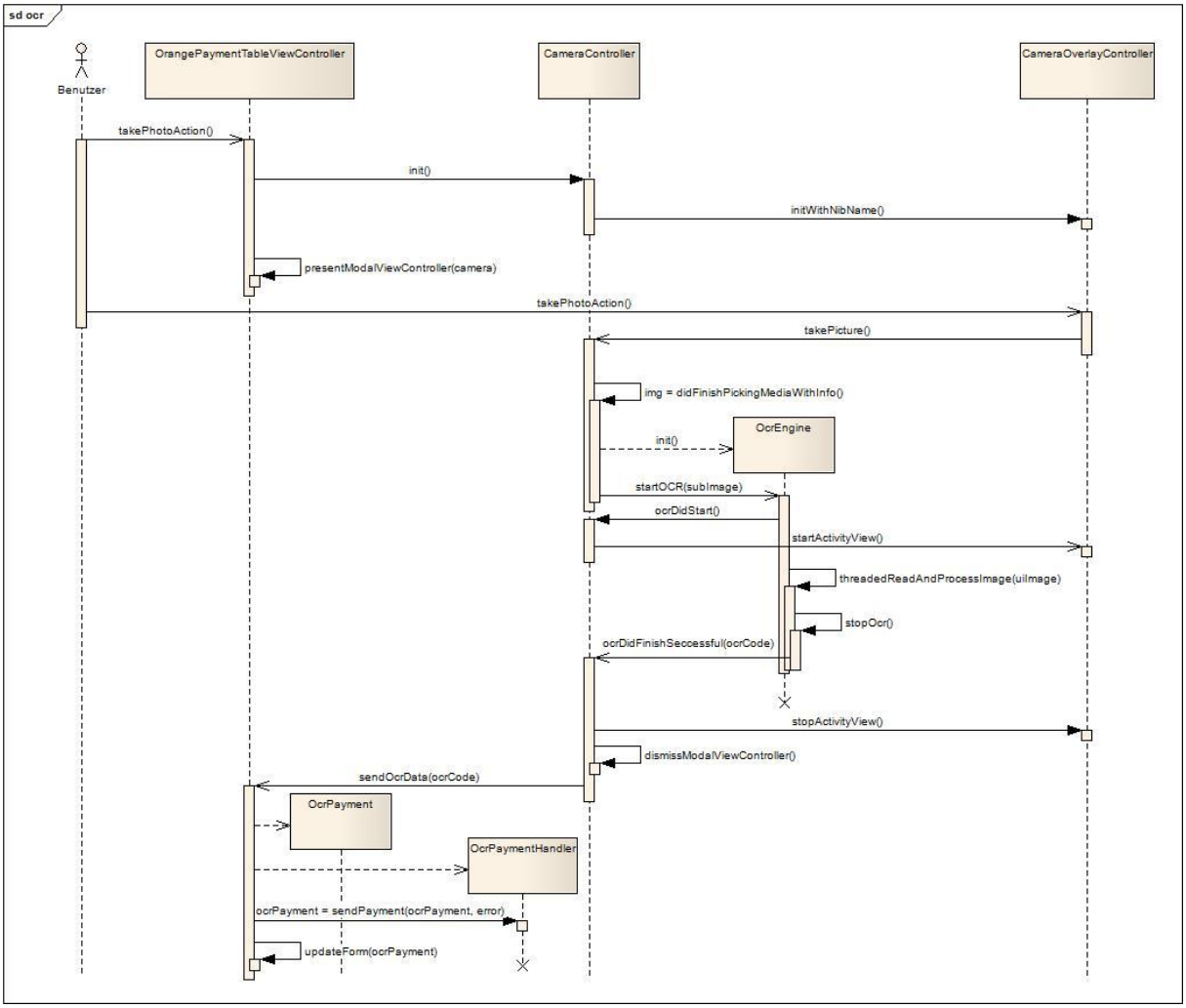


Abbildung 27 - SSD Fotografieren eines Einzahlungsscheins

Das zweite Sequenzdiagramm stellt den Ablauf dar, welcher beim Ausführen einer Zahlung auftritt. Das Formular ist vollständig ausgefüllt und der Benutzer sendet mittels Button die Zahlung an den Server. Daraufhin sind drei verschiedene Abläufe möglich.

- Die Zahlung ist gültig und wird vom Server akzeptiert.
- Die Zahlung enthält ein ungültiges Feld und der Benutzer wird darauf hingewiesen.
- Die Zahlung muss signiert werden.

Abbildung 28 - SSD Zahlungsverkehr

9.5.2 Package Problem Domain und Utilities

9.5.2.1 Beschreibung der Pakete

Package	Beschreibung
problemdomain	Das Package problemdomain ist für die Logik der finApp verantwortlich. Je nach Eingaben des Benutzers werden verschiedene Tätigkeiten (z.B unterschiedliche Zahlungen) ausgeführt.
problemdomain.payment	Das Subpackage payment beinhaltet die drei Zahlungs-Datenklassen (Orange-, OCR- und Sign-Payment). Diese Klassen repräsentieren die Zahlungsobjekte und enthalten mehrere Attribute (inkl. Properties).
problemdomain.handler	Die Klassen im Subpackage handler sind für das Umwandeln der Zahlungsobjekte und das Senden resp. Empfangen der umgewandelten Objekte an den resp. vom Server verantwortlich.
problemdomain.ocr	Das Package ocr beinhaltet die Header-Dateien der Tesseract-Library (C++), sowie die von uns geschriebene Klasse OcrEngine. Mithilfe der TessBaseAPI-Schnittstelle erkennt die OcrEngine den Text bzw. die Codierzeile auf einem Foto. Da die Analyse eines Fotos ein paar Sekunden dauern kann, wird diese Aufgabe in einen Thread ausgelagert. Dies ist notwendig, damit kein blockieren des GUIs auftritt.
dataconverter	Das Package dataconverter stellt sowohl die Methoden zur Umwandlung eine Zahlungsobjektes in einen XML-String (xmlwriter) als auch die Methoden zur Umwandlung des XML-Strings zurück in ein Zahlungsobjekt (xmlparser) zur Verfügung.
dataconverter.xmlwriter	Das Subpackage xmlwriter beinhaltet eine Klasse zur Umwandlung einer Objective-C-Klasse in einen XML-String. Dieser XML-String wird danach von einer handler-Klasse an den Server geschickt.
dataconverter.xmlparser	Die Response des Servers wird widerum im XML-Format geliefert, das Subpackage xmlparser wandelt diese Response wieder um und setzt die Daten des XML's in die Properties der Zahlungs-Objekte ein.

Tabelle 8 - Beschreibung der Pakete PD_DC

9.5.2.2 Package Übersicht

Abbildung 29 - Packages der Problem Domain und des DataConverter**Klassendiagramm Problem Domain und DataConverter**

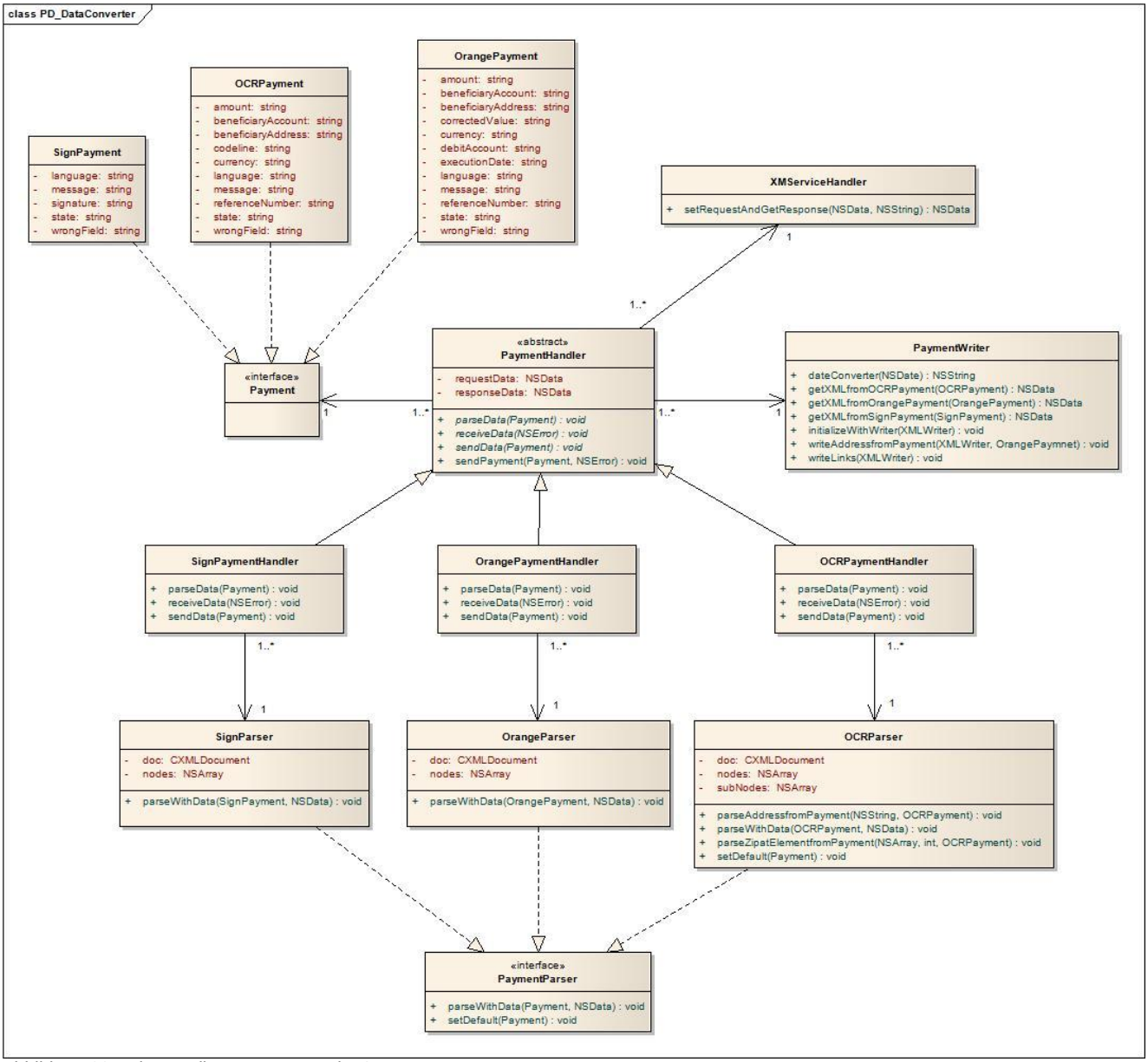


Abbildung 30 - Klassendiagramm PD und DC

9.5.2.4 Klassendiagramme Sub Packages

Payment

Abbildung 31 - Klassendiagramm Payment

Klasse	Beschreibung
Payment	Protokoll (Interface) für die unterschiedlichen Zahlungstypen
OCR-, Orange-, SignPayment	Zahlungsobjekte mit den benötigten Attributen und den dazugehörigen Properties. Diese Objekte werden von den jeweiligen Handler in ein XML-Dokument umgewandelt.

Tabelle 9 - Klassendiagramm Payment

XMLParser / -Handler

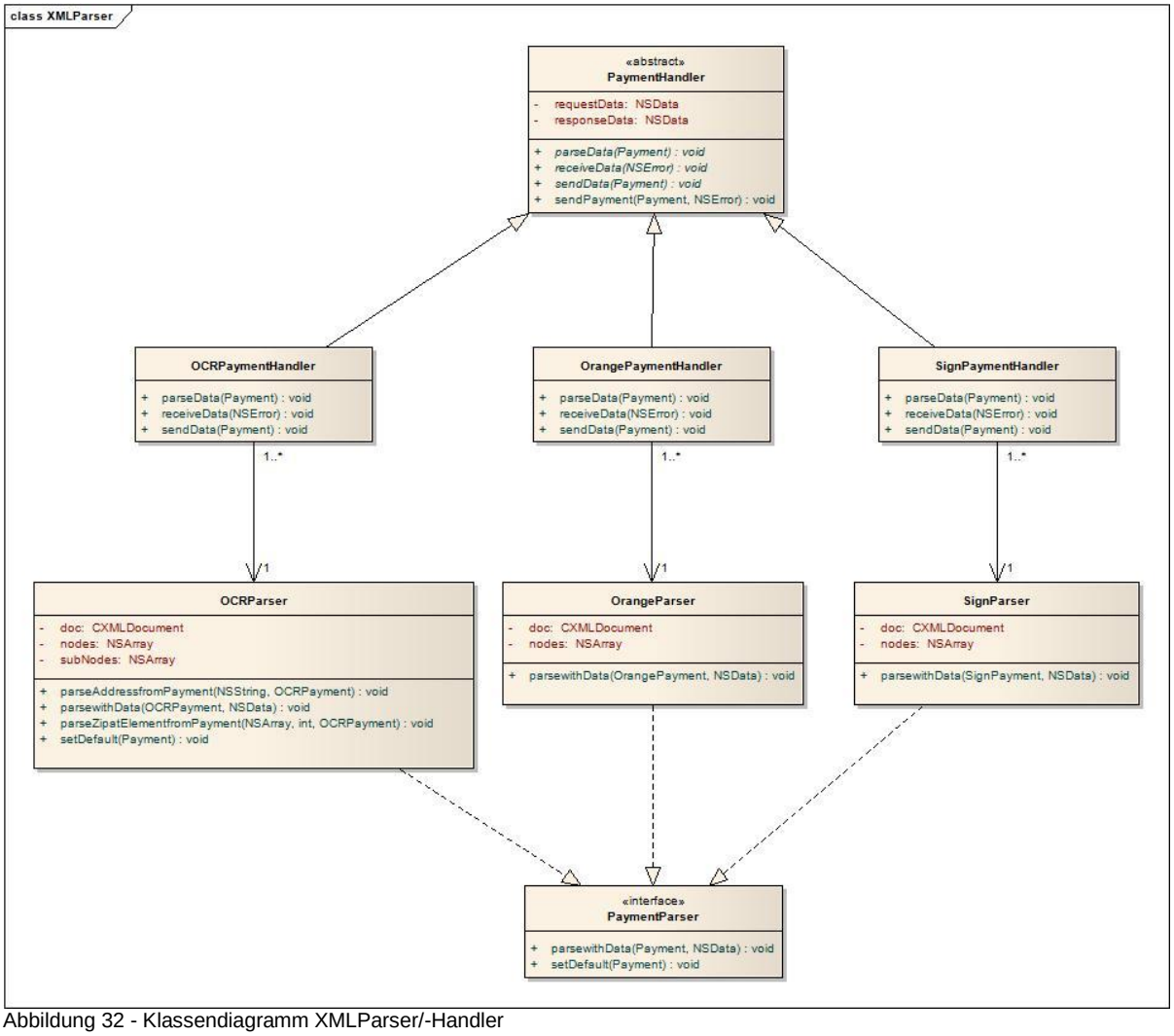


Abbildung 32 - Klassendiagramm XMLParser/-Handler

Klasse	Beschreibung
PaymentHandler	Abstrakte Klasse, welche die Template Methode sendPayment() enthält und drei abstrakte Methoden
OCR-, Orange-, SignPaymentHandler	Die drei Handler implementieren die drei abstrakten Methoden sendData(), receiveData() und parseData() der PaymentHandler-Klasse.
PaymentParser	Protokoll (Interface), welches die parseWithData-Methode beinhaltet
OCR-, Orange-, SignParser	Diese Klasse enthält eine Methode parseWithData(), mit den Parametern Payment (Zahlungsobjekt) und xmlData (entspricht dem XML-String des Servers, welcher über die beiden Handler zum Parser gelangt). Die parse()-Methode wandelt diesen XML-String (xmlData) um und speichert die Werte in die jeweiligen Properties des Zahlungsobjektes (Payment).

Tabelle 10 - Klassendiagramm XMLParser/-Handler

XMLWriter

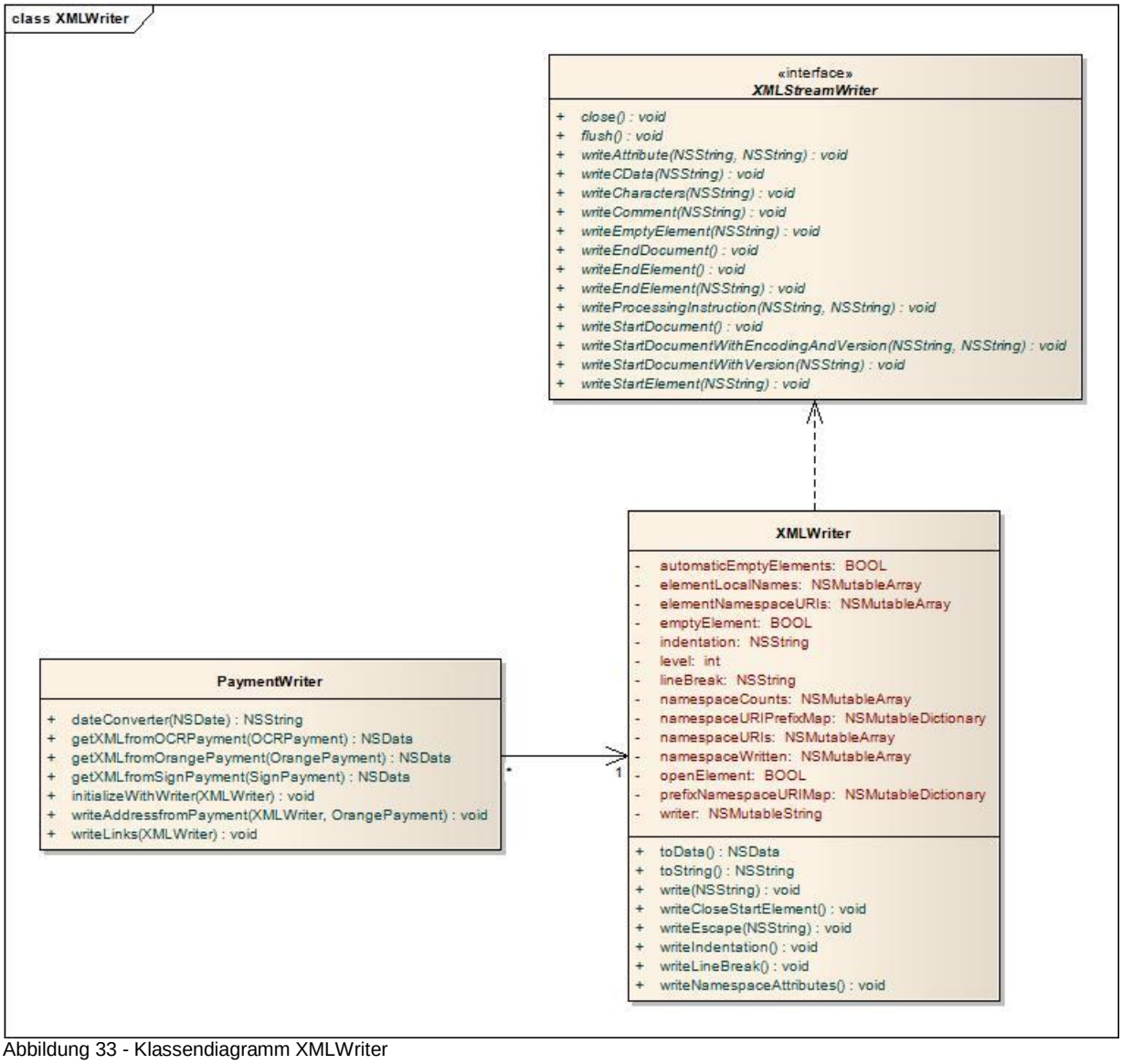


Abbildung 33 - Klassendiagramm XMLWriter

Klasse	Beschreibung
XMLStreamWriter	Interface für die Serialisierung eines XML ohne Namespaces
XMLWriter	Implementiert das XMLStreamWriter Interface
PaymentWriter	Bietet die Methoden an für die Umwandlung der Objective-C-Klassen (Payments) in XML-Files

Tabelle 11 - Klassendiagramm XMLWriter

XMLServiceHandler

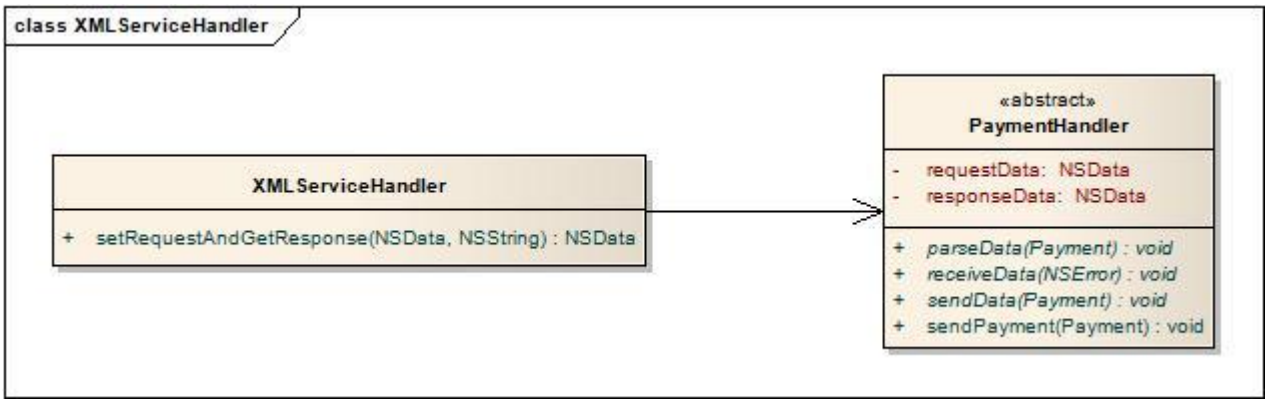


Abbildung 34 - Klassendiagramm XMLServiceHandler

Klasse	Beschreibung
PaymentHandler	Abstrakte Klasse mit der Template Methode sendPayment(Payment)
XMLServiceHandler	Enthält die Methode setRequestAndGetResponse() mit den Parametern xmlData und einem urlString. Die XMLServiceHandler-Klasse erhält von einem der PaymentHandler den erstellten XML-String (xmlData) und sendet diesen an den Server (urlString). Danach liest er die Response des Servers und gibt diese in Form eines XML-Dokuments an den PaymentHandler zurück.

Tabelle 12 - Klassendiagramm XMLServiceHandler

HandlerHelper

Die Handler-Helper-Klasse beinhaltet eine Reihe von globalen Variabeln (z.B. die URL des eMob-Converter). Da es sich bei der finApp lediglich um einen Prototypen handelt, wurden die Environment-Properties statisch implementiert, d.h. sie sind nicht konfigurierbar.

9.5.2.5 Operationen

Das erste Sequenzdiagramm stellt den Ablauf eines OCR-Payment dar, d.h. der User fotografiert mit der Kamera die Codierzeile eines orangen Einzahlungsscheins. Die Daten werden in XML umgewandelt und an den Server (eMob-Converter) geschickt. Die Antwort wird aus einem XML direkt in ein OrangePayment umgewandelt und dieses wird wiederum an den als Request an den Server geschickt und die Zahlung wird ausgeführt.

Abbildung 35 - SSD OCRPaymentHandlingDas zweite Sequenzdiagramm beschreibt das Signieren einer Zahlung. Nachdem ein OrangePayment an den Server geschickt wurde, erhält der User eine Signierungsaufforderung.
Dieser gibt den geheimen Code ein und ein SignPayment wird an den eMob-Converter geschickt. Falls dieses SignPayment akzeptiert wird, wird die Zahlung ausgeführt.

Abbildung 36 - SSD SignPaymentHandling

9.6 XML-Struktur

In diesem Kapitel wird die verwendete XML-Strukturen anhand vom OrangePayment- und vom OCR-Payment-Request dargestellt. Diese dienen lediglich dem besseren Verständnis resp. der Veranschaulichung.

9.6.1 Struktur des OrangePayment

```
<?xml version="1.0" encoding="UTF-8"?>
<fmb:OrangePaymentRequest
    beneficiaryAccount="01-1595-7"
    debitAccount="1004412"
    executionDate="2011-02-07"
    language="deu"
    referenceNumber="216006212194000000104053638"
    beneficiaryAddress="HSR Hochschule für TEchnis&#xA;Oberseestrasse 10&#xA;8640 Rapperswil"
    xmlns:fmb="http://www.finnova.ch/mobilebanking"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.finnova.ch/mobilebanking OrangePaymentRequest.xsd ">
  <fmb:amount amount="1.50" currency="CHF"/>
</fmb:OrangePaymentRequest>
```

Abbildung 37 - OrangePaymentRequest

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OrangePaymentResponse xmlns="http://www.finnova.ch/mobilebanking"
    message="Ihr Auftrag über CHF 1.50 wurde entgegengenommen und
    wird bei entsprechendem Guthaben am 07.02.2011 ausgeführt."
    state="1"/>
</OrangePaymentResponse>
```

Abbildung 38 - OrangePaymentResponse

9.6.2 Struktur des OCRPayment

```
<?xml version="1.0" encoding="UTF-8"?>
<fmb:OCRPaymentRequest
    codeline="0100000039305>040000048008775400073677679+010422639>"
    language="deu" xmlns:fmb="http://www.finnova.ch/mobilebanking"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.finnova.ch/mobilebanking OCRPaymentRequest.xsd ">
</fmb:OCRPaymentRequest>
```

Abbildung 39 - OCRPaymentRequest

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OCRPaymentResponse xmlns="http://www.finnova.ch/mobilebanking"
    referenceNumber="040000048008775400073677679"
    beneficiaryAddress="Buch.ch AG&#xA;Industriestrasse 26&#xA;8411 Winterthur"
    beneficiaryAccount="01-42263-9" state="1">
    <amount currency="" amount="39.3"/>
</OCRPaymentResponse>
```

Abbildung 40 - OCRPaymentResponse

9.7 Serverarchitektur – Spezifikation des eMob-Converter

9.7.1 Systembeschreibung

Der eMob-Converter ist eine Java Webapplikation, welche auf einem Server der Finnova implementiert wird. Als Servlet-Container für die Webapplikation wird der Apache Tomcat [25] in der Version 6.0.29 mit dem Java Runtime Environment 1.6.0.21 (64-Bit) verwendet. Das Loggen der Anwendungsmeldungen übernimmt das log4j Framework Version 1.2 der Apache Software Foundation [26].

Die Applikation beinhaltet einen Filter (AuthorizationFilter.class), welcher alle eingehenden Requests auf Ihre Berechtigung überprüft (Session vorhanden und am System authentisiert) und aus Java Servlets, welche jeweils einen Service abarbeiten. Alle Packages der Applikation werden mit dem Prefix ch.finnova.mobile gekennzeichnet.

Das Marshalling und Unmarshalling der XML-Daten wird mithilfe von Java Architecture for XML Binding (JAXB) 2.0, welches ein Teil der Java Platform, Standard Edition, 6.0 ist, umgesetzt.
JAXB ermöglich eine automatische Datenbindung aus einer XML-Schema-Instanz an Java-Klassen, welche ihrerseits mittels dem Schema generiert worden sind.

Für die Unittests wird das Testframework TestNG in Kombination mit dem HttpClient von der Apache Software Foundation eingesetzt.

9.7.2 Architektur

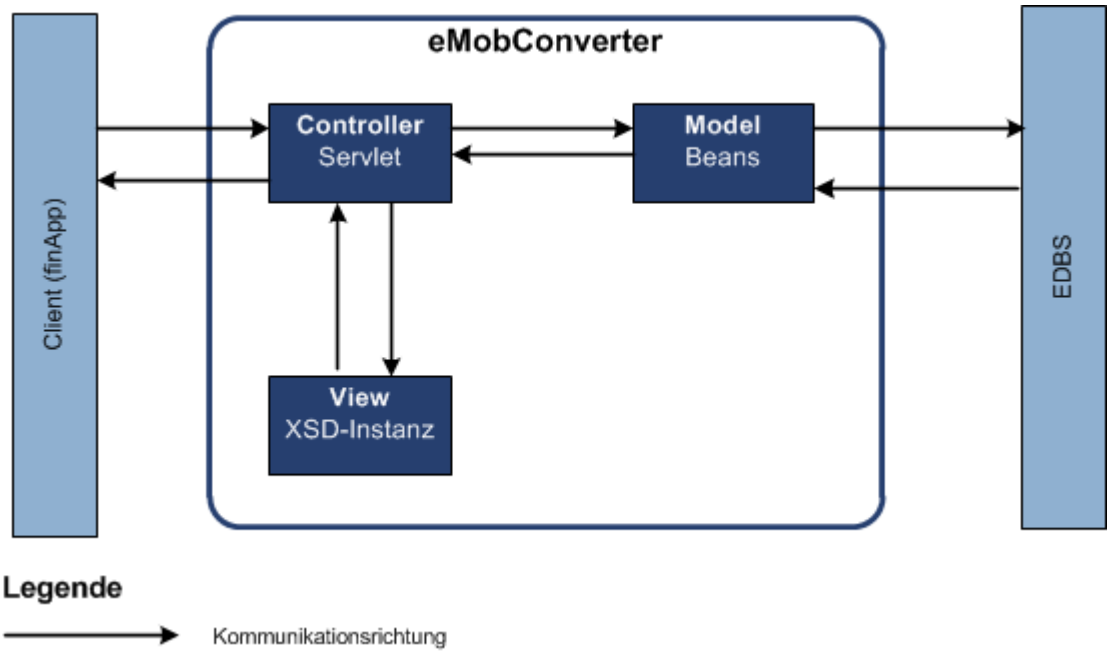


Abbildung 41 - Architektur

Die Architektur des eMob-Converters hält sich an das MVC-Pattern, wobei wie folgt die drei Einheiten betrachtet werden können.

- Das **Model** des eMob-Converters sind Java-Klassen (Beans), welchen mittels dem EDBS Zugriff auf den Finnova-Core ermöglich wird.
- Die **Controller** werden von den Servlets dargestellt, wobei zu beachten ist, dass für jeden Service, der über die XML-Schnittstelle angeboten wird, ein Servlet existiert.
- Die **View** wird mithilfe von JAXB als XML-Schema-Instanzen dargestellt. Clients müssen diese Daten parsen und selbst eine View implementieren

9.7.3 Klassendiagramm

9.7.3.1 Übersicht

In der Abbildung 42 - Übersicht ist ersichtlich, welche Packages für den Betrieb des eMobC benötigt werden.

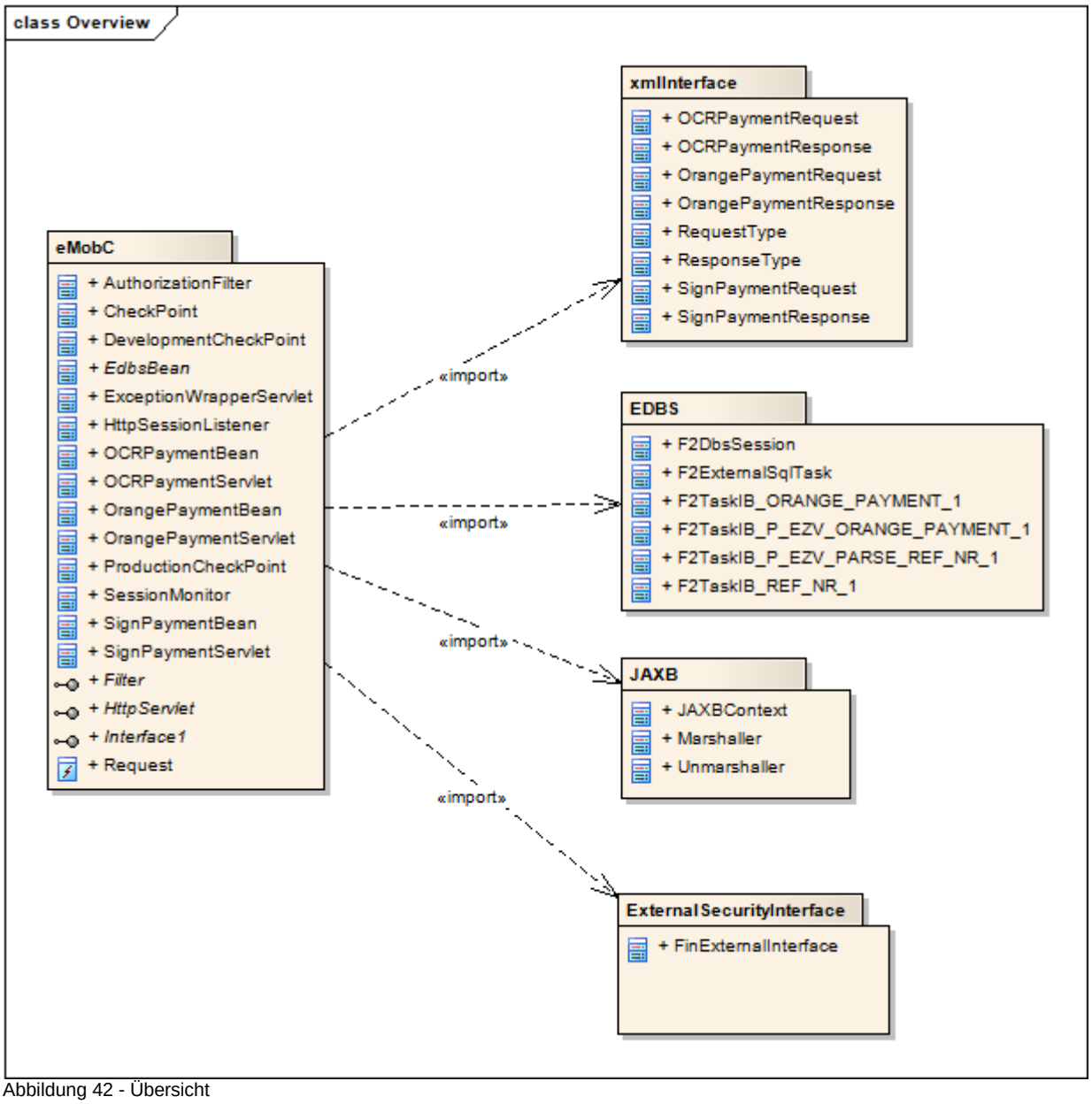


Abbildung 42 - Übersicht

9.7.3.2 eMobC

Der eMobC beinhaltet alle Klassen, welche für die Abarbeitung eines Services benutzt werden.

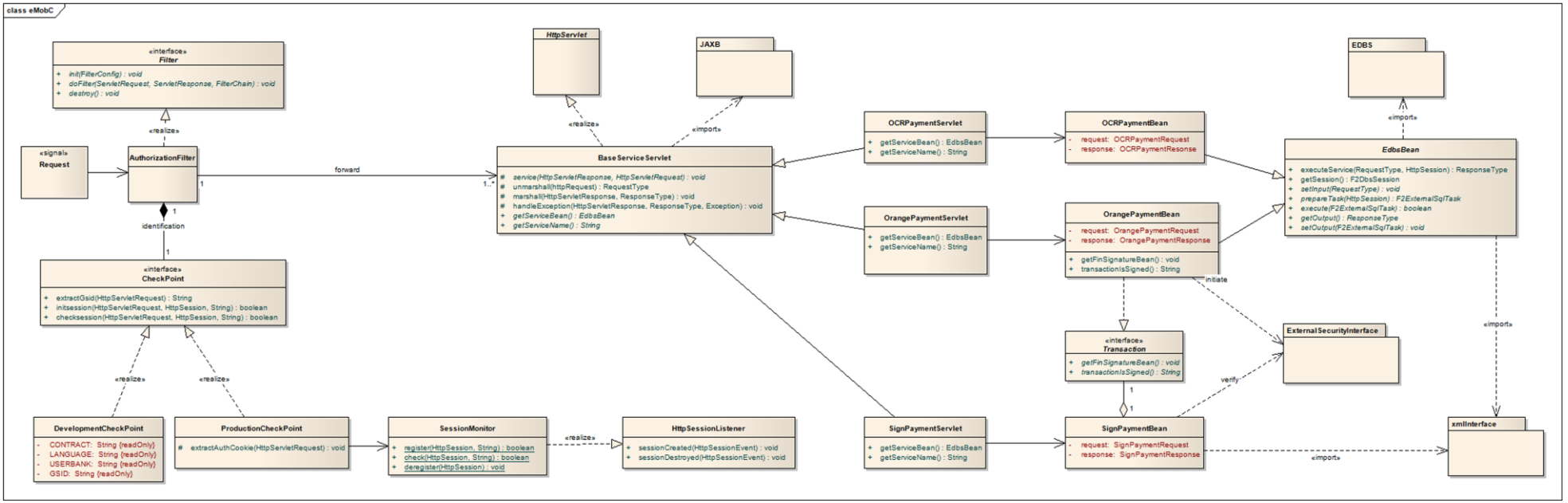


Abbildung 43 - eMobC

Im Anhang befindet sich noch eine grössere Version dieses Klassendiagramms [ANH08].

Klasse	Beschreibung
AuthorizationFilter	Diese Klasse ist der Single Access Point [27] des eMobC und überwacht jeden eingehend Request. Die Identifikation der einzelenen Requests wird an den CheckPoint delegiert. Besteh noch keine HttpSession oder DbsSession wird diese vom AuthorizationFilter erstellt.
CheckPoint	Der CheckPoint [28] wird eingesetzt, um unauthorisierte Zugriffe zu erkennen und diese dem AuthorizationFilter zurückzumelden. Besteht noch keine gültige HttpSession, überprüft der CheckPoint folgende Credentials, welche im Request enthalten sein müssen: • GSID (Global Session Identifier) • Vertragsnummer • Sprache • Userbank • Sind alle Credentials vorhanden und die Gültigkeit ist gewährleistet, wird eine neue HttpSession initialisiert und beim SessionMonitor registriert • Besteht bereits eine gültige Session, wird diese über den SessionMonitor auf ihre Gültigkeit überprüft. Zu Entwicklungs- und Testzwecken kann der DevelopmentCheckPoint geladen werden, welcher die Credentials als Konstanten definiert hat
SessionMonitor	Der SessionMonitor implementiert den HttpSessionListener und ermöglicht es, neue Sessions zu registrieren, nach einem Logout eine Session zu entfernen und die Gültigkeit einer Session zu überprüfen. Bei der Registrierung wird die GSID der Session und die Session-ID abgespeichert. Mittels diesen Credentials kann überprüft werden, ob eine HttpSession noch gültig ist.
BaseServiceServlet	Das BaseServiceServlet ist die Oberklasse aller Service-Servlets und beinhaltet unmarshalling/marshalling und das Exception-Handling der ableitenden Klassen. Die Abarbeitung eines Service-Requests ist jeweils in der service-Methode der abgeleiteten Klassen implementiert. Der Algorithmus der service-Methode beinhaltet folgende Schritte für jeden eingehenden Serviceaufruf: • Unmarshalling der spezifischen XML-Schema-Instanz (abgeleitet von RequestType), welche im Body des HTTP-Post Request übermittelt wird • Binding der eingehenden Daten an das zugehörige Bean. • Auslösen der Übermittlung der Daten an den Finnova-Core • Marshalling der Antwort des Finnova-Core in die spezifische XML-Schema-Instanz (abgeleitet von ResponseType) • Einbetten der Response in den Response-Body Die Abhandlung eines Services ist im Kapitel 9.7.4 (Interaktionen zwischen finApp und eMobC) in Form eines Interaktionsdiagram-

	mes dargestellt.
OCRPaymentServlet	Spezifisches Servlet, das den Service OCRPaymentRequest abarbeitet, und eine OCRPaymentResponse in den Outputstream schreibt.
OCRPaymentBean	Bean, welches die empfangenen Daten aus dem OCRPaymentRequest dem EDBS-Task übergibt, die Antwort des Tasks ausliest und abspeichert.
OrangePaymentServlet	Spezifisches Servlet, das den Service OrangePaymentRequest abarbeitet, und eine OrangePaymentResponse in den Outputstream schreibt.
OrangePaymentBean	Bean, welches die empfangenen Daten aus dem OrangePaymentRequest dem EDBS-Task übergibt und die Antwort des Tasks ausliest und abspeichert. Das OrangePaymentBean erweitert die abstrakte Klasse EDBSBean und implementiert das Interface Transaction. Das Interface Transaction wird dann benötigt, wenn die Zahlung vom Benutzer signiert werden muss. Ist dies der Fall, wird die Transaction in der HttpSession gespeichert und dem SignPaymentBean somit ein Interface geliefert, über welches die Transaction zu einem späteren Zeitpunkt wiederaufgenommen werden kann.
SignPaymentServlet	Spezifisches Servlet, das den Service SignPaymentRequest abarbeitet, und eine SignPaymentResponse in den Outputstream schreibt.
SignPaymentBean	Bean, welches die empfangenen Daten aus dem SignPaymentRequest dem ExternalSecurityInterface übergibt.
EdbsBean	Abstrakte Java Klasse, über welche eine ableitende Klasse eine F2DbsSession für einen EDBS-Call erhält und das Interface der ableitenden Klassen definiert ist. Die Klasse stellt die Template-Methode executeService bereit, welche vom BaseServiceServlet aufgerufen werden kann.
Transaction	Interface für die Kommunikation zwischen einem PaymentBean und dem SignPaymentBean.

Tabelle 13 - Beschreibung eMobC

9.7.3.3 JAXB

Das Package JAXB beinhaltet die Klassen, welche vom JAXB-Framework zur Verfügung gestellt werden. Hier werden kurz die drei wichtigsten Klassen beschrieben.

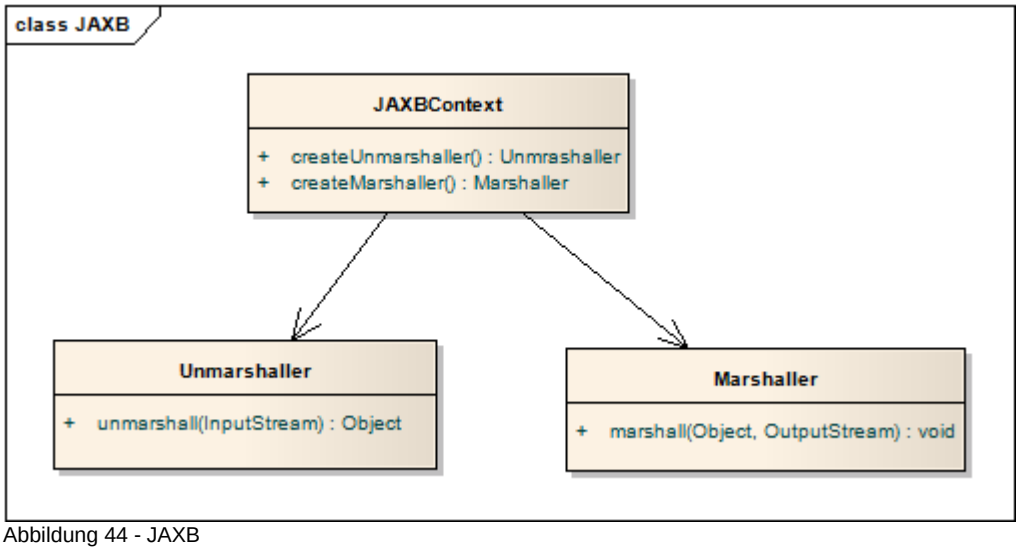


Abbildung 44 - JAXB

Klasse	Beschreibung
JAXBContext	Über den JAXBContext kann der Namespace der generierten XML-Instanzen angegeben werden, über diesen Namespace kann der JAXBContext die konkreten Java-Klassen für einen Service laden.
Unmarshaller	Dem Unmarshaller wird der InputStream des Requests übergeben. Via dem JAXBContext wird das Binding der XML-Instanz an die konkrete Requestklasse vorgenommen.
Marshaller	Der Marshaller schreibt einen ResponseType in den übergebenen ServletOutputStream.

Tabelle 14 - Beschreibung JAXB

9.7.3.4 XML-Interface

Im Package XML-Interface sind die aus den Schemas (mittels JAXB) generierten Klassen enthalten. Diese werden für das Marshalling und Unmarshalling verwendet.

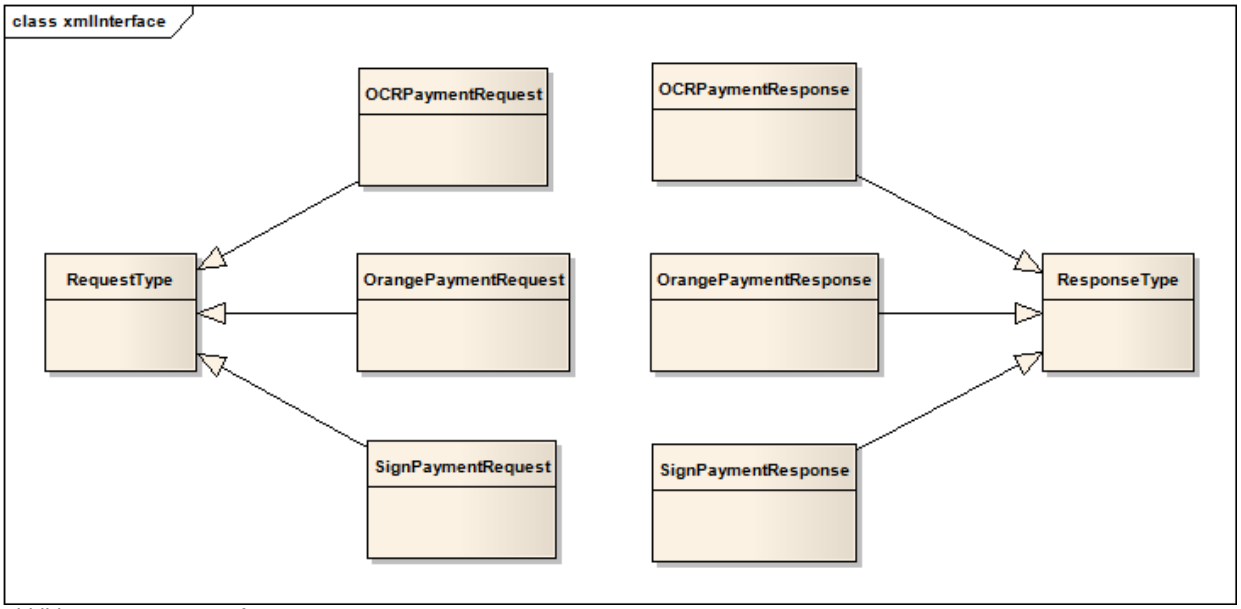


Abbildung 45 - XML-Interface

Klasse	Beschreibung
RequestType	Oberklasse für alle eingehenden Services. Im RequestType muss immer die Sprache enthalten sein.
ResponseType	Oberklasse für alle ausgehenden Services. Über den ResponseType ist es möglich, eine oder mehrere Exceptions zu retournieren.
OCRPaymentRequest	Servicebeschreibung der Anfrage mittels XML-Schema, welche an den eMob-Converter gesendet wird. Die XML-Schema-Instanz wird im Body des HTTP-Post Request übermittelt.
OCRPaymentResponse	Antwort auf den Service OCRPayment, welche vom eMob-Converter retourniert wird. Die Beschreibung des Service wird mittels einem XML-Schema vorgegeben.
OrangePaymentRequest	Servicebeschreibung der Anfrage mittels XML-Schema, welche an den eMob-Converter gesendet wird. Die XML-Schema-Instanz wird im Body des HTTP-Post Request übermittelt.
OrangePaymentResponse	Antwort auf den Service OrangePayment, welche vom eMob-Converter retourniert wird. Die Beschreibung des Service wird mittels einem XML-Schema vorgegeben.

SignPaymentRequest	Servicebeschreibung der Anfrage mittels XML-Schema, welche an den eMob-Converter gesendet wird. Die XML-Schema-Instanz wird im Body des HTTP-Post Request übermittelt.
SignPaymentResponse	Antwort auf den Service SignPayment, welche vom eMob-Converter retourniert wird. Die Beschreibung des Service wird mittels einem XML-Schema vorgegeben.

Tabelle 15 - Beschreibung XML-Interface

9.7.3.5 EDBS

Im Package EDBS sind die mittels einem Stubcreator erstellten Klassen für die Kommunikation mit dem Finnova Core enthalten.

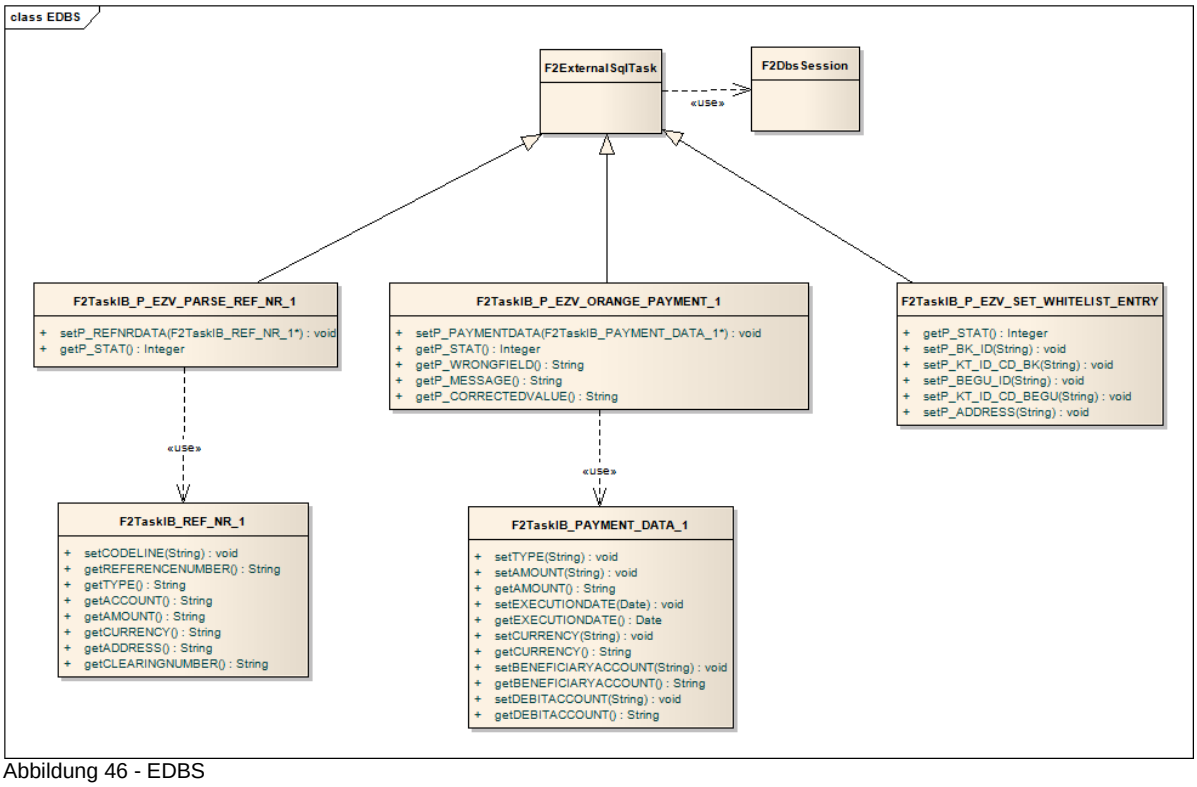


Abbildung 46 - EDBS

Klasse	Beschreibung
F2ExternalSqlTask	Standardklasse des EDBS, welche für die Kommunikation mit dem DBS benutzt werden muss. Alle selbst definierten Task erweitern den F2ExternalSqlTask.
F2DbsSession	Standardklasse des EDBS, welche für die Kommunikation mit dem DBS benutzt werden muss. Für die Kommunikation mittels den Task wird immer ein F2DbsSession Objekt benötigt. Das Pooling der Sessions ist in der Klasse F2DbsSession implementiert, wesshalb im eMobC auf dies keine Rücksicht genommen werden muss.
F2TaskIB_P_EZV_PARSE_REF_NR	Mittels dem Task kann eine Kodierzeile dem Finnova Core übergeben werden. Die vom Core ermittelten Daten werden im Task F2TaskIB_REF_NR_1 gekapselt und an den eMobC retourniert.

F2TaskIB_REF_NR_1	Task der einem Object Type auf der Datenbank entspricht. In diesem Task werden die ermittelten Daten einer Kodierzeile vom Core zum eMobC transferiert.
F2TaskIB_P_EZV_ORANGE_PAYMENT_1	Über diesen Task werden die Daten eines Orangen Einzahlungsschein an den Core übergeben. Die Daten sind im Task F2TaskIB_PAYMENT_DATA_1 gekapselt.
F2TaskIB_PAYMENT_DATA_1	Task der einem Object Type auf der Datenbank entspricht. In diesem Task werden die Zahlungsdaten an den Finnova Core transferiert.
F2TaskIB_P_EZV_SET_WHITELIST_ENTRY	Über diesen Task kann ein signierter Begünstigter in der Whitelist aufgenommen werden.

Tabelle 16 - Beschreibung EDBS

9.7.4 Interaktionen zwischen finApp und eMobC

9.7.4.1 Genereller Serviceablauf

Das Interatkionsdiagramm Abbildung 47 - SSD Service zeigt die erforderlichen Sequenzen auf, welche für einen Serviceaufruf benötigt werden.

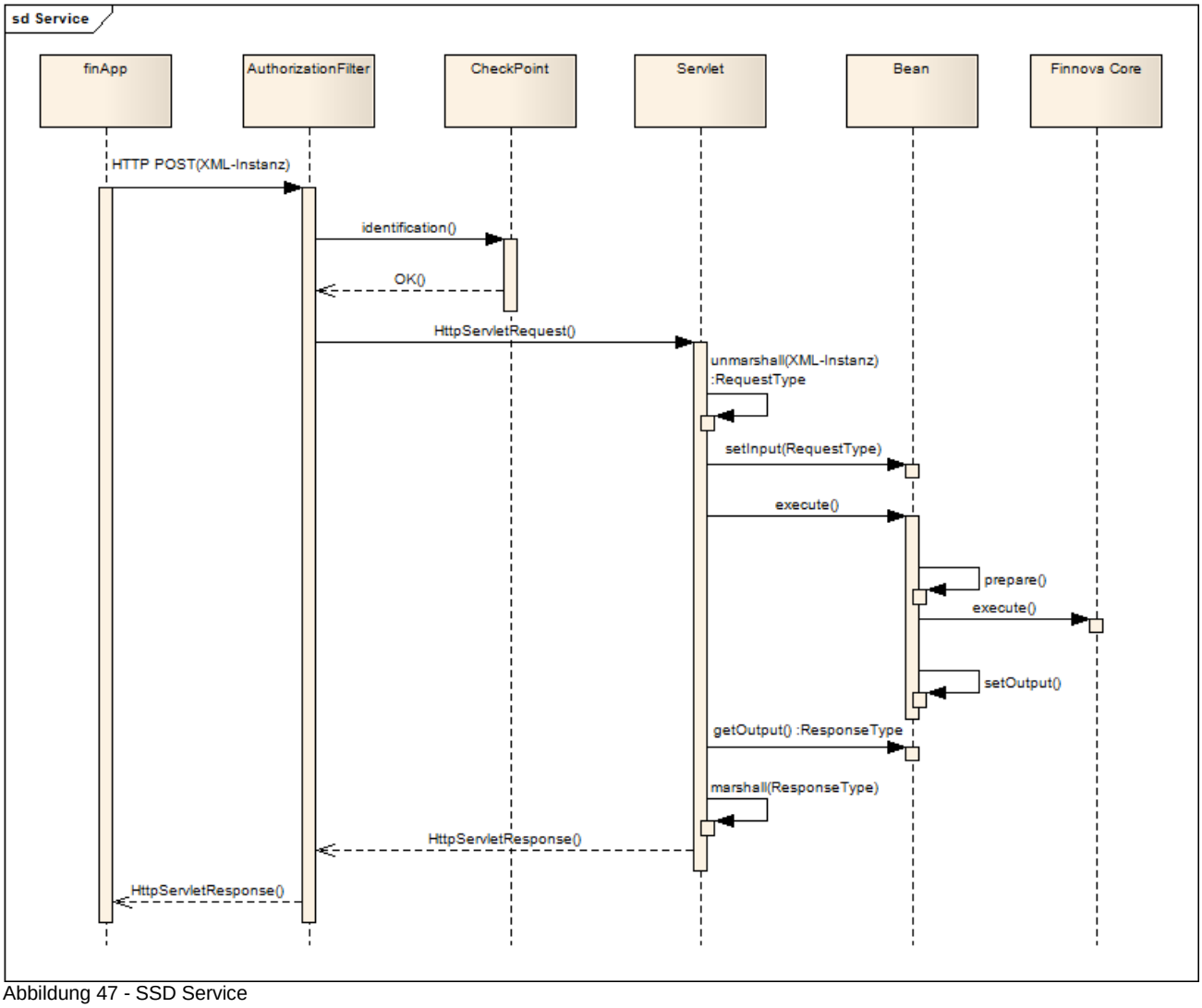


Abbildung 47 - SSD Service

9.7.4.2 Zahlung signieren

Das Interaktionsdiagramm Abbildung 48 - SSD signieren zeigt die erforderlichen Sequenzen auf, wenn eine Zahlung signiert werden muss. Die Zwischenschritte, welche im Diagramm Abbildung 47 - SSD Service aufgezeigt sind, wurden in diesem Sequenzdiagramm weggelassen.

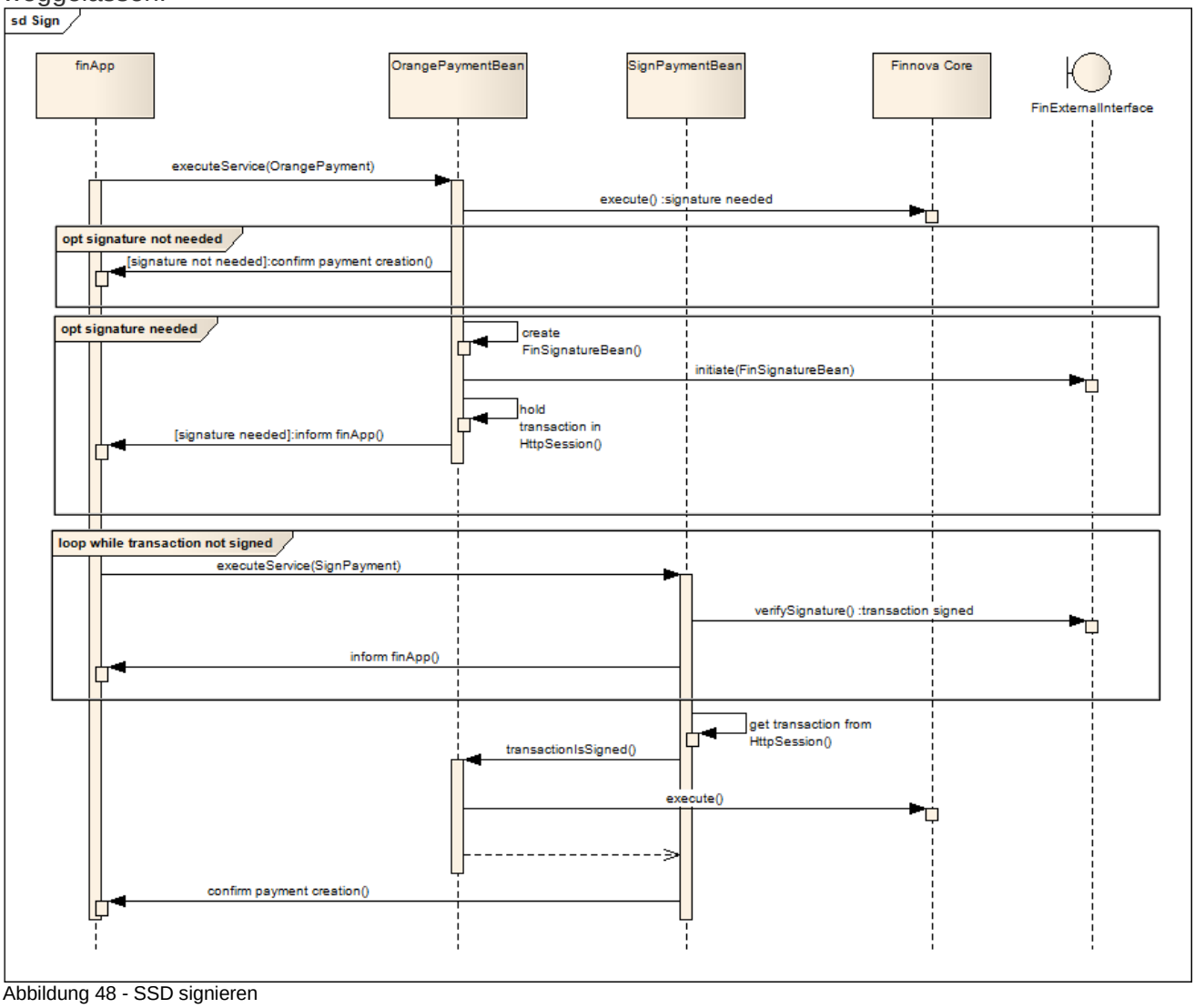


Abbildung 48 - SSD signieren

9.7.5 Konfigurationsmanagement des Servers

9.7.5.1 Concurrent Version System (CVS)

Der Software-Quelltext des eMob-Converters wird in das bestehende CVS des Internetbankings der Finnova integriert. Eine Trennung der Bereiche Internetbanking und Mobilebanking ist im Moment noch nicht vorgesehen. In der Verzeichnisstruktur des CVS werden die Ressourcen wie folgt eingebunden (ausgehend vom finnojet-root):

Verzeichnis	Beschreibung
src/java/mobilebanking/ch.finnova.mobilebanking.eMobConverter	Ordner inklusive Package, der die Sofware-Quelltexte des eMob-Converter beinhaltet
src/java/mobilebanking/ch.finnova.mobilebanking.xmlInterface	Ordner inklusive Package, der die Java-Klassen der XML-Schnittstelle, welche mittels JAXB generiert werden, beinhaltet
rsc/conf/ib.conf	Konfigurationstemplate für die Banken, wird für die Installation verwendet
rsc/mobilebanking/web.xml	Konfigurationsfile der Webapplikation
rsc/mobilebanking/log4j.xml	Konfigurationsfile des Loggers
rsc/mobilebanking/interface	Ordner, der die XSD-Dateien, welche die XML-Schnittstelle beschreiben, beinhaltet
install/mobilebanking/mobilebanking.install	Installationsskript, welches bankindividuelle Parametrierungen der Applikation hinzufügt, und das WAR auf einem Tomcat installiert
install/mobilebanking/mobilebanking.uninstall	Deinstallationsskirpt, welches alle Installierten Komponenten entfernt
build.xml	Konfiguration der Ant-Tasks, welche die automatische Übersetzung sowie das Erstellen der fertigen Applikation vornehmen

Tabelle 17 - eMobC CVS

9.7.5.2 Deployment und Distribution

Unter Deployment wird bei der Finnova das Übersetzen der Applikation in der Entwicklungsumgebung verstanden. Eine Distribution ist ein Zip-File, welches an die Bank ausgeliefert wird.
Das Zip-File enthält Konfigurations- und Installationsdateien sowie die WAR.

Das bestehende build.xml, welches die automatische Übersetzung der Applikation vornimmt, wird um folgende Ant-Tasks erweitert:

Taskname	Beschreibung
mobilebanking.compile	Kompilieren des Java-Sourcecode
int.mobilebanking.copy-server-xml	Server.xml-Vorlage in den rsc-Ordner des mobilebanking kopieren.
int.mobilebanking.replace-server-xml-vars.deploy	Konfigurationsvariablen im Server.xml für den Deploy der Applikation ersetzen
int.mobilebanking.replace-log4j-xml-vars.deploy	Konfigurationsvariablen im log4j.xml für den Deploy der Applikation ersetzen
int.mobilebanking.replace-log4j-xml-vars.dist	Konfigurationsvariablen im log4j.xml für die Distribution der Applikation ersetzen
int.mobilebanking.war	Kreieren des WAR File für Entwicklung oder Distribution.
mobilebanking.deploy	Deployment der Applikation in der Entwicklungsumgebung
mobilebanking.dist	Erstellen der Distribution (Zip-File welches Konfigurations- und Installationsdateien sowie das WAR enthält.

Tabelle 18 - eMobC Build Tasks

9.7.6 Spezifikation der Schnittstelle "Mobilgerät-eMobC"

Nachfolgend die Beschreibung der einzelnen Services, die vom eMobC bedient werden. Beschrieben werden alle Inputparameter eines Requests an den Service, sowie die Outputparmater der Response.

9.7.6.1 XML Schemas

Die XML-Schemas zu den einzelnen Services werden mit dem eMobC ausgeliefert und befinden sich im Verzeichnis "rsc/mobilebanking/interface"

9.7.6.2 Service "OCRPayment"

Ermittlung der Zahlungsdaten aus der Codierzeile des roten oder orangen Einzahlungsscheins.

OCRPaymentRequest

Parameter	Beschreibung	Erforderlich
Codeline	fotografierte Codierzeile eines roten oder orangen Einzahlungsscheins	ja

Tabelle 19 - OCRPaymentRequest

OCRPaymentResponse

Parameter	Beschreibung	Erforderlich
State	Status • 1 = OK, Codierzeile erfolgreich geparst • 2 = Fehler, Codierzeile konnte nicht geparst werden • 3 = Ungültig, kein gültiger Empfänger (nicht in Whitelist enthalten)	ja
BeneficiaryAccount	ermitteltes Begünstigtenkonto der Zahlung aus der Codierzeile	ja
Amount	ermittelter Betrag der Zahlung aus der Codierzeile	nein
ReferenceNumber	ermittelte Referenznummer aus der Codierzeile	nein
BeneficiaryAddress	in Whitelist abgelegte Begünstigtenadresse	ja
ClearingNumber	in Whitelist abgelegte Clearingnummer der Bank	nein
BankAddress	in Whitelist abgelegte Bankadresse	nein

Tabelle 20 - OCRPaymentResponse

9.7.6.3 Service "OrangePayment"

Übermitteln der Zahlungsdaten eines orangen Einzahlungsscheins. Wird eine Signierung der Zahlung gefordert, muss zusätzlich der Service "SignPayment" aufgerufen werden.

OrangePaymentRequest

Parameter	Beschreibung	Erforderlich
ExecutionDate	Ausführungsdatum der Zahlung	ja
Amount	Betrag der Zahlung	ja
CurrencyCode	Währungscode der Zahlung	ja
BeneficiaryAccount	Begünstigtenkonto	ja
BeneficiaryAddress	Adresse des Begünstigten (aus Whitelist extrahiert oder vom Benutzer eingegeben)	ja
ReferenceNumber	Referenznummer der Zahlung	ja
DebitAccount	Belastungskonto	ja

Tabelle 21 - OrangePaymentRequest

OrangePaymentResponse

Parameter	Beschreibung	Erforderlich
State	Status • 1 = OK, Zahlung erfolgreich entgegengenommen • 2 = Fehler, die Zahlung hat einen Fehler (siehe Wrongfield) • 3 = Signatur, Signieren der Zahlung notwendig	ja
Wrongfield	ID des Inputparameter, welcher Fehlerhaft ist	nein
Message	Info-/Fehlermeldung, die dem User anzuzeigen ist	nein
CorrectedValue	Falls Fehler vom System behoben werden kann, ist dies der korrekte Wert, welcher noch vom Benutzer bestätigt werden muss (Zahlung wird noch nicht ausgeführt)	nein

Tabelle 22 - OrangePaymentResponse

9.7.6.4 Service "SignPayment"

Signieren der zuletzt übermittelten Zahlung.

SignPaymentRequest

Parameter	Beschreibung	Erforderlich
Signature	zu lieferndes Credential	ja

Tabelle 23 - SignPaymentRequest

SignPaymentResponse

Parameter	Beschreibung	Erforderlich
State	Status • 1 = OK, Zahlung wurde erfolgreich signiert und ausgeführt • 2 = Fehler, Zahlung konnte nicht signiert werden	ja
Message	Info-/Fehlermeldung, die dem User anzuzeigen ist	nein

Tabelle 24 - SignPaymentResponse

10 Tests

10.1System Tests

10.1.1 Betriebssystem Kompatbilität

Da für die Erkennung der Codierzeilen die Klasse NSRegularExpression verwendet wird, ist für die finApp eine iOS-Version von 4.0 oder neuer Voraussetzung.

10.1.2 Performance

Getestete Funktionalität	iPod Touch (2Generation)	iPhone 3G	iPhone 4
Starten der App	1.0 Sekunden	1.3 Sekunden	Unter 1.0 Sekunden
Kamera-View laden	---	1.0 Sekunde Verzögerung	1.0 Sekunde Verzögerung
Ausführen/Senden einer Zahlung	Keine Verzögerung erkennbar	Keine Verzögerung erkennbar	Keine Verzögerung erkennbar

Tabelle 25 - Performance auf verschiedenen Devices

10.1.3 OCR

10.1.3.1 iPhone 4

Auf dem iPhone 4 stellt die Texterkennung beim fotografieren der Codierzeile kein Problem dar. Bei guten Lichtverhältnissen und ruhiger Kameraführung gelingt die Erkennung meistens innerhalb von zwei Versuchen. Probleme können allerdings Auftreten, wenn das Foto nicht gleichmässig belichtet ist (z.B. durch Schatten von Objekten).

10.1.3.2 iPhone 3G

Trotz mehrerer Versuche ist es uns nicht gelungen, die Codierzeile aus einem Foto auszulesen. Auf dem iPhone 3G ist es nicht möglich, ein scharfes Foto der Codierzeile aufzunehmen.

11 Zeitauswertung

Das erste Diagramm (Abbildung 49) zeigt den Soll-/Ist-Zustand. Es ist ersichtlich, dass wir trotz dem agilen Software-Entwicklungsvorgehen (Scrum) die Zeit für die Sprints zu knapp berechnet haben. Ansonsten zeigt die Übersicht ein relativ ausgeglichenes Ergebnis, lediglich das Software-Architektur-Dokument (Design) ging schneller voran, als von uns geplant.

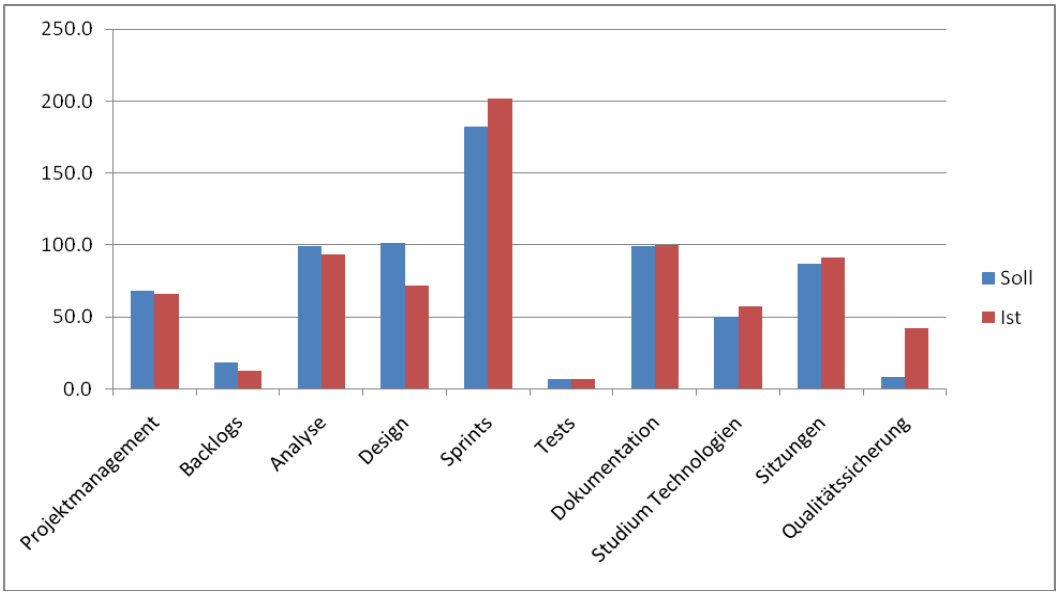


Abbildung 49 - Zeitauswertung Soll/Ist

Das untenstehende Diagramm (Abbildung 50) stellt den Ist-Zustand dar. Knapp ein Drittel der Arbeitszeit haben wir mit der Implementierung (Sprints) verbracht. Das Dokumentieren, sowie die Analyse- und Design-Dokumente, machen den nächsten Drittel aus. Die 1 Prozent für das Testen enthält lediglich die System-Tests, das Unit-Testing ist bereits in den Sprints enthalten.

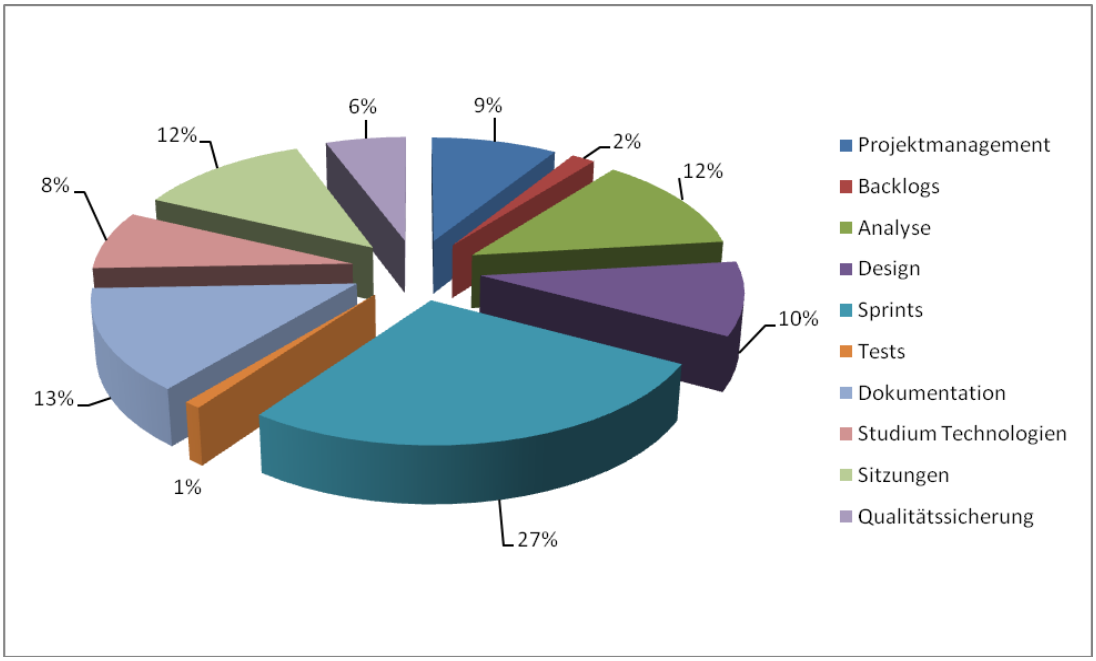


Abbildung 50 - Ist Zustand

12 Erfahrungsberichte

12.1 Matthias Good

12.1.1 Allgemein

Von vielen Seiten habe ich bereits vor dem Projektstart vernommen, dass Objective-C eine nicht einfach zu lernenden Programmiersprache ist. Dies habe ich dann auch rasch bei der Einarbeitung in die neue Technologie gemerkt. Vieles bleibt bei Objective-C vom Programmierer abhängig, d.h. man muss sich um viele Dinge kümmern, die einem andere Sprachen wie .NET oder Java abnehmen. Der Vorteil daran ist, dass man sich bewusst Gedanken macht, wie man genau vorgehen will bei der Implementierung.

12.1.2 Ablauf des Projektes

Als Sven Lenz uns das Thema "iPhone-App für E-Banking" vorschlug, war ich sofort begeistert von der Idee. Einerseits würde die Arbeit auch noch nach dem Abschluss als Studienarbeit weiterleben und andererseits hatte mich die Smart-Phone-Programmierung schon länger interessiert. Das Einarbeiten in die neue Technologie war wie bereits erwähnt ein harter Brocken, der sich jedoch gelohnt hat. Die "Product Planning"-Phase war sehr abwechslungsreich, einmal ein Paper-Prototyp erstellt, ein Interview mit Kunden, ein anderes Mal eine Vertiefung in OCR.
Die Implementierung ging dann meist wie geplant von statten Natürlich gab es hin und wieder kleine Probleme, welche aber meist in nützlicher Frist gelöst werden konnten. Die Muss-Anforderungen an die Arbeit konnten alle umgesetzt werden, für ein paar optionale Features reichte die Zeit leider nicht mehr.

12.1.3 Teamarbeit

Sven Lenz, Philipp Marugg und ich harmonierten sehr gut miteinander. Das lag unter anderem an unseren täglichen Meetings, bei denen jedes Teammitglied erfuhr, was die anderen im Moment arbeiteten. Das Arbeitsklima war stets sehr angenehm und wir pflegten einen respektvollen Umgang untereinander.

12.1.4 Fazit

Ich würde dieses Thema sofort noch ein weiteres Mal wählen. Das Projekt hat Spass gemacht, dies vor allem auch dank einem tollen Team. Mit meinem jetzigen Wissenstand wäre die Implementierung sehr wahrscheinlich ein wenig schneller vorangegangen, doch viel geändert hätte sich am Produkt nicht.
Objective-C ist eine spezielle, aber interessante Sprache und ich werde auch in Zukunft privat noch einige iPhone-Applikationen entwickeln.

12.2 Sven Lenz

12.2.1 Allgemein

Da der Projektpartner dieser Studienarbeit mein Arbeitgeber ist und ich somit auch während der Arbeit mit der finApp (Phase 1) beschäftigt bin, hat mich dieses Thema von Anfang an interessiert. Die Phase 1 steckt ebenfalls noch in den Kinderschuhen, und so stellte sich mir endlich einmal die Möglichkeit die Architektur eines Applikationservers zu designen und von Grund auf auf die Beine zu stellen.

12.2.2 Ablauf des Projektes

Da ich mittlerweile bereits sechs Jahre Berufserfahrung in der Entwicklung von Applikationserver habe, war für mich die Phase des "Product Plannings" sicherlich eine der grösseren Herausforderungen. Die Projektplanung und Anforderungsanalyse waren eine willkommene Abwechslung zum Design und der Implementation und ich habe dadurch viel Neues lernen können.
Der Ablauf des Projektes ist meiner Ansicht nach ohne grössere Probleme vonstatten gegangen und die Aufteilung in die einzelnen Phasen sowie Sprints liessen sich gut umsetzten. Gut fand ich auch, dass genug Zeit für ein Refactoring des eMob-Converter verblieben ist, wodurch die Qualität der Software verbessert werden konnte.

12.2.3 Teamarbeit

Die Zusammenarbeit mit Matthias und Philipp hat von Anfang an reibungslos funktioniert. Uns war bewusst, dass wir mit einer Dreierarbeit auch mehr Verwaltungsaufwand auf uns nehmen müssen. Da wir jedoch Scrum als Vorgehensmodell ausgewählt haben, konnten wir uns täglich gegenseitig auf den neusten Stand bringen und bei Problemen helfen.
Durch eine klare Aufgabenverteilung konnten wir schnell und effektiv unsere Studienarbeit umsetzten.

12.2.4 Fazit

Mit dem Resultat unserer Studienarbeit bin ich sehr zufrieden. Wir haben innert kürzester Zeit einen lauffähigen Prototyp entwickelt, der einem User eine einfache und sichere Art der Transaktionserfassung ermöglicht. Der eMob-Converter wird bereits in der Phase 1 der finApp benutzt, was bedeutet, dass der Prototyp bereits in absehbarer Zeit in die Produktion übernommen wird.

12.3Philipp Marugg

12.3.1 Allgemein

Die Softwareentwicklung auf mobilen Devices interessierte mich schon lange vor der Studienarbeit. Da Apps momentan sehr im Trend sind, wollte ich mir unbedingt ein wenig Know-How in diesem Bereich aneignen. Die Idee eine E-Banking App für das iPhone zu entwickeln nahm ich daher mit Begeisterung auf.

12.3.2 Ablauf des Projektes

Da ich keinerlei Erfahrung im Bereich iPhone OS und Objectiv-C hatte, habe ich mir schon vor Semesterbeginn drei Bücher gekauft, welche mir den Einstieg in die mir noch unbekannte Technologie erleichtern sollten.

Trotz der Lektüre war der Einstieg in die iPhone-Programmierung nicht einfach. Oft musste ich in Büchern oder im Internet nach Tipps und Lösungen suchen. Besonders mühsam war es die Kamerafunktionen zu implementieren. Da der iPhone-Simulator keine Kamerafunktionen bietet, musste ich bei jeder Codeänderung die App direkt auf mein iPhone installieren. Im Vergleich zum Simulator dauerte dies jedes Mal eine Ewigkeit.

Auch auf den ersten Blick kleinere Probleme kosteten mich, durch fehlende Erfahrung, teilweise enorm viel Zeit.

Da unsere App die Möglichkeit bieten sollte, einen Einzahlungsschein per Foto einlesen zu können. Musste ich mich auch mit OCR auseinandersetzen. Da ich überhaupt keine Kenntnisse in diesem Bereich hatte, war auch hier eine intensive Lektüre unumgänglich.

12.3.3 Teamarbeit

Unser Team hat stets gut zusammengearbeitet und wir haben uns immer gegenseitig unterstützt. Da wir von Anfang an die Arbeitspakte untereinander aufgeteilt haben, gab es nie grössere Konflikte.

12.3.4 Fazit

Die Arbeit mit dem iPhone Framework war für mich eine tolle Erfahrung und ich kann das neu erlernte Wissen auch Zukünftig wiederverwenden. Mit meinem jetzigen Wissensstand wäre es bestimmt einfacher gewesen, die Applikation zu entwickeln.

13 Glossar

Begriff	Definition
Finnova	finnova AG Bankeware, Auftraggeber
Finnova Core	Core Datenbank des Finnova System. Beinhaltet die Implementierung der Business Logik.
EDBS	"External Database System". Finnovas standard Java-Schnittstelle auf den Finnova Core.
IB	Internetbanking
finApp	Finnova iPhone-Applikation für das Internetbanking
UI	User Interface
User	Benutzer der finApp und des Internetbanking
Bank	Kunde der Finnova
OCR	Optische Zeichenerkennung
ZV	Zahlungsverkehr
eMobC	eMob-Converter. Tomcat-Server der Anfragen der finApp konvertiert und abarbeitet.
Credential	Sicherheitsmerkmal, das nur ein User kennt
DMZ	Demilitarized Zone. Nur kontrollierte Zugriffe auf die Server sind möglich.

Tabelle 26 - Glossar

14 Literaturverzeichnis

14.1 Kapitel 6 (Projektplan)

[01] Beschreibung der Methodik Scrum, http://de.wikipedia.org/wiki/Scrum, letzter Zugriff am 21.12.2010

[02] Beschreibung des Scrumablauf, http://www.scrum-kompakt.de/einfuehrung-in-scrum/scrum-prozess/, letzter Zugriff am 21.12.2010

[03] GHUnit, Test-Framework für das iPhone, https://github.com/gabriel/gh-unit, letzter Zugriff am 21.12.2010

14.2 Kapitel 7 (Analyse)

[04] Telefonate im Vorfeld der Studienarbeit mit Diego Stalder, finnova AG Bankeware, Merkurstrasse 6, 5600 Lenzburg

[05] Telefonat mit einem Kunden der Finnova. Aus Geheimhaltungsgründen dürfen die Kunden nicht namentlich erwähnt werden.

14.3 Kapitel 8 (Studien)

[06] Weiterführende Informationen zu dem Streichlisten-Verfahren, https://www.ebas.ch/index.php?option=com_content&view=article&id=134&Itemid=10&lang=de, letzter Zugriff am 22.10.2010

[07] Weiterführende Informationen zu dem Smarctard-Verfahren

- http://www.financesecurity.de/de/datacenter-security/authentisierung/emv-cap-dpa, letzter Zugriff am 22.10.2010
- http://atm-locator.six-card-solutions.com/site/download/flyer_ebanking_de.pdf, letzter Zugriff am 22.10.2010

[08] Weiterführende Informationen zu dem mTan-Verfahren, http://www.financesecurity.de/de/datacenter-security/authentisierung/mtan-verfahren, letzter Zugriff am 22.10.2010

[09] Phishing Attacken auf einem Mobilgerät, http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html, letzer Zugriff am 22.10.2010

[10] IBM e-Banking Security-Event, 04.10.2010

[11] ABBYY Mobile OCR Engine SDK,http://www.abbyy.com/mobileocr/,letzter Zugriff am 21.12.2010

[12] GOCR open-source character recognition, http://jocr.sourceforge.net/, letzter Zugriff am 21.12.2010

[13] GOCR open-source character recognition,http://code.google.com/p/tesseract-ocr/, letzter Zugriff am 21.12.2010

[14] NSXMLParser – Standardparser von Objective-C, http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSXMLParser_Class/Reference/Reference.html, letzter Zugriff am 21.12.2010

[15] Libxml2-Parser, C-Parser entwickelt für das Gnome Project http://xmlsoft.org/, letzter Zugriff am 21.12.2010

[16] TBXML-Parser, leichtgewichtiger XML-Parser für die Apple iPad, iPhone & iPod Touch Geräte, http://www.tbxml.co.uk/TBXML/TBXML_Free.html, letzter Zugriff am 21.12.2010

[17] TouchXML-Parser, leichtegewichtiger Ersatz für den NSXML-Parser, https://github.com/schwa/TouchXML, letzter Zugriff am 21.12.2010

[18] KissXML-Parser, leichtegewichtiger Ersatz für den NSXML-Parser mit Write-Funktion, http://code.google.com/p/kissxml/, letzter Zugriff am 21.12.2010

[19] TinyXML-Parser, simpler C++-Parser, welcher einfach in andere Programme zu implementieren ist, http://www.grinninglizard.com/tinyxml/, letzter Zugriff am 21.12.2010

[20] GData-Parser, XML-Parser aus der Google Data API, http://code.google.com/p/google-gdata/, letzter Zugriff am 21.12.2010

[21] XML-Performance, kleine iPhone-App um versch. Parser zu testen, http://developer.apple.com/library/ios/#samplecode/XMLPerformance/Introduction/Intro.html, letzter Zugriff am 21.12.2010

[22] Tool um XML-Dokumente zu erzeugen, http://code.google.com/p/xswi/, letzter Zugriff am 29.11.2010

14.4 Kapitel 9 (Software Architektur Design)

[23] Informationen zur Web-Application-Firewall Airlock der Firma Ergon Informatik AG, http://www.ergon.ch/de/security/, letzter Zugriff am 12.10.2010

[24] Markus Stäuble, Programmierung fürs iPhone, ISBN 978-3-89864-635-2

[25] Informationen zum Apache Tomcat, http://tomcat.apache.org/, letzter Zugriff am 12.10.2010

[26] Informationen zu log4j, http://logging.apache.org/log4j/, letzter Zugriff am 14.10.2010

[27] Markus Schumacher, Security Patterns, ISBN 0-470-85884-2, Single Access Point, pp. 267-286

[28] Markus Schumacher, Security Patterns, ISBN 0-470-85884-2, Check Point, pp. 287-296

15 Anhangsverzeichnis

Die folgenden Dokumente befinden sich im Anhang

Referenz	Titel	Dokumentname
[ANH00]	Abstract	00_Abstract.doc
[ANH01]	Poster	01_Poster.ppt
[ANH02]	Projektplan	02_Projektplan.xlsx
[ANH03]	Protokolle und Traktanden	03_Protokolle_Traktanden.doc
[ANH04]	Product Backlog	04_productBacklog.xlsx
[ANH05]	Sprint Backlog 1	05_sprintBacklog01.xlsx
[ANH06]	Sprint Backlog 2	06_sprintBacklog02.xlsx
[ANH07]	Sprint Backlog 3	07_sprintBacklog03.xlsx
[ANH08]	Klassendiagramm eMobC	08_eMobC.png
[ANH09]	Code eMobC	09_eMobC.zip
[ANH10]	Code finApp	10_finApp.zip
[ANH11]	Muster der Geheimhaltungserklärung	11_NDA.doc

Tabelle 27 - Anhangsverzeichnis

16 Tabellenverzeichnis

17 Abbildungsverzeichnis