



Die „KiBa“-App

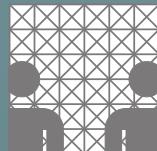
Projektbericht

von

Alexander Droste
Markus Fasselt
Marco F. Jendryczko
Konstantin S. M. Möllers
Michael Schaarschmidt
Julius Wulk

Veranstalter

Dr. Guido Gryczan, Dr. Martin Christof Kindsmüller und Christian Zoller



Universität Hamburg

Fachbereich Informatik

Inhaltsverzeichnis

1	Einleitung und Projektkontext	4
2	App-Idee	5
2.1	Problematik	5
2.2	Vision	6
2.3	Name	7
2.4	Gerät	7
2.5	Funktionen	8
2.5.1	Finder	8
2.5.2	Authentifizierung	8
2.5.3	Self-Service	9
2.5.4	Individueller Finanzierungsrechner	10
2.5.5	Weitere Funktionen	10
3	Kontexterkundung	11
3.1	Fragebogen	11
3.2	Ergebnisse	11
4	User-Interface & User-Experience	14
4.1	Mockups	14
4.2	Prototyp	14
4.3	Logo & Farbharmonie	16
4.4	UI-Elemente	17
4.5	UX Inspektion & Test	17
5	Vorgehen	19
5.1	Scrum	19
5.2	Von Alex	20
6	Ergebnis	26
6.1	Features	26
6.1.1	Dashboard	26
6.1.2	Filialfinder	26
6.1.3	Sortenanfrage	26

6.1.4	Authentifizierung	27
6.1.5	Überweisung	28
6.1.6	Self-Service Station	28
6.1.7	Bescheinigungen	29
6.1.8	Kontoauszüge	29
6.1.9	Umbuchungen	30
6.1.10	Finanzierungsrechner	30
6.1.11	Mein Bereich	31
6.2	Die Design-Highlights	32
6.3	Herausforderungen	32
7	Architektur	34
7.1	Umsetzung des MVC-Ansatzes	34
7.2	Datenmodell	35
7.3	Sicherheitsaspekte	36
7.4	Dependency Injection	36
8	Toolchain	39
8.1	Agiles Vorgehen im Projekt	39
8.2	Versionsverwaltung und Informationsaustausch	40
8.3	Konzept und erster Entwurf	42
8.4	Designtools	42
9	Qualitätssicherung	43
9.1	Entwicklungsprozess	43
9.2	Qualität des Designs	44
9.3	Umgang mit Feedback	44
9.4	Ausblick	45
10	Fazit	46
Abkürzungsverzeichnis		48
Abbildungsverzeichnis		48
Programmausdrucke		49

1 Einleitung und Projektkontext von Michael Schaarschmidt

Der vorliegende Bericht soll die rückblickende Betrachtung eines universitären Software-Projekts konstituieren, bei dem über ein Semester hinweg in Kooperation mit der T-Systems MMS eine iOS-Anwendung entwickelt wurde. Die zentrale Herausforderung war dabei, universitäre Erfordernisse mit kommerziellen Anforderungen in Einklang zu bringen. Ziel war es, anhand einer gegebenen Aufgabenstellung in Gruppen mit universitär heterogenem Hintergrund einen Prototypen zu entwickeln. Dieser sollte in Architektur und Gestaltung das Verständnis softwaretechnischer wie interaktiver Konzepte deutlich machen, gleichzeitig aber geeignet sein, in einer konkreten Produktdemonstration im Projektabschluss kommerziellen Nutzen zu demonstrieren. Zusätzlich wurden wir im Rahmen einer in das Projekt integrierten, einführenden Zertifizierung mit der Scrum-Methodik vertraut gemacht, welche dann als Grundlage für die Projektarbeit diente.

Die unser Gruppe übertragene Aufgabe bestand darin, eine App zu entwickeln, welche die Bindung zwischen einer fiktiven Filialbank und ihren Kunden erhöht.

In den ersten Überlegungen wurde deutlich, dass nahezu jede Dienstleistung einer Filiale auf die ein oder andere Weise von Direktbanken abgebildet werden kann. Nicht umsonst ist das Filialgeschäft im Abschwung¹. Es erscheint insofern zunächst widersinnig, ein Geschäftsmodell, welches zunehmend effizient durch Softwarereprodukte ersetzt wird und nur noch wenig profitabel ist, um jeden Preis erhalten zu wollen. Während nämlich auch im Bankgeschäft parallel zu sozialen Netzwerken Social-Banking-Konzepte erste Anwendung finden², haben diese nicht unmittelbar Bezug zum Filialgeschäft.

Hier wurde uns bewusst, dass als Entwickler in Auftragsarbeit die eigenen Überlegungen hinsichtlich der Entwicklung eines Geschäftsfeldes oder Produktes zurückstehen müssen; viel mehr müssen innerhalb eines schwierigen Marktes Möglichkeiten gefunden werden, innerhalb der vorhandenen Strukturen zu innovieren.

¹<http://www.welt.de/print/wams/wirtschaft/article124228415/Das-ist-ein-Sterben-auf-Raten.html>

²<http://www.visiblebanking.com/commbank-launches-first-social-banking-app-facebook-p2p-payments-7693/>

2 App-Idee von Michael Schaarschmidt

2.1 Problematik

Mit Rücksicht auf oben genannte Aspekte haben wir uns in der Ideenfindungsphase auf Lokalität als entscheidendes Schlagwort konzentriert. Apps wie etwa foursquare³, welche Bezug auf lokale Unternehmen nehmen und Benutzern die Möglichkeit geben, sich darüber auszutauschen, sind beispielhaft für die erfolgreiche Umsetzung dieses Konzepts. Für das Bankgeschäft ergab sich jedoch das Problem, dass die eigenen finanziellen Verhältnisse in den innersten persönlichen Lebensbereich fallen und in der Regel darüber zumindest weniger Austausch stattfindet als über Restaurants, Nachtclubs oder Freizeitaktivitäten. So erschien es uns nachvollziehbar, dass die App eine Interaktion zwischen einem Kunden und einer Filiale begünstigen soll, nicht aber den Austausch zwischen mehreren Kunden - der dann ohnehin wieder kaum eine Filiale benötigen würde.

Erste Überlegungen zielten in Richtung interaktiver Beratungsdienstleistungen, welche jederzeit über Videotelefonie Kontakt zum individuell vertrauten Berater einer bestimmten Filiale herstellen sollten. In den wöchentlichen Diskussionen im Projektplenum wurden uns aber Mängel dieses Konzepts aufgezeigt; der individuelle Berater ist kein Alleinstellungsmerkmal der Filialbank mehr, eine virtuelle Filiale ist ohne weiteres denkbar.

Allerdings, und hier haben wir die Chance unserer Anwendung gesehen, bedeutet die technische Möglichkeit eines Vertriebsweges nicht gleich, dass dieser auch angenommen wird. Eine von uns im Sinne einer Markterkundung durchgeführte Umfrage ergab ein gewisses Misstrauen gegenüber der Idee, Finanztransaktionen per App durchzuführen oder überhaupt eine Bank-App zu benutzen. Gründe hierfür könnten vielfältig sein, beispielsweise könnten die befragten Studenten am Informatikcampus eine besondere Sensibilität bezüglich Aspekten des Datenschutzes und der Transaktionssicherheit haben. Eine von der Unternehmensberatung Bain and Company durchgeführte Studie zur Frage, wie digitalisierte Vertriebswege das Privatkundengeschäft der

³<https://de.foursquare.com/>

Banken zukünftigen prägen könnten, ergab jedoch einen ähnlichen Tenor⁴.

Insbesondere ergab die Studie, dass für Geldanlagen dem persönlichen Gespräch der Vorzug vor digitaler Beratung gegeben würde. Inwiefern eine solche Untersuchung aussagekräftig für eine Banking-App der Zukunft ist, sei dahingestellt. Schließlich wäre es auch möglich, dass hier einfach noch keine Gewöhnung stattgefunden hat und die Studie nur einen Übergangseffekt reflektiert, der für eine langfristig strategische Ausrichtung irrelevant wäre.

Bemerkenswert ist aber vielleicht auch, dass der Mensch in Finanzfragen nicht immer von Vernunft geleitet ist, sondern zunehmend auch nicht-funktionale Anforderungen in den Vordergrund stellt, wie etwa das Bedürfnis nach nachhaltigen Investitionen. Dies mag auch für die Art gelten, wie Finanzberatung vermittelt wird. Somit könnte es trotz aller technischen Möglichkeiten auch langfristig noch einen Markt für persönliche Beratung von Angesicht zu Angesicht geben.

2.2 *Vision*

Schließlich entstand die Vision einer Anwendung, die nicht versucht, aus einer Filialbank eine Direktbank mit Filialen als Zusatzangebot zu machen, sondern die bestehenden Produkte des Filialgeschäfts dem Kunden näher bringt und ihm Grund gibt, in die Filiale zu gehen. Dabei muss mit Blick auf die Stakeholder Kunde und Bank ein Mehrwert geschaffen werden, der es für den Kunden rechtfertigt, eventuell höhere Gebühren als bei einer Direktbank zu bezahlen.

Als Kernprodukte und Kompetenzen haben wir das klassische Sparbuch, die Anlageberatung sowie die Finanzierung ausgemacht. Das Sparbuch bietet scheinbar keine Funktionalität, die einen konkreten Vorteil gegenüber beispielsweise dem Tagesgeldkonto einer Direktbank hätte. Wie aber einführend erläutert, ist dies nicht ausschlaggebend. Denn ungeachtet seiner Nachteile ist das Sparbuch populär; vielleicht auch, weil in den Verwerfungen der europäischen Finanzkrise hier besondere Sicherheit vermutet wird. Tatsächlich ist aber ein Tagesgeldkonto bei einer Direktbank ebenso über die Einlagensicherung geschützt wie ein Sparbuch⁵.

⁴http://www.bain.de/Images/Retail_Banking_II_Digitalisierung_ES.pdf

⁵vgl. Einlagensicherungs- und Anlegerentschädigungsgesetz

Diesem Sicherheitsbedürfnis der Kunden wollten wir dabei konsequent nicht nur in der kryptographischen Theorie, sondern auch mit Rücksicht auf die vom Kunden wahrgenommene Sicherheit gerecht werden - etwa durch in der App prominente visuelle Elemente, die mit Sicherheit assoziiert werden.

Filialbanken bieten heute sicher deutlich mehr an, als eben beschrieben. So kann man über ein bestimmtes Konto der Hamburger Sparkasse vergünstigte Reisen im eigenen Reise-Shop erwerben⁶; allerdings kann diese Expansion in andere Geschäftsfelder von der Fahrradversicherung bis zum Telefon-Rechtsschutz als Ausdruck wachsender Verzweiflung am schwindenden Kerngeschäft interpretiert werden. Diese als Webservice umgesetzten Zusatzangebote bilden aber wiederum keine Alleinstellungsmerkmale und sind daher von uns nicht integriert worden.

In den nächsten Abschnitten wird nun erläutert werden, welche Funktionen wir uns in Konsequenz überlegt haben und mit welchen Use-Cases diese korrespondieren.

2.3 Name

Der Name der App, „KiBa“, steht in Anlehnung an eine bekannte Direktbank für eine kundeninteressierte Bank, die mit der Anwendung mehr über ihre Kunden erfahren möchte und mit ihnen in Austausch treten will, um spezifischere Angebote und Beratungen anbieten zu können.

2.4 Gerät

Bei der Erstellung eines konkreten Konzepts der Funktionalität haben wir zunächst die Anwendungsfälle besprochen, um die Gerätfrage zu klären. Zweifelsohne gibt es Features, die in einem mobilen Anwendungsfall wahrscheinlicher sind, etwa das Auffinden einer Filiale oder eines Geldautomaten. Eine solche Funktionalität wäre aber nicht filialbankspezifisch. Überhaupt erschien es uns, als wäre der typische Anwendungsfall eher stationär, auf dem Sofa, im Büro, jedenfalls aber in Ruhe. Ein überzeugendes Argument für eine iPad-App ist auch, dass Kunde und Berater in der Filiale zusammen mit der App interagieren und Dinge visualisieren können.

⁶<https://www.haspajoker.de/index/Service/Reise-Service.html>

Somit überwogen in der Gruppe ganz eindeutig die Argumente für eine iPad-Anwendung. Zudem war es Aufgabe, eine Vision für die Banking-App der Zukunft zu entwickeln. Der Trend geht unserer Meinung nach zum ubiquitären W-Lan; so gibt es beispielsweise Bestrebungen, öffentliche Netzwerke in der Innenstadt einzurichten. Insofern darf davon ausgegangen werden, dass die mobile Konnektivität zukünftig auch beim iPad zukünftig unproblematisch ist und somit webbasierte Funktionalität auch unterwegs gegeben ist.

2.5 Funktionen

2.5.1 Finder

Ein intuitiv klarer Anwendungsfall ist der Wunsch eines Nutzers, die nächstliegende Filiale aus der App heraus auf einer Karte zu finden und dorthin navigieren zu können. Um diese Funktionalität noch gegenüber einem bekannten Kartendienst abzuheben, soll schon an dieser Stelle zusätzliche Interaktion mit einer Filiale möglich sein. Über eine Sortenanfrage können Devisen bestellt werden und aus der Filialseite, die aus der Karte geöffnet wird, kann ein Termin vereinbart werden.

2.5.2 Authentifizierung

Der Authentifizierungsmechanismus trennt die Funktionen in sicherheitsrelevante und unkritische. Ohne Authentifizierung steht dem Benutzer nur passive Funktionalität zur Verfügung, also etwa der Filialfinder und die Umsatzanzeige. Um die App voll nutzen zu können, muss der Benutzer in eine KiBa-Filiale und von einem Berater einen Sicherheitscode eingeben lassen. Dieser garantiert dann in Kombination mit der App-ID der Anwendung eine eindeutige Zuordnung eines Benutzers an sein Gerät. Ähnlich einer Kreditkarte soll dann bei Gerätverlust auch die App selbst jederzeit gesperrt werden können. Insbesondere dient die Aktivierung aber auch dazu, den Kunden in einem Beratungsgespräch besser kennenzulernen und in einer Datenbank einen persönlichen Ansprechpartner festzuhalten.

2.5.3 Self-Service

Im Zuge der Plenumsdiskussionen und anschließenden internen Debatten haben wir den zeitlichen Aspekt als zentrale Komponente identifiziert. Lokalität bedeutet, Dinge direkt zur Verfügung gestellt bekommen zu können. Viele Bescheinigungen und Unterlagen im täglichen Leben werden auch heute noch konkret ausgedruckt benötigt. Der typische Ablauf einer Direktbank sieht so aus, eine Anfrage - etwa nach Wertpapierzweitschriften oder Bonität - per Kontaktformular abzuschicken und dann einige Tage auf die entsprechenden Ausdrucke zu warten. Unsere Idee besteht in einer Self-Service Station innerhalb der Bank, die das Konzept bestehender Automaten erweitert. Ein Kunde kann sein Ipad auf eine Ablage legen und sich mit der Station verbinden. Die Station beinhaltet dann einen Multifunktionsdrucker und per App können verschiedenste Bescheinigungen ausgedruckt und Konten eröffnet werden. Das entscheidende dabei ist, dass der Kunde durch die Authentifizierung in einer Filiale seinem Gerät zugeordnet wurde. Somit können an derartigen Stationen auch Interaktionen vorgenommen werden, die normalerweise die Identifikation per Lichtbildausweis am Schalter erfordern würden. Somit entsteht hier Mehrwert für alle Stakeholder: Kunden müssen weniger anstehen für standardisierte Abläufe, die Bank spart unter Umständen Personalkosten und kann Mitarbeiter für das wesentliche, die Beratung, einsetzen.

Umbuchungen am Sparbuch können üblicherweise nur in einer Filiale vorgenommen werden, überwiesen werden kann nur auf das Sparkonto. Die Greifbarkeit des Sparbuchs vermittelt konservativen, besorgten Sparern ein Gefühl von Sicherheit. Gleichzeitig kann es aber durch diese funktionale Einschränkung auch zu unerwünschten Situationen kommen: ist etwa durch eine Fehlkalkulation an einem Samstagabend kein Geld mehr auf dem Girokonto, muss bis zur Öffnung einer Filiale am Montag gewartet werden, um Guthaben umbuchen zu können. Um dem Vorzubeugen, soll über die Self-Service Station auch Geld umgebucht werden können. Da die Station im Vorraum der Filiale steht, ist sie immer zugänglich.

2.5.4 Individueller Finanzierungsrechner

Wie oben erläutert, besteht von Filialbanken besteht in der Finanzierung, etwa für Eigenheime. Im Zentrum unserer Überlegungen stand dann auch die Frage, wie eine App dabei helfen kann, diese Dienstleistung für den Endkunden zu verbessern. Im Ergebnis möchten wir einen individualisierten Kreditrechner anbieten, der den App-Benutzer in die Lage versetzt, mittels individuell berechneter Profildaten für sich selbst Finanzierungsrechnungen durchzuführen. Die Idee dahinter ist, dass ein Kunde zunächst für sich selbst einige Finanzierungsvarianten durchspielen kann. Hat er sich eine Variante überlegt, kann ein Termin mit dem persönlichen Berater vereinbart werden und dabei optional gleich der Finanzierungsvorschlag exportiert werden. Im Filialgespräch kann der Berater dann noch individuelle Ratschläge bezüglich Laufzeit und Umfang einer Finanzierung geben. Wichtig dabei ist, dass die angebotenen Finanzierungsdaten mit Rücksicht auf die der Bank zur Verfügung stehenden Informationen so konservativ gewählt sind, dass jedes berechnete Angebot auch vom Kunden angenommen werden kann.

2.5.5 Weitere Funktionen

Als Startbildschirm nach dem Login ist für die App ein Dashboard vorgesehen, das einen grafischen Überblick über Vermögensverlauf und Transaktionen bereitstellt. Hilfreich war hier die Überlegung, dass die meisten Benutzer ihren Kontostand grob kennen und weniger an einer Zahl als vielmehr an den Entwicklungen interessiert sind. Zudem gibt es einen Nachrichtenbereich, in dem die Antworten auf beispielsweise Terminanfragen abgelegt werden. Einfache Überweisungen können ebenfalls getätigt werden, schlicht, weil dies unserer Ansicht nach von einer Bankanwendung erwartet wird.

3 Kontexterkundung von Julius Wulk

Im Rahmen der oben beschriebenen Überlegungen wurde bereits angesprochen, dass wir eine Markterkundung durchgeführt haben. Deren Umsetzung soll hier noch einmal im Detail erläutert werden, um daraus Spezifizierungen für die Mockups zu identifizieren.

3.1 Fragebogen

Bei der Datenerhebung haben wir uns, wie in Abbildung 1 zu sehen, für eine Umfrage mittels Fragebogen entschieden, da die App in vielen verschiedenen Kontexten eingesetzt werden soll und wir möglichst viele Meinungen einfließen lassen wollten.

Für unser Konzept wollten wir wissen, welche Funktionen besonders beliebt sind und aus welchen Gründen die Teilnehmer Befürchtungen bei der Nutzung von Banking-Apps haben. Außerdem war es uns ein Anliegen, nicht nur quantitative, sondern auch qualitative Daten zu erheben. Daher gab es Freitextfelder, in denen die Teilnehmer Ideen, Kritik und allgemeine Bedenken in Bezug auf Banking-Apps äußern konnten. Dieses half uns, die potentiellen Nutzer besser zu verstehen und Dinge aufzudecken, an die wir nicht gedacht hätten. An der Umfrage am Informatikum haben 92 Studenten teilgenommen. Hier folgen nun die wichtigsten Ergebnisse.

3.2 Ergebnisse

Die drei nützlichsten Features sind der Bankautomatenfinder, Filialenfinder und die Kartensperre. Der Bankautmatenfinder wurde von 93% der Befragten als nützlich bewertet. Das sehr ähnliche Feature Filialenfinder fanden 81% nützlich und die Kartensperre wurde von 61% als nützlich angesehen. Diese Ergebnisse zeigen, dass wir die Kernfeatures einer Banking-App nicht aus den Augen verlieren dürfen und dass ein optimierter Finder Potential hat.

48% der befragten Studenten hielten die Features „Depot-Verwaltung für Aktien“ und „Personalisierte Angebote der Bank“ für nicht nützlich. Eine mögliche Erklärung dafür ist, dass die Studenten bei diesen Themen einfach noch keine eigene Relevanz sehen.

Ein überraschendes Ergebnis der Umfrage war, dass nur 17% der befragten Studenten eine

Umfragebogen zu einer Banking-App					
1. Welchen Studiengang belegst du?					
Informatik <input type="checkbox"/> Wirtschaftsinformatik <input type="checkbox"/> Mensch-Computer-Interaktion <input type="checkbox"/> Andere, und zwar: _____					
2. Findest du folgende Funktionen für eine Banking-App nützlich?					
	sehr nützlich	nicht nützlich			
A	<input type="checkbox"/>				
B	<input type="checkbox"/>				
C	<input type="checkbox"/>				
D	<input type="checkbox"/>				
E	<input type="checkbox"/>				
F	<input type="checkbox"/>				
G	<input type="checkbox"/>				
H	<input type="checkbox"/>				
I	<input type="checkbox"/>				
J	<input type="checkbox"/>				
K	<input type="checkbox"/>				
L	<input type="checkbox"/>				
M	<input type="checkbox"/>				
3. Welche weiteren Features findest du nützlich? (Bitte angeben)					
_____ _____ _____					
4. Wie oft kontrollierst du deinen Kontostand?					
	stetig	<input type="checkbox"/>			
	2-3 Mal die Woche	<input type="checkbox"/>			
	1 Mal die Woche	<input type="checkbox"/>			
	alle 2 Wochen	<input type="checkbox"/>			
	sel tener	<input type="checkbox"/>			
Umfragebogen zu einer Banking-App					
5. Benutzt du derzeit eine Banking-App?					
Ja <input type="checkbox"/> Nein, und zwar aus folgendem Grund: _____					
5A. Wenn ja, was gefällt dir an der App?					
_____ _____					
5B. ... und was gefällt dir nicht?					
_____ _____					
6. Hast du Bedenken bezüglich folgender Punkte bei der Bedienung einer mobilen App?					
	keine Bedenken	große Bedenken			
A	<input type="checkbox"/>				
B	<input type="checkbox"/>				
C	<input type="checkbox"/>				
D	<input type="checkbox"/>				
E	<input type="checkbox"/>				
F	<input type="checkbox"/>				
G	<input type="checkbox"/>				
6H. Sieht du etwas besonders kritisch?					
_____ _____					
7. Das wollte ich noch hervorheben ...					
_____ _____					

Herzlichen Dank für deine Unterstützung!

Abbildung 1: Endfassung des Umfragebogens

Banking App nutzen. Für uns ist es natürlich interessant, aus welchen Gründen Banking-Apps gemieden werden. Studenten gaben an, dass „Apps zu unsicher“ wären, „Apple, Microsoft und Co. Zugriff auf meine Daten“ nähmen, „Unsicherheit der Datenübertragung“, „Bankgeschäfte sollte man in Ruhe bearbeiten und nicht hektisch to go“.

In allen abgefragten Kategorien hatte mindestens über die Hälfte der Studenten Bedenken. Fast 80% der befragten Studenten gaben an, Angst vor Missbrauch der Daten zu haben. Das zeigt, dass das Thema Sicherheit der eigenen Daten beim Banking eine Schlüsselrolle spielt und wir beim Konzept diesem Aspekt eine besondere Bedeutung zukommen lassen sollten.

Abschließend wollen wir die Ergebnisse in Form eines Diagramms (Abbildung 2) präsentieren.

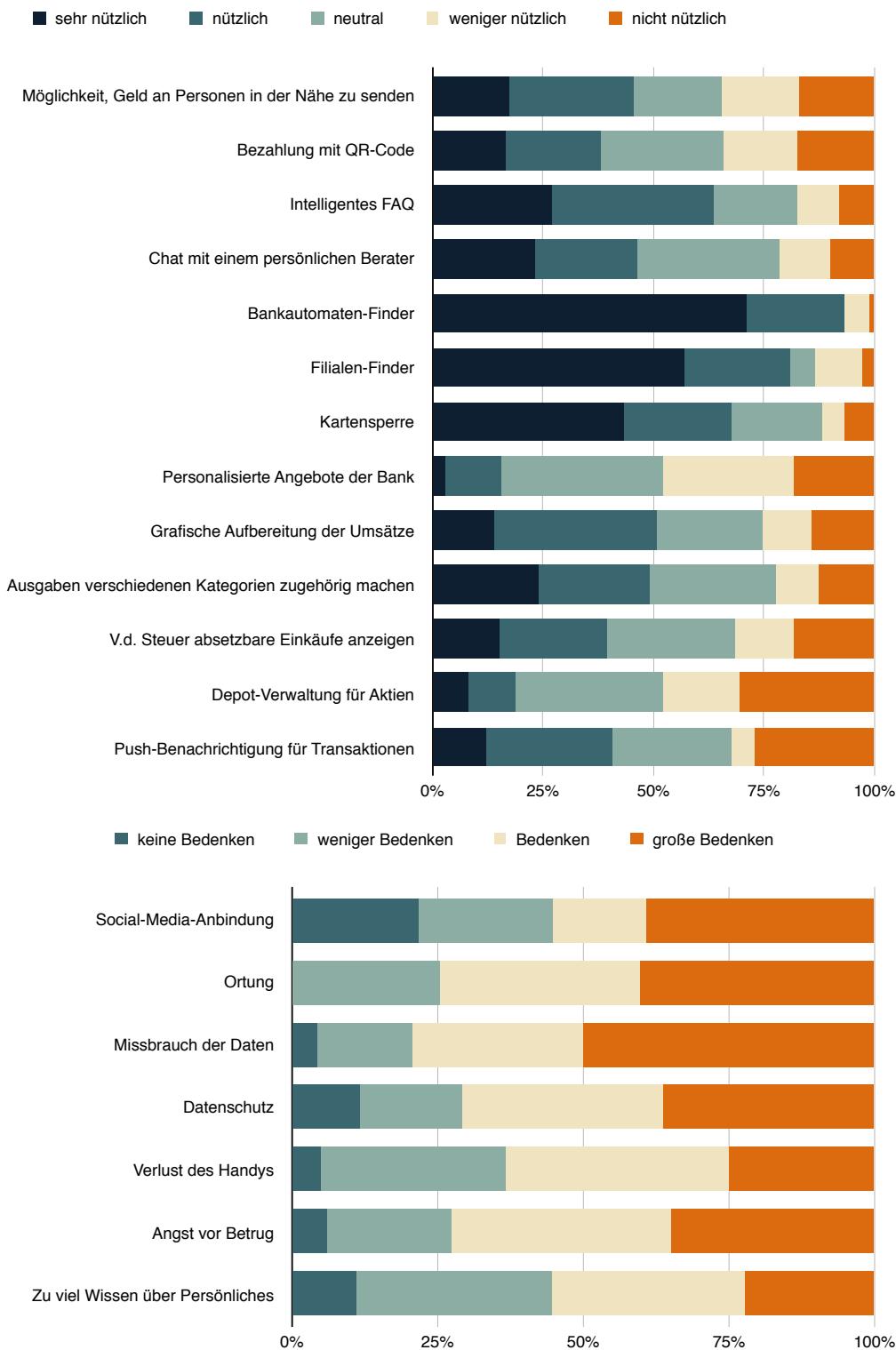


Abbildung 2: Ergebnisse der Kontexterkundung

4 User-Interface & User-Experience von Julius Wulk

4.1 Mockups

Vor Beginn der Gestaltung der Mockups haben wir uns auf dem Markt der Banking-Apps umgeschaut, um die verschiedenen UI-Patterns, die in diesem Zusammenhang genutzt werden, zu vergleichen. Mit Balsamiq entwarfen wir anschließend erste Mockups. Bei der Anordnung der Elemente haben wir uns die Prinzipien der Gestaltungsgesetze zunutze gemacht. Insbesondere halfen uns das Gesetz der Ähnlichkeit und das Gesetz der Nähe, Ordnung in die UI zu bringen.

Unser Ziel war es, ein möglichst gebrauchstaugliches Design zu erreichen. Dazu haben wir uns an den Grundsätzen der Dialoggestaltung nach der DIN EN ISO 9241 110 orientiert. Bei der Gestaltung haben wir nur eine Schriftart verwendet und uns nach festgelegte Farbsemantiken gerichtet. Dadurch geht der Fokus auf den Inhalt nicht verloren und der Nutzer kann die App erwartungskonform nutzen. Bei der Navigation haben wir uns für eine Master-Detail-View entschieden. Dadurch haben wir eine flache Informationsstruktur, die es uns ermöglicht, alle wichtigen Funktionen von der Hauptseite direkt zu erreichen. Auf diese Weise haben wir unnötige Interaktionen minimiert und der Nutzer kann effizient und aufgabenangemessen mit der App interagieren.

Auf jeder Detail-View befindet sich unten rechts ein Informationbutton. Mit diesem Button wollen wir die Selbstbeschreibungsfähigkeit der App verbessern und die Nutzer bei Problemen unterstützen.

Die Elemente in der Detail-View sind asymmetrisch angeordnet mit einer Gewichtung oben links und unten rechts. So erreichen wir eine diagonale Balance, die optimal für den optischen Fluss ist.

4.2 Prototyp

Allgemein haben wir uns beim Design des User-Interface an den Guidelines von IOS7 orientiert. In diesem Zusammenhang haben wir uns mit den wichtigsten Grundsätzen laut Apple „Defence“, „Clarity“ und „Depth“ beschäftigt, um diese auf unser User-Interface zu übertragen.



Abbildung 3: Mockup des Dashboards, der Filial-Seite und der Überweisung

„Deference“ haben wir bewirkt, indem wir den Fokus auf die Wahrnehmung von Inhalten gelegt haben und versucht haben, UI-Elemente, die in Konflikt mit dem Inhalt stehen, wegzulassen. Uns war es wichtig, das User-Interface möglichst funktional zu gestalten, was auch im Sinne von „Clarity“ ist. Daher haben wir eine möglichst selbsterklärende Ikonographie gewählt und auf eine unnötige Farbenvielfalt verzichtet. Den Guidelines entsprechend haben wir die Icons mindestens in der Größe 44×44 pts dimensioniert. Andernfalls wird es schwierig, ein Icon mit dem Finger zu treffen. Außerdem gibt es zu jedem Icon 2 Versionen, damit auf einem Retina iPad die hohe Auflösung ausgenutzt werden kann. Mit iOS7 setzt Apple auf Flat-Design. Ganz im Sinne des Flat-Designs haben wir eine dünne seriflose Typography ausgewählt und mit kräftig Kontrasten gearbeitet statt Schlagschatten oder Gradienten.

Aufgrund von leichten Konzeptänderungen während des Projekts haben wir manche Details im Prototypen anders als in den Mockups gestaltet.

Beispielsweise gab uns Markus Foos den Tipp, dass Nutzer mehr interessiert sind an den Umsätzen als an dem aktuellen Kontostand. Das ist besonders der Fall bei denjenigen Leuten, deren Kontostand niedrig ist oder die sogar im Minus liegen..



Abbildung 4: App-Icon mit Farbskala

4.3 Logo & Farbharmonie

Unsere ausgewählte Farbharmonie spiegelt die Werte einer modernen Bank wider. Mit den Farben wollen wir Sicherheit, Nachhaltigkeit und Seriosität vermitteln. Allerdings waren wir bei der Logogestaltung in dieser Hinsicht nicht ganz konsequent. Wir hatten an dieser Stelle die Diskussion, ob wir uns nicht für ein anderes Logo entscheiden sollten. Mit dem KiBa-Glas im Logo haben wir uns dann darauf geeinigt, als Studenten ein wenig Humor zuzulassen. Im Zusammenspiel mit dem recht nüchternen Design wirkt das Logo auflockernd.

4.4 UI-Elemente

Grundsätzlich haben wir mit den Standard-View-Elementen von iOS7 gearbeitet. Allerdings haben wir in manchen Fällen auf ein Custom-Element zurückgegriffen und dieses dann konsequent eingesetzt. Bei dem Standard-Text-Field störte uns etwa, dass der Hint-Text bei der Eingabe verschwand und so die Eingabe gelöscht werden musste, um diesen wieder zu sehen. Man stelle sich vor, man möchte ein Überweisungsformular ausfüllen und während der Eingabe eines Feldes ist man sich nicht mehr sicher, wofür dieses steht.

Daher entschieden wir uns für eine Eigenlösung, die den Vorteil hat, dass der Hint-Text bei erfolgter Eingabe nach oben rückt und auf diese Weise sichtbar bleibt.

4.5 UX Inspektion & Test

Wöchentlich hatten wir im Plenum die Möglichkeit, unseren Stand der App vorzustellen. Hier konnten in einer anschließenden Diskussion Unklarheiten im Konzept und Design angesprochen werden. Da sich im Plenum unter Anderem Dr. Kindsmüller befand, konnten wir von dem Wissen eines erfahrenen UX-Experten profitieren. Beispielsweise deckte er auf, dass wir eine Geste nicht erwartungskonform verwendeten. Bei der Scheck-Überweisung wollten wir anfangs mit der Wischgeste nach unten die Überweisung abschicken. Allerdings wird diese Wischgeste normalerweise als „Pull-to-refresh“-Geste verwendet. Daher war hier die Affordanz (Angebotscharakter) nicht gegeben. Um Verwirrung beim potentiellen Nutzer zu vermeiden, haben wir die Geste durch einen einfachen Button zum Abschicken der Überweisung ersetzt und die Animation entsprechend angepasst.

Zwischendurch haben wir auch Kommilitonen die KiBa-App testen lassen. Wir stellten bei diesen kleinen Usability-Tests fest, dass das Konzept und der Sinn hinter der Authentifizierung nicht ganz verständlich war. Daher entschieden wir uns einen Comic zu erstellen, der die Kernschritte und Vorteile explizit erklärt. Auch ging aus der App nicht hervor, bei welchen Funktionen die Authentifizierung eine Rolle spielt. Diesen Mangel behoben wir, indem wir die Funktionen, welche erst nach der Authentifizierung verwendbar sind, mit einem Schloss-Icon markierten.

Im Schlussprint hatten wir die Möglichkeit, mit Herrn Kindsmüller jede View zu inspirieren,

um auch die kleinen Interaktionsmängel aufzudecken. Beim Comic, der das Authentifizierungsverfahren erklärt, war beispielsweise der „Vorteil genießen“- Button für Nutzer nicht auffordernd genug. Dieses Problem der Salienz lösten wir durch eine leichte Animation. Des Weiteren wurde bei dem Kreditrechner nicht klar, dass individuelle Konditionen beim Berechnen eine Rolle spielen. Dr. Kindsmüller schlug uns vor, persönliche Assoziationen zu verwenden, wie „Ihr Kreditrechner“ oder „Johns Kreditrechner“.

All diese Erfahrungen zeigten uns, wie wichtig es ist, Kritik von außen einzuholen. Da wir in der Gruppe viel Vorwissen hatten, erschienen uns Dinge eindeutig, die für spätere Nutzer nicht unbedingt eindeutig sind. Sicherlich wäre besser gewesen, früher mit dem Testen anzufangen.

Dadurch das wir mit High-Fidelity-Prototypen getestet haben, gab es hauptsächlich Feedback zu kleineren Details und das gesamte Konzept wurde nicht in Frage gestellt. Trotzdem haben wir viel hilfreiche Kritik bekommen und konnten auf diese Weise mögliche Abbruchstellen in der UI beseitigen.

5 Vorgehen von Alexander Droste

5.1 Scrum

Im Rahmen des Projekts wurden wir mit der Scrum-Methodik vertraut gemacht und hatten Gelegenheit, eine einführende Zertifizierung zu absolvieren. Während der ersten Hälfte des Semesters standen Vision, Konzeption und Lernprozesse im Vordergrund. Mit Beginn der eigentlichen Implementierung in der zweiten Semesterhälfte bestand dann die Herausforderung darin, den Scrum-Prozess auf die zeitlichen Gegebenheiten einer Gruppe von Studenten mit unterschiedlichen Stundenplänen abzustimmen. Zunächst einmal war es, abgesehen von Ausnahmen, kaum möglich, sich außerhalb der festen Projekttermine in voller Gruppenstärke zu einem festen Termin zu treffen.

Um in unseren Arbeitsabläufen dennoch eine gewisse Kontinuität herzustellen, haben wir zwei wöchentliche Termine festgelegt, bei denen immer mindestens die Hälfte der Gruppe anwesend sein konnte. Als Sprintdauer haben wir zwei Wochen festgelegt, hauptsächlich basierend auf der Erfahrung, wie lange einzelne Features in anderen universitären Projekten gedauert haben. In gewisser Hinsicht haben wir hier also eine Scrum-ähnliche Retrospektive benutzt, um unseren ersten Sprint zu planen.

Das Scrum-Framework verbietet eine Aufteilung in Unterteams. Zugleich wurde uns aber vermittelt, die Wegnahme einzelner Scrum-Elemente oder Missachtung einzelner Regeln bedeute, gar nicht mehr Scrum zu benutzen, da Scrum unteilbar sei. Ein weiteres Problem ergab sich dadurch, dass Scrum unserem Verständnis nach vorsieht, dass das Entwicklungsteam zu Beginn der Arbeit bereits alle notwendigen technischen Kompetenzen zur Umsetzung eines Projekts besitzt. Zuverlässige Schätzungen für die Entwicklungszeit sind andernfalls schwer möglich. Universitäre Projekte haben aber natürlich auch immer eine Komponente, in der sich die Teilnehmer selbstständig das Wissen erarbeiten, das zur Vollendung einer Aufgabe notwendig ist. Diese Überlegung haben wir in unsere Schätzung mit einbezogen. Dennoch ist es an technisch schwierigen Stellen schwer abzusehen, wie lange es dauern wird, eine spezielle Lösung zu finden. Insofern sind Hilfen wie Burdown-Charts dann nicht besonders indikativ dafür, wie sich bisher Erledigtes zu verbleibenden Aufgaben mit Blick auf die Einarbeitungsschwierigkeiten

verhält.

5.2 Von Alex

Zu Beginn des Vorgehens standen Ideenfindung, Konzeption, Lernprozesse und finden eines gemeinsamen Arbeitsrhythmus im Vordergrund. Eine erste grobe Orientierung welche Aufgabenteile die einzelnen Beteiligten der Gruppe im Verlauf annehmen würden, ergab sich nach der „Kick-Off“-Veranstaltung bei T-Systems. Die Kompetenzen des Teams wurden anschließend auf die Positionen des Entwicklers, Konzepters, Projektleiters und Beraters aufgeteilt. Zum Zeitpunkt der Ideenfindung wurden Vorgehen noch weniger zentral organisiert, wie es später durch den Projektleiter realisiert werden sollte. Es galt zunächst herauszufinden mit welchen konkreten Aufgaben man sich im weiteren Verlauf konfrontieren konnte. Um eine grundlegende Kommunikation zwischen den Teammitgliedern zu etablieren wurde eine Facebook-Gruppe gegründet, die dazu diente Ideen festzuhalten, zu besprechen und Termine oder Aufgaben für kommende Treffen zu vereinbaren. Um die Ideen darüber hinaus geordnet aufzuschreiben und für alle Beteiligten abrufbar zu machen, wurde zusammen mit dem Coderepository ein Wiki bei Github angelegt. Das Wiki fand im speziellen in dieser frühen Phase Verwendung.

Neben der Formfindung von Konzepten diente das Wiki ebenfalls bspw. zum Festhalten von Code-Style Konventionen (NYTimes Objectvice-C-Style Guide) oder Workflows wie dem tool-unterstützten Erstellen von Doxygen-kompatiblen Methodenkommentaren mit dem Xcodeplugin VVDocumenter (<https://github.com/onevcv/VVDocumenter-Xcode>). Auf Basis des letzterem ließe sich bei Bedarf aus den Kommentaren eine Dokumentation im html-Format generieren.

Einen der ersten Schritte im Arbeitsprozess nach Erstellung Exposés konnten wir mit Hilfe einer Umfrage bezüglich der Kundenbedürfnisse, -erwartungen und -bedenken respektive des Genres der Applikation, sowie unserer spezifischen Vorstellung, sprich der möglichen Features des Endprodukts, erreichen. Im Rahmen der ersten internen Treffens des Teams und der Plenumsveranstaltungen konnten sich die Vorstellungen langsam konkretisieren. Deutlich wurde dabei wie die diese zu priorisieren sind und welche Einschränken (wie bspw. Sicherheitsaspekte, Wertigkeit des Features) an eine mögliche Umsetzung geknüpft sind. In diesem Prozess wurde das Konzept stetig auf die Rückmeldungen der Plena angepasst. Begleitet wurde das Vorgehen

durch das Erstellen von Mockups, die bereits in diesem Stadium helfen konnten Inhalte nicht zuletzt visuell zu vermitteln und zu besprechen. Die Verwendung von Mockups fand aber auch im weiteren stets Anwendung um Skizzen für eine mögliche Visualierung zu erstellen, bevor diese final umgesetzt wurden.

Besonderes Augenmerk lag in der frühen Phase bezogen auf das Konzept im Herausarbeiten eines Alleinstellungsmerkmals durch das sich mit Hilfe der Applikation die Filialbank von der Direktbank positiv abheben kann. Dieser Punkt markierte gewissermaßen bei unserem Team die Hürde, um mit der eigentlichen Umsetzung zu beginnen. Die Featureideen sortierten wir fortlaufend neu nach geschätzter Priorität. Im Wiki unterschieden wir dabei grundsätzlich zwischen „Major“ und „Minor“. Mit den Major-Features sollten die Kernfunktionalitäten der Applikation beschrieben werden. Hierzu zählten unter anderem der Filialfinder, der individuell angepasste Kreditrechner, sowie später primär fokussiert der Self-Service. Features welche die App in der Gesamtheit aufwerten sollten aber nicht zur Kernfunktionalität gehören, bildeten die Minor Kategorie. Ein Beispiel hierfür ist die Überweisungsfunktion für Girokonten. Sie wird in diesem Kontext erwartet, stellt aber keine Möglichkeit zur Abgrenzung gegenüber ähnlichen Anwendungen dar. Sie trägt außerdem nicht direkt dazu bei, die übergeordnete Fragestellung den Kunden wieder mehr an die Filialbank zu binden, zu erfüllen. Im Rahmen der Priorisierung verdeutlichte sich des weiteren, welche Features später nicht umgesetzt würden. Ein Feature das aufgrund niedrig eingestufter Priorität nicht den Weg bis in das Endprodukt geschafft hat, ist bspw. der SEPA-Umrechner, der das zuvor bestehende Format in die neue Kodierung umrechnen sollte.

Ebenfalls gab ausgedehnte Überlegungen, ob die Software für iPad, iPhone oder sogar dual für beide Geräte entwickelt werden sollte. Wie bei den Features war es auch hier hilfreich, wenn auch nicht einfach, demokratisch über eine Entscheidung abzustimmen. Dies zog selbstredend nach sich, dass das Kollektiv den Einzelnen teils entgegen seiner eigenen zu einer gemeinsamen Lösung stimmte. Da die Kernfrage der Kundenbindung an die Filialbank schwer zu beantworten war, kam die Fragestellung des Zielgeräts und der damit verbundenen Auswahl an Features nach einer zuvor vermeintlich finalen Entscheidung mehrmals auf. Schlussendlich fiel die Entscheidung auf das iPad, weil wir die Kernfeatures durch dieses besser abdeckt sahen und sich nach der Umfrage der Eindruck einstellte, dass Nutzer sicherheitskritische Anwendungen bevorzugt in Ruhe Zuhause statt unterwegs verwenden wollen. Der Entschluss sich bei der Entwicklung

auf ein Gerät zu fokussieren, wurde teamintern unter den Gesichtspunkten mangelnden Gewinns einer dualen Entwicklung und dem damit verbundenen zusätzlichen Aufwand entschieden.

Nachdem das grundlegende Konzept ausgearbeitet war, konnten wir mit konkreten Umsetzungen beginnen. Infolgedessen musste sich das Team auf ein Satz von Tools einigen, mit denen die Entwicklungsaufgaben verwaltet, zugeteilt und festgehalten wurden. Initial fiel die Entscheidung auf eine Kombination von Google-Docs und Github-Issues/Milestones. In Github lassen sich sog. Issues definieren. Diese Issues stellen Aufgaben dar, die Mitgliedern des Teams zugewiesen werden. Einzelne Issues werden wiederum Zwischenzielen zugewiesen, den Milestones. Sind alle Issues bezüglich eines Milestones abgearbeitet, ist dieses erfüllt. Die Verwaltung und Verteilung der Aufgaben wurde von diesem Zeitpunkt an durch den Projektleiter zentral durchgeführt. Alle neu entstehenden Aufgaben wurden also immer in Absprache mit diesem in das Issuesystem eingefügt. Dies war für die Übersicht der kommenden Aufgaben, verbunden mit der Gewissheit, dass alle Mitglieder über den Fortschritt im Ganzen wie im Einzelnen den gleichen Informationsstand haben, eine echte Hilfe. Die in Github eingetragenen Issues wurden neben der Zuweisung an Personen mit Labels versehen, die zum einen die Priorität zum anderen, das Überthema der Aufgabe beschreiben. Hochprior wurde u.a. das Erstellen einer Basisarchitektur auf der im weiteren softwaretechnisch aufgebaut werden sollte, sowie das Beschreiben der vorkommenden Entitäten und der damit verbundenen Relationen eingestuft.

Während wir Github zum Zuteilen und Festhalten der Aufgaben einsetzten, konnte mit Google-Docs der zeitliche Aufwand dafür festgehalten werden. Pro Aufgabe wurde eine Zeile mit entsprechendem Titel angelegt. Falls vorhanden, ist in der Tabelle das korrespondierende Github-Issue mit Verlinkung eingetragen. Die den Aufgaben zugewiesenen Teammitglieder sind mit Initialen vermerkt. Aus der tatsächlich aufgewendeten Zeit multipliziert mit der Anzahl der zugeteilten Personen resultiert der kumulierte Aufwand. Die Vorstellung der zu erledigenden Arbeiten war zu diesem Zeitpunkt allerdings meist unvollständig konkretisiert. Davon abgesehen gab es keine Vorerfahrungen bezüglich des iOS-Frameworks. Entsprechend mussten wir davon ausgehen, dass Aufwandsschätzungen zu Beginn nur bedingt hilfreich sein konnten. Trotzdem war dies als Vorlauf evtl. wichtig, um im weiteren Verlauf ein gutes gemeinsames Verständnis für die Dokumentation und Verwaltung des Arbeitsaufwands zu entwickeln und zu präzisieren.

Während der Anfangsphase etablierte sich im Team ein Arbeitsrhythmus mit zwei Terminen pro Woche in denen jeweils das Arbeitspensum eines Werktages oder darüber hinaus absolviert wurde. Die zeitlichen Gegebenheiten mussten auf unsere Gruppe von Studenten mit je unterschiedlichen Stundenplänen abgestimmt werden. In voller Gruppenstärke war es, abgesehen von Ausnahmen, folglich kaum möglich, außerhalb der festen Projekttermine zusammenzukommen. Um in unseren Arbeitsabläufen dennoch Kontinuität herzustellen, haben wir die vereinbarten Termine entsprechend organisiert, dass mindestens die Hälfte der Gruppe anwesend sein konnte. Folglich wurden die Termine alternierend in unterschiedlichen Besetzungen durchgeführt.

Die frühen Teamtreffen waren neben den eigentlichen Aufgabenstellungen stark davon geprägt sich mit Objective-C dem iOS-Framework sowie der Entwicklungsumgebung vertraut zu machen. Vielleicht wurde auch infolge dessen zunächst primär eine technische Umsetzung unter Vernachlässigung visueller Aspekte versiert. Nach der Rückmeldung in einem der frühen Plenumstermine, dass visuelle Gestaltung der technischen zeitlich nicht nachsteht und parallel dazu entwickelt werden sollte, wurde die Arbeitsweise dahingehend umgestellt. Technische und visuelle Gestaltung wurden von diesem Zeitpunkt an gleichzeitig entwickelt.

Im Anschluss an die Scrum-Zertifizierung wurden Arbeitsabläufe und Positionierung der Mitglieder im Team erneut reflektiert, um einen Scrum-Prozess mit den entsprechenden Rollen zu realisieren. Korrespondierend zu den zuvor zugewiesenen Positionen des Entwicklers, Projektleiters, Beraters und Konzepters wurde nach Äquivalenten in der Scrum-Terminologie gesucht. Entwickler und Konzepter wurden der Gruppe des Development Teams zugewiesen. Während beim Berater eine Analogie zum Scrum Master gebildet wurde, lag es nahe den Projektleiter dem Product Owner zuzuordnen.

Bereits vor der Zertifizierung hatte sich durch den Scrum Master die Praxis eines Daily Scrums eingestellt. Das Daily Scrum Meeting wurde immer zu Beginn jedes Treffens umgesetzt. Dies brachte den positiven Effekt mit sich, dass trotz alternierender Besetzungen sich alle Mitglieder zu Arbeitsbeginn über den aktuellsten Stand ausgetauscht hatten. Der Product Owner zuvor Projektleiter verwaltete weiterhin wie zuvor beschrieben zentral das Aufgabenmanagement. Problematisch erwies sich die zuvor getroffene Toolauswahl zur Organisation der Aufgaben unter den Gesichtspunkten von Scrum. Primär fehlte es den Tools an Möglichkeiten der Idee des ite-

rativen Sprints gerecht zu werden. Infolgedessen stellte das Team die gesamte Toolchain zur Verteilung der Aufgaben und deren Aufwandsmessung um. Die Wahl fiel auf das Tracking-Tool Pivotal Tracker in welchem die Terminologie Scrums mit Sprint-Backlog, Sprint-Velocity, „Done“- Status, usw. vertreten war. Die zuvor erstellten Issues und Aufwandseinschätzungen der noch offenen Aufgaben wurden komplett in die neue Umgebung übertragen.

Als Iterationsdauer legten wir einen Zeitrahmen von zwei Wochen fest. Diese Dauer erschien uns als ein sinnvoll; Einerseits lang genug um kleine oder mittelgroße Aufgaben innerhalb einer Iteration zu bewerkstelligen, anderseits nicht zu lang um dynamisch auf geänderte Anforderungen reagieren zu können, ohne den Sprint-Backlog innerhalb eines Sprints modifizieren zu müssen. Der Prozess des dynamischen Reagierens auf Kritik und sich damit ändernden Anforderungen vollzog sich über die gesamten Zeitspanne des Projekts. Das stetige Überdenken und Anpassen voriger Überlegungen und Ausarbeitungen stellte sich als eine der größten, wenn nicht als die größte Herausforderung des Projekts dar. Eine ständige Korrektur führt auch zum Verwerfen voriger Arbeit, woraus die Anforderung entsteht sich stetig für neue Richtungswechsel zu motivieren.

Nach der Anfangsphase wurden die Aufwandsschätzungen präziser. Weiterhin war es zwar schwierig genaue Schätzungen für einzelne Aufgaben vorzunehmen, insgesamt konnten wir aber trotz alledem auf Basis der sich ergebenden Sprint-Velocity ab dem letzten Drittels des Semesters eine sinnvolle Einschätzung über den verbleibenden Arbeitsaufwand und die dafür benötigte Zeit abgeben. Wie sich herausstellte, konnten wir diese Einschätzung auch einhalten.

Bezüglich der strengen Richtlinien von Scrum, entweder es wird auf voller Linie praktiziert oder aber das Vorgehensmodell ist nicht eingehalten und somit auch anders zu benennen, ist noch erwähnenswert, dass diese eigentlich vorsehen, dass das Development Team zu Beginn seiner Arbeit bereits alle notwendigen technischen Kompetenzen zur Umsetzung eines Projekts besitzt. Wir waren allerdings damit konfrontiert selbstständig fortlaufend Wissen zu erarbeiten, das zur Bewerkstelligung der Aufgaben notwendig war. Davon abgesehen war das Team auch in weiteren Punkten nicht durchweg sicher ob es alle Vorgaben des Modells einhalten könne.

Neben den erwähnten Vorteilen des Tracking-Tools respektive Struktur und Terminologie, ermöglichte uns die neue Umgebung das Verhältnis von bestehenden und noch zu erledigenden

Aufgaben in Balkendiagrammen und Burndown-Charts zu visualisieren. Der Burndown-Chart bzw. Progress-Report über die gesamte zweite Hälfte der Projektdauer ergibt sich wie folgt:

Aus der Tendenz ist ersichtlich, dass das die Sprint-Velocity unter Vernachlässigung der Feierzeit stetig zur bis zur Deadline zunimmt und sich aufgrund zeitlichen Abschließens in der letzten Woche entspannt.

6 Ergebnis von Marco F. Jendryczko

6.1 Features

6.1.1 Dashboard

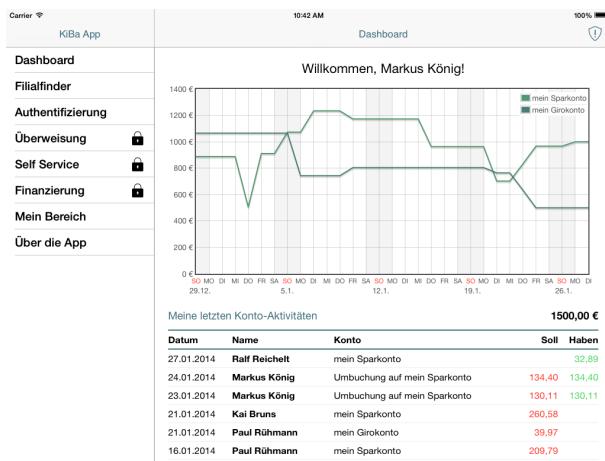


Abbildung 5: Dashboard

Das Dashboard ist die zentrale Anlaufstelle der Applikation, dort haben wir einen Überblick über den Verlauf der Kontostände aller unserer Konten. In der Tabelle darunter haben wir die Möglichkeit alle Kontobewegungen nach zu verfolgen.

6.1.2 Filialfinder

Beim Filialfinder findet man eine Karte von Googlemaps mit Markierungen, welche die Standorte der einzelnen Filialen markieren. Mit einem Klick auf das Infosymbol gelangt man zur Filialseite auf der man die Öffnungszeiten nachschauen, aber auch eine Terminanfrage oder eine Sortenanfrage stellen kann.

6.1.3 Sortenanfrage

Mit einem Klick auf Sortenanfrage bekommt der Nutzer ein Pop-Over zu sehen, in welchem er seine Wunschwährung auswählen kann die entsprechenden Umrechnungskurse werden intern

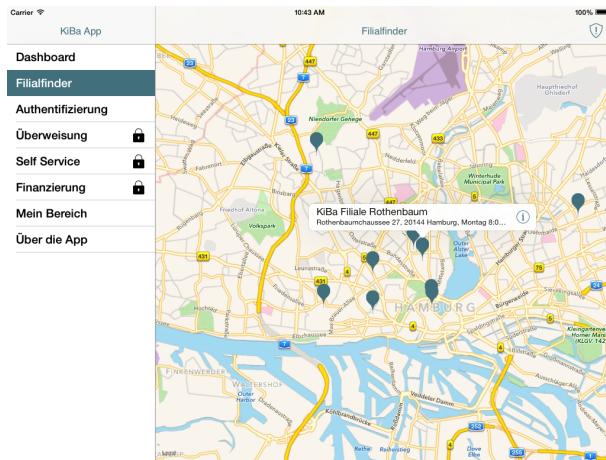


Abbildung 6: Goolemaps mit Filialmarkierungen

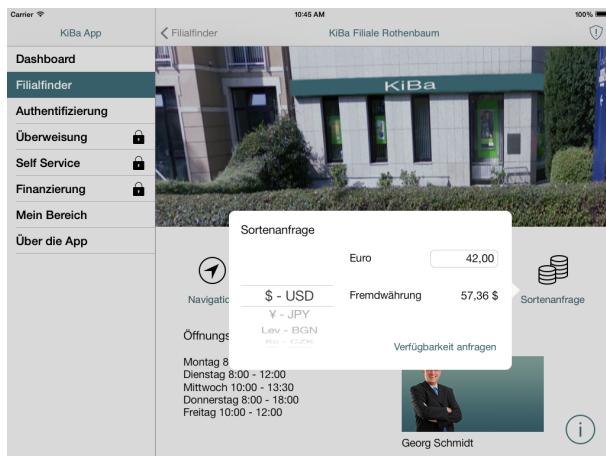


Abbildung 7: Filialseite mit Sortenanfrage Pop-Over

direkt von der Europäischen Zentralbank bezogen. Hat man die Summe, welche umgetauscht werden soll eingegeben, so kann man die Anfrage stellen und bekommt eine Nachricht, ob die gewünschte Währung in der gewünschten Höhe bei der gewünschten Filiale verfügbar ist.

6.1.4 Authentifizierung

Im oberen rechten Rand der Applikation ist ein kleines Symbol in Form eines Schildes zu sehen, dieses gibt den Authentifizierungsstatus zurück. Ein Schild mit einem Ausrufezeichen symbolisiert eine fehlende Authentifizierung, ein Haken eine gültige Authentifizierung.

Solange das Gerät nicht authentifiziert ist, kann man einige Funktionen nicht benutzen, diese



Abbildung 8: Die Authentifizierungsstati

sind durch das Symbol mit dem geschlossenen Schloss gekennzeichnet.

Unter dem Menüpunkt Authentifizierung findet man ein kleines Comic vor. Dieses erläutert wie der Nutzer sein Gerät bei seiner Filiale authentifizieren kann. Außerdem wird der Nutzer auf seine Vorteile aufmerksam gemacht.

6.1.5 Überweisung

Unter dem Menüpunkt Überweisung findet man ein ganz gewöhnliches Überweisungsformular, die Überweisung muss man abschließend noch mit einem Tan bestätigen.

6.1.6 Self-Service Station

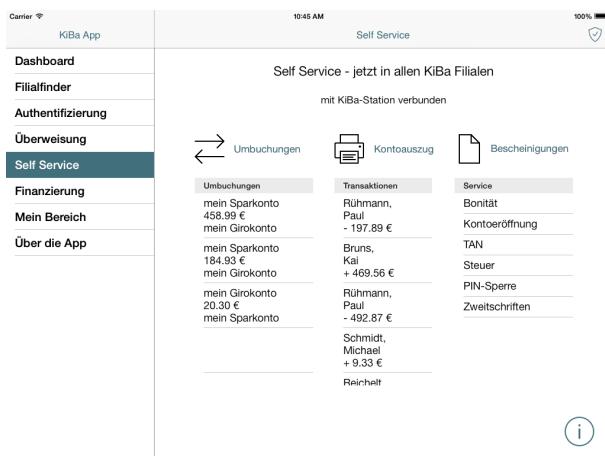


Abbildung 9: Self-Service Startseite. Die App ist verbunden.

Bei der Self-Service Station haben wir die Möglichkeit, sofern wir mit dem Stationsgerät verbunden sind, drei verschiedene Aktionen eigenständig durchzuführen. Eine Umbuchung, Kontoauszüge anschauen und drucken, sowie Bescheinigungen ansehen, ausfüllen und ausdrucken gegeben falls direkt abschicken.

Jede Auswahlmöglichkeit hat eine kleine Tabelle mit Inhalten, die den User erwarten, wenn er auf den entsprechenden Button drückt.

6.1.7 Bescheinigungen

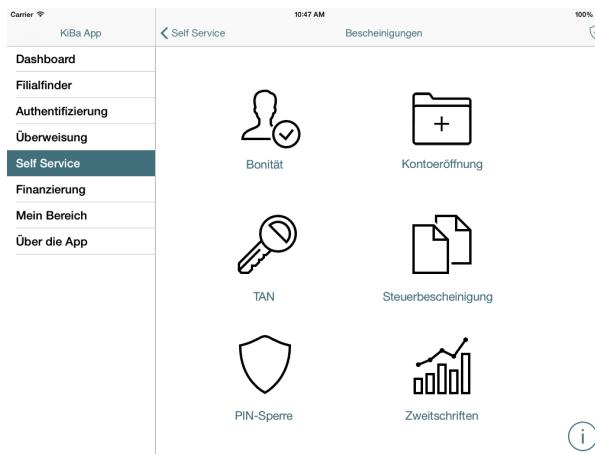


Abbildung 10: Die verschiedenen Dokumenttypen

Bei den Bescheinigungen sehen wir verschiedene von der Bank zur Verfügung gestellten Dokumente die der Nutzer auswählen kann.

6.1.8 Kontoauszüge

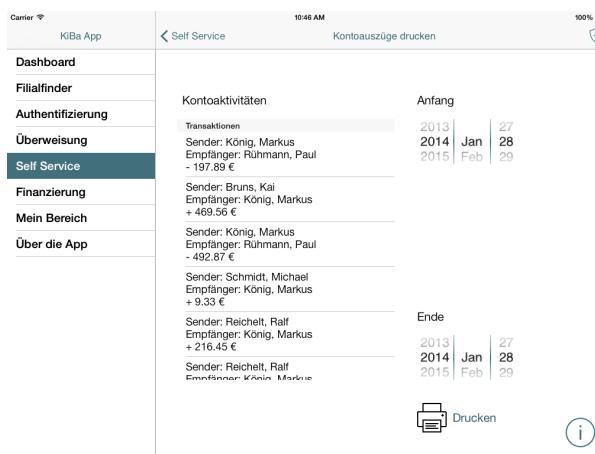


Abbildung 11: Übersicht der Kontoauszüge

Im Kontoauszugsbildschirm kann man Kontoaktivitäten in einem bestimmten Zeitraum drucken lassen. Somit kann man auch Kontoauszüge ausdrucken, welche man beispielsweise verloren hat und muss dies nicht beim Schalter beantragen.

6.1.9 Umbuchungen

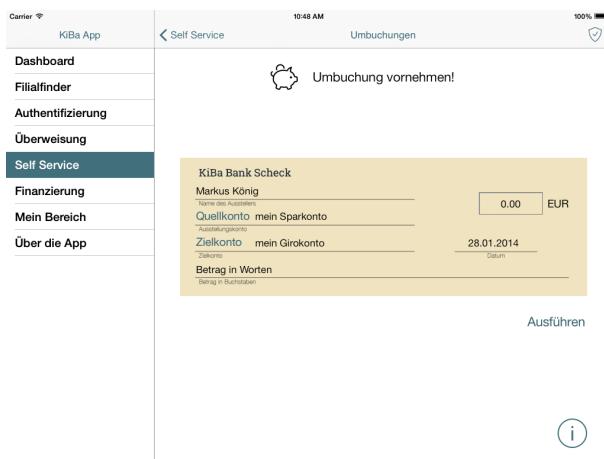


Abbildung 12: Eine Scheckgrafik als Formular für die Umbuchung

Im Umbuchungsbildschirm kann der Nutzer sowohl Ziel- als auch Quellkonto auswählen, zwischen denen der angegebene Betrag transferiert werden soll. Das ganze wird mit einem Button bestätigt.

6.1.10 Finanzierungsrechner

Mit dem Finanzierungsrechner kann man sich Kreditkonditionen zusammenstellen. Die Werte der einzelnen Schieberegler werden aus dem Profil des Kundens von der Bank vorgegeben.

Man kann danach auch wieder einen Beratungstermin vereinbaren um die Konditionen mit seinem Berater nochmal durchsprechen zu können, inwiefern die ausgewählten Konditionen empfehlenswert für den Kunden sind oder ob die Bank vielleicht noch bessere Sonderkonditionen anbieten kann.

The screenshot shows the KiBa App interface for financing calculations. The top bar displays 'Carrier', 'KiBa App', the time '10:49 AM', and battery level '100%'. The main section is titled 'Ihr persönlicher Kreditrechner' (Your personal credit calculator). It includes input fields for 'Gesamtbetrag' (12000 €), 'Laufzeit' (10 Jahre), 'Zins' (5.8 %), and 'Jahresraten' (1615.00 €). Below these, it states 'Netto Darlehensbetrag: 17482 €' and 'Beratung vereinbaren' (Consultation to be arranged). A help icon (i) is located at the bottom right.

Abbildung 13: Finanzierungsrechner mit individuellen Konditionen

The screenshot shows the KiBa App interface for the user's area ('Mein Bereich'). The top bar displays 'Carrier', 'KiBa App', the time '10:49 AM', and battery level '100%'. The main section is titled 'Nachrichten' (Messages). It lists several messages from 'Sehr geehrter Herr König': '28. Januar 2014 Ihre Sortenanfrage', 'Ihre Terminanfrage', 'Terminbestätigung', '06. Januar 2014 Ihr verbesselter Zins', and five additional messages represented by horizontal lines. A help icon (i) is located at the bottom right.

Abbildung 14: Ein minimalistischer Posteingang des Nutzers

6.1.11 Mein Bereich

Der Nutzer kann in seinem persönlichen Bereich Nachrichten der Bank empfangen und somit beispielsweise eine Bestätigung für seine Terminanfrage erhalten. Mit einem Klick auf die Zelle einer Nachricht bekommt man die entsprechende Nachricht in einer neuen Ansicht mit dem kompletten Nachrichtentext.

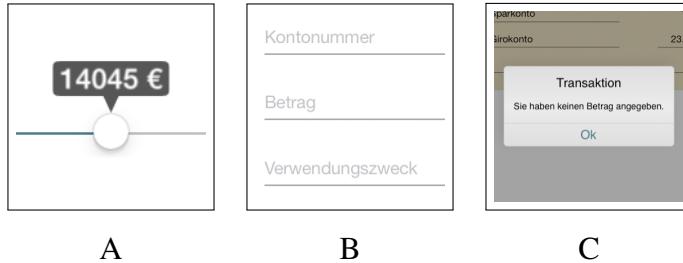


Abbildung 15: Eigenkreationen für das Design

6.2 Die Design-Highlights

Um unserer Applikation in mancher Hinsicht besser auf unsere Bedürfnisse anzupassen haben wir uns bei einigen Designelementen über die Standardimplementierungen von Apple hinweggesetzt. Dennoch blieben die iOS7 Richtlinien ein Qualitätsmaßstab für uns.

Wie in (A) zu sehen ist, bekam unser Schieberegler als kleine Erweiterung eine schwarze Sprechblase, welche den aktuellen Wert anzeigt. Dies ist unter anderem für die Bedienbarkeit implementiert worden, damit man beim Verschieben des Regler nicht immer auf die linke Seite schauen muss um den aktuellen Wert abzulesen.

Die mitgegebenen Textfelder von Apple haben in der Regel immer einen Text, welcher dem Nutzer einen Hinweis gibt was für ein Inhalt erwartet wird. Diese Umsetzung sieht oftmals unschön aus und nimmt viel Platz ein. Mit unserer Implementierung (B) verliert das Textfeld keine Funktionalität ist aber schlanker.

Die Standard Pop-Over lassen sich nur bedingt anpassen. So konnten wir die Farbe des Buttons nicht in unserem Farbton einfärben. Daher mussten wir auf eine Eigenimplementation (C) zurückgreifen um die Konsistenz zu wahren.

6.3 Herausforderungen

Mit zu den größten Herausforderungen war das Design der einzelnen Bildschirme. Das „Flat“-Design braucht ein gutes Händchen und kleine Details können schon zu Unstimmigkeiten führen. Gleichzeitig war uns bewusst, dass wir eine Bank repräsentieren und somit ein gewisses Maß an Seriosität erforderlich war. Gleichzeitig mussten wir die Usability für den Endnutzer gewährleisten.

Wir mussten ein App-Design erschaffen, welches drei Stakeholder gleichzeitig zufriedenstellt.

Eine Herausforderung geht auch immer mit einer neuen Programmiersprache und deren Entwicklungsumgebung einher. Die ersten Wochen hatten wir eine relativ geringer Produktivität, weil wir uns auf diese Gegebenheiten erst einstellen mussten. Je weiter jedoch das Projekt voranschritt, desto sicherer fühlten wir uns in Objective-C und im Umgang mit Xcode.

Während der Entwicklung stießen wir auch immer wieder auf Einschränkungen seitens Apple bezüglich der Standard GUI-Elemente. So konnten wir beispielsweise die Tint-Color eines Buttons im Pop-Over nicht ändern. Apple lässt es nicht zu. So mussten wir unser eigenes Pop-Over implementieren.

Nichtsdestotrotz konnten wir alle größeren und kleineren Herausforderungen meistern und hatten nie das Gefühl vor einem unüberwindbaren Problem zu stehen.

7 Architektur von Markus Fasselt

Nachdem zunächst in diesem Bericht bereits Konzepte und Ergebnisse erläutert wurden, gehen wir an dieser Stelle nun auf die Implementierung und technische Umsetzung der App ein. Im Fokus dabei steht die Architektur, die maßgeblich zum Ablauf der Entwicklung und zur Organisation der Kernkomponenten beiträgt.

Unsere App soll den Kontakt zwischen einer Filialbank und seinen Kunden stärken. Da es sich bei KiBa lediglich um eine fiktive Bank handelt und die App auch anderen interessierten Banken vorgestellt werden soll, bietet es sich an, einen „Click-Dummy“ zu entwickeln. Dieser soll sich bereits wie eine vollwertige Banking-App bedienen lassen, die jedoch an keine reale Bank, respektive deren Datenbank, angeschlossen ist. Aufgrund dieser Rahmenbedingungen haben wir uns für die Architektur entschieden, die im Folgenden vorgestellt wird.

7.1 Umsetzung des MVC-Ansatzes

Die Architektur muss uns dabei auf die Entwicklung der Kernfeatures fokussieren. Wenn nicht klar ist, wo welche Teile der Logik, der GUI oder anderer Komponenten zu platzieren sind, verzögert dies den Entwicklungsprozess und macht das Entwicklungsteam weniger produktiv.

Daher fiel unsere Entscheidung auf den MVC-Ansatz; das heißt, wir versuchten unseren Code in Datenmodellierung, Visualisierung und Kontextualisierung zu gliedern.

Auf oberster Ebene befinden sich deswegen nun die Ordner „Entities“ für die Modellierungsanitäten, „Services“ für Dienstleisterklassen, „Views“ für View-Controller und die die View repräsentierenden xib-Dateien und zuletzt ein „Library“-Ordner, der für alle Bereiche unterstützende Helferklassen sammelt.

Innerhalb dieser Ordner haben wir darüber hinaus eine funktionale Substruktur erschaffen, die nach unseren Feature-Komponenten aufgebaut ist. Das hatte auch den positiven Nebeneffekt, dass kollaboratives Arbeiten am Projekt erleichtert wurde. Denn auf diese Weise war es nicht zwangsläufig für jede Einheit erforderlich, eine eigene Git-Branch zu eröffnen, da in den Ordnern gearbeitet werden konnte.

7.2 Datenmodell

Zur Modellbildung der App fiel unsere Entscheidung auf ein klassisches Entitäten-Beziehungs- system. Unser Weg zur Abstraktion einer Bank führt daher über Klassen, die Modelle zu Objekten der realen Welt darstellen. Wie wir dabei vorgegangen sind, ist im Entitäten-Relationen-Diagramm unseres Datenmodells in Abbildung 16 zu sehen.

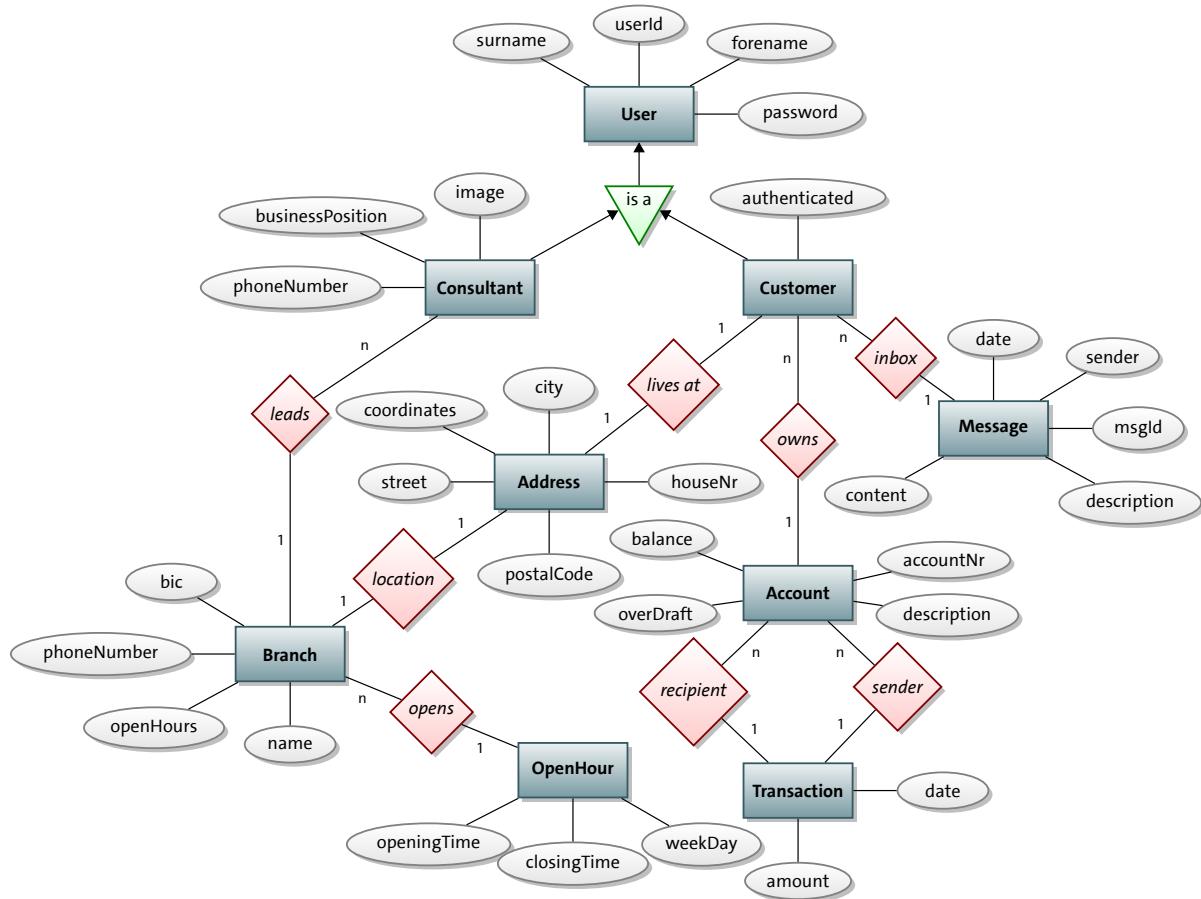


Abbildung 16: Entitäten-Relationen-Diagramm

Da gibt es zum einen die mit der Bank agierenden Benutzer, nämlich den Berater und den Kunden, die von einer gemeinsamen Klasse `User` erben. Kunden, sprich `Customer`, besitzen mehrere Konten (`Account`) und erhalten mehrere Nachrichten (`Message`). Des weiteren verfügen sie, ebenso wie die Filialen, über eine Adresse. Diese Filialen (`Branch`) wiederum werden von Beratern geleitet und sind zu verschiedenen Öffnungszeiten (`OpenHour`) besuchbar. Darüber hinaus finden Transaktionen (`Transaction`) zwischen Konten statt.

7.3 Sicherheitsaspekte

Ein wichtiger Aspekt, insbesondere im Hinblick auf den Umgang mit sensiblen Finanzdaten, ist die Sicherheit der App. Dabei ist es auch eine Aufgabe der Architektur, diese gewährleisten zu können. Im Folgenden erläutern wir die Schritte, die wir dementsprechend für die realistische Umsetzung zogen.

Eine Fragestellung von essentieller Bedeutung für uns ist, was mit der App im Falle eines Diebstahls passieren würde. Da wir sowohl der Bank als auch dem Kunden gewährleisten müssen, dass ihre Daten sicher sind, haben wir das Risiko zu groß eingestuft, die Daten auf dem Gerät zu speichern. Deswegen liegen alle Informationen nur im Zwischenspeicher. Das hat für uns den Vorteil, dass beim Beenden der App alle Informationen verloren gehen, wenn der Kunde nicht mehr eingeloggt ist.

Zudem würde in einer über einen Dummy hinausgehenden Implementierung ein Time-Out Token implementiert, welche nach beispielsweise zehn Minuten ohne Interaktion die Anwendung beendete. Außerdem gewährleistet der Authentifizierungsmechanismus, dass kritische Features serverseitig deaktiviert werden, indem der entsprechende Benutzer wieder auf nicht authentifiziert geschaltet wird. Dieser Mechanismus kann einerseits bei einer Verlustmeldung greifen, aber auch bei in irgendeiner Weise verdächtigem Verhalten, also ungewöhnlich viele oder große Transaktionen in bestimmten Zeiträumen.

Wichtig ist außerdem, dass alle Daten direkt übertragen werden. Wir stellen uns ein direktes Protokoll wie eine REST-Schnittstelle oder ein SOAP-Verfahren zum Austausch der Informationen zwischen App und Server der Bank vor.

Insbesondere letzter Punkt kann es erforderlich machen, dass die Datenquelle anpassbar sein muss. Eine Möglichkeit, das zu realisieren, erläutern wir im nächsten Abschnitt.

7.4 Dependency Injection

Eine zentrale Anforderung der App-Architektur für uns ist außerdem das Austauschen von einzelnen Kernkomponenten, wie etwa die Datenschicht. Denn hierdurch kann aus dem Click-Dummy eine vollwertige Banking-App erschaffen werden können, ohne große Änderungen am

Code vornehmen zu müssen. Sie muss uns den Eindruck nehmen, an eine echte Bank gebunden zu sein.

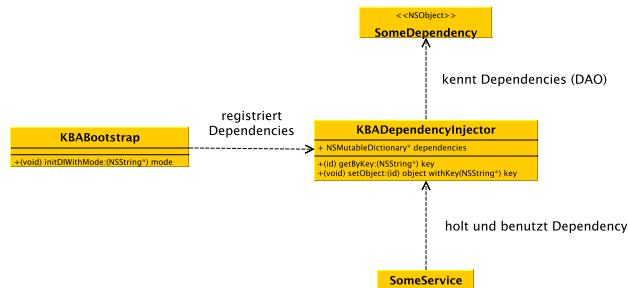


Abbildung 17: Klassendiagramm unserer Dependency Injection

Daher ist für uns Dependency Injection als Entwicklungsmuster die Lösung dieses Problems. Wir haben es in einer reduzierten Form selbst versucht zu implementieren. Dabei wird an einer zentralen Stelle im Quelltext, nämlich im *Bootstrapping*, in den KBADependencyInjector wie in Abbildung 17 sichtbar registriert, welche konkrete Implementierung einer Abhängigkeit in der Anwendung verwendet werden soll. Beim KBADependencyInjector handelt es sich dabei nur um einen einfachen Schlüssel-Wert-Speicher. Der Bootstrapping-Prozess ist im Programm-ausdruck 1 wiedergegeben.

Programmausdruck 1: Der Bootstrapping-Vorgang

```

+ (void) initDependencyInjectorWithMode:(NSString *) mode {
    id<KBABranchDao> branchDao;
    id<KBAExchangeRateDao> exchangeRateDao;
    id<KBACustomerDao> customerDao;
    id<KBAccountDao> accountDAO;
    id<KBATransactionDao> transDao;
    id<KBCreditRatingDao> creditDao;
    id<KBAMessageDao> messageDao;
    KBAAuth *auth = [KBAAuth new];

    exchangeRateDao = [KBAExchangeRateDaoRest new];

    if ([mode isEqualToString:@"dev"]) {
        // if development / click dummy
        branchDao = [KBABranchDaoDummy new];
        customerDao = [KBACustomerDaoDummy new];
        accountDAO = [KBAccountDaoDummy new];
        transDao = [KBATransactionDaoDummy new];
        creditDao = [KBCreditRatingDaoDummy new];
        messageDao = [KBAMessageDaoDummy new];
    }
    else {
        // for production
    }
}
  
```

```

creditDao = [KBACreditRatingDaoRest new];
branchDao = [KBABranchDaoRest new];
customerDao = [KBACustomerDaoRest new];
accountDAO = [KBAAccountDaoRest new];
transDao = [KBATransactionDaoRest new];
messageDao = [KBAMessageDaoRest new];
}

[KBADependencyInjector setObject:branchDao forKey:@"branchDao"];
[KBADependencyInjector setObject:exchangeRateDao forKey:@"rateDao"];
[KBADependencyInjector setObject:customerDao forKey:@"customerDao"];
[KBADependencyInjector setObject:accountDAO forKey:@"accountDao"];
[KBADependencyInjector setObject:auth forKey:@"auth"];
[KBADependencyInjector setObject:transDao forKey:@"transDao"];
[KBADependencyInjector setObject:creditDao forKey:@"creditDao"];
[KBADependencyInjector setObject:messageDao forKey:@"messageDao"];
[auth login: @"max" withPassword:@"test"];

} // end of initDependencyInjectorWithMode:

```

Wie zu erkennen ist, besitzt jedes Data Access Object (DAO) ein Protocol, welches auf zwei Arten instanziert werden kann. Dadurch erreichen wir, dass ein Programm auf verschiedene Art und Weise ausführbar ist.

Ein weiterer Vorteil dieser Abstraktion ist die Testbarkeit der App. Dadurch, dass im Click-Dummy feste Daten hinterlegt sind, kann man sich zum Testen der App verschiedene Fälle erzeugen, die einfach in den jeweiligen DummyDaos benutzt werden.

Darüber hinaus erlaubt uns dieses Verfahren auch die erleichterte Austauschbarkeit einzelner Komponenten. Wenn man dieses Verfahren auf andere Funktionen der App anwendet, insbesondere dann, wenn benutzerspezifische Elemente erforderlich sind, so ist es schnell möglich, eine White-Label-App zu erzeugen. Das heißt also wir können eine App schaffen, die auf verschiedene Kunden zugeschnitten werden kann. Dies hat für unseren Kunden T-Systems MMS in seiner Eigenschaft als Vermarkter den Vorteil, ihre App leicht auf Endkunden anpassen zu können.

Wir haben festgestellt, dass die frühe Auseinandersetzung mit Fragen der Softwarearchitektur eine gute Entscheidung ist. Durch die Implementierung unserer eigenen Dependency-Injection und der zentralen Regelung von Abhängigkeiten an einem Ort haben wir eine dynamische und nachhaltige Methode geschaffen, unsere App sukzessive erweitern zu können.

8 Toolchain von Konstantin S. M. Möllers

Im Rahmen unseres Projekts wurde uns schnell klar, dass wir auf viele Tools angewiesen sein würden. So musste unser Quelltext versioniert, die grobe Struktur und Views festgehalten und Aufgabenpakete erstellt und koordiniert werden. Darüber hinaus war uns auch Design und Qualität wichtig. Auf unsere Erfahrungen in diesem Zusammenhang gehen wir im folgenden Abschnitt ein.

8.1 Agiles Vorgehen im Projekt

Da uns freundlicherweise von C1 WPS die Möglichkeit gegeben wurde, eine Scrum-Zertifizierung zu erhalten, lag uns viel daran, den Scrum-Ansatz so gut es ging in unseren Projekt-Ablauf zu integrieren.

Dazu verwendeten wir – nach ersten Versuchen mit GitHub-Issues – die Software PivotalTracker. Mit ihr können wir, wie in Abbildung 18 dargestellt, einzelne Stories erstellen und überwachen. Ein großer Vorteil dabei war, dass Scrum-Iterationen automatisch von der Software generiert werden, berechnet durch unsere durchschnittliche Arbeitskraft (genannt „Velocity“)⁷ und unseren Vorhersagen für den Aufwand einer Story angegeben durch Punkte.

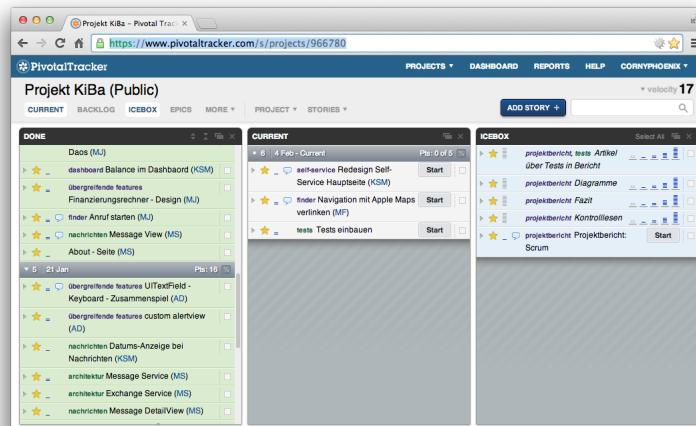


Abbildung 18: Stories in PivotalTracker

⁷<http://www.pivotaltracker.com/community/tracker-blog/velocity-matters>

Zum anderen generiert PivotalTracker auch Burndown-Charts, welche für das Vorgehen während der Scrum-Iterationen unabdinglich sind. Denn so können wir am besten den Fortschritt unserer App messen und haben ein Monitoring für das Fortschreiten der Entwicklung unserer Features, wie in Abbildung 19 zu sehen ist.

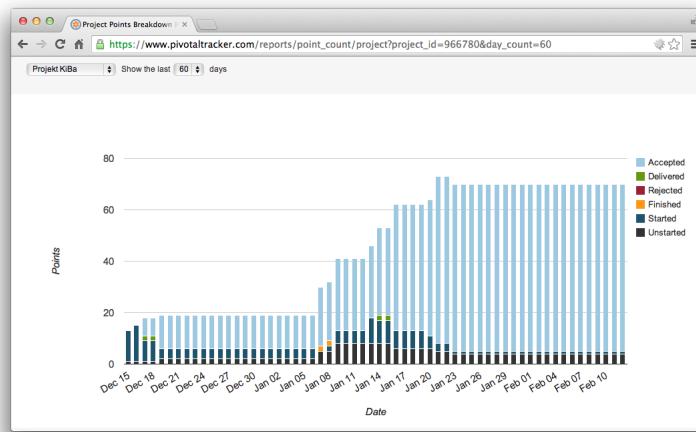


Abbildung 19: Burndown-Chart in PivotalTracker

8.2 Versionsverwaltung und Informationsaustausch

Um kollaborativ an einer Software zu arbeiten ist es selbstverständlich erforderlich, mit Quelltextversionierung zu arbeiten. Da bei uns Team-Mitgliedern die Erfahrung mit Git⁸ aus dem Universitäts- und Arbeitsumfeld am größten war, viel unsere Wahl für ein VCS darauf. Entscheidend dabei war auch der größere Komfort gegenüber Subversion und die Möglichkeit, die Plattform GitHub⁹ als Repository-Hoster und anfangs auch als Issue-Tracker zu verwenden. Wie unser Repository bei GitHub aussieht ist zu sehen in Abbildung 20.

Ein weiteres nützliches Feature von GitHub für uns war außerdem ein Wiki, dargestellt in Abbildung 21. In ihm tauschen wir Informationen aus wie beispielsweise erste Feature-Ideen oder Kontaktdaten aller Mitglieder. Allerdings nutzten wir es darüber hinaus auch für das Festlegen eines Arbeitsablaufs mit Xcode.

⁸<http://git-scm.com/>

⁹<https://github.com/>

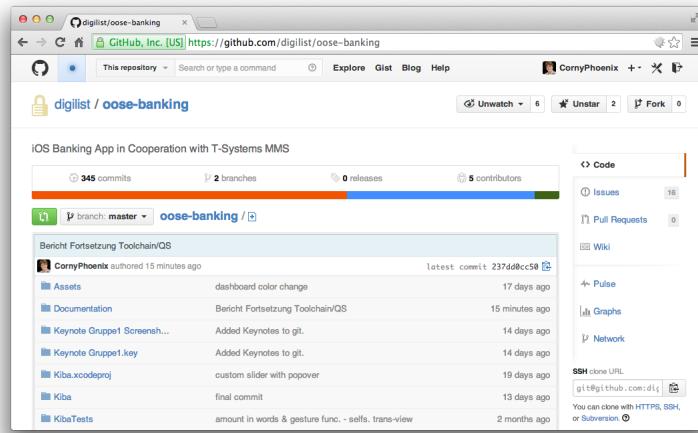


Abbildung 20: Quelltext-Hosting bei GitHub

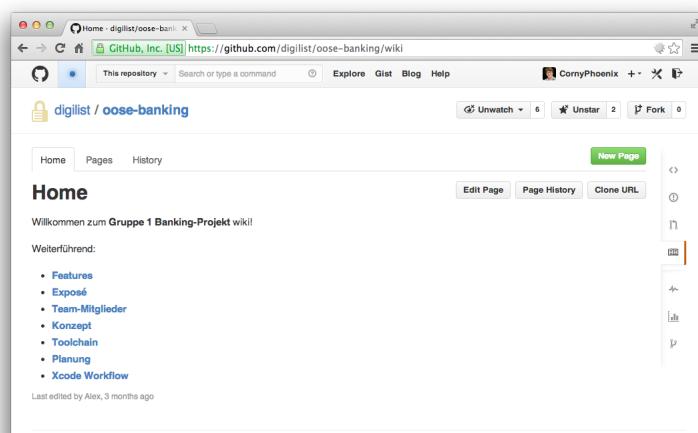


Abbildung 21: GitHub-Wiki für den Wissensaustausch

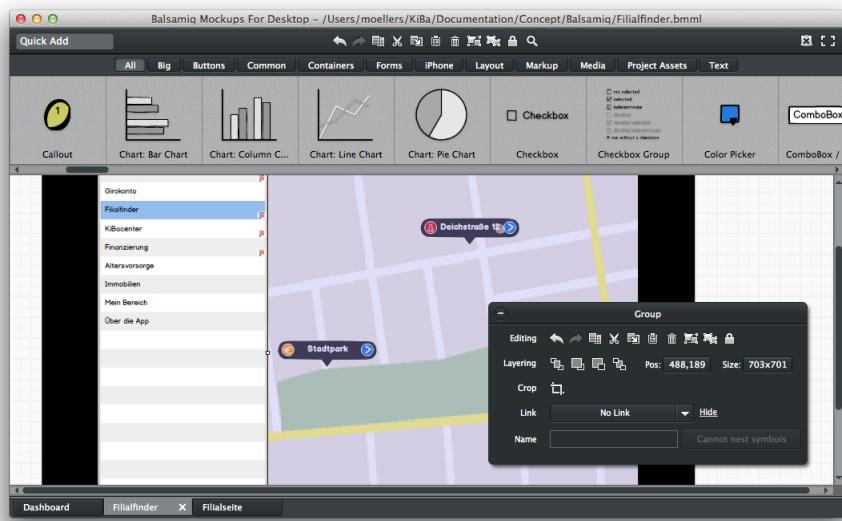


Abbildung 22: Entwerfen mit Balsamiq

8.3 Konzept und erster Entwurf

Balsamiq

8.4 Designtools

Adobe Illustrator/Photoshop

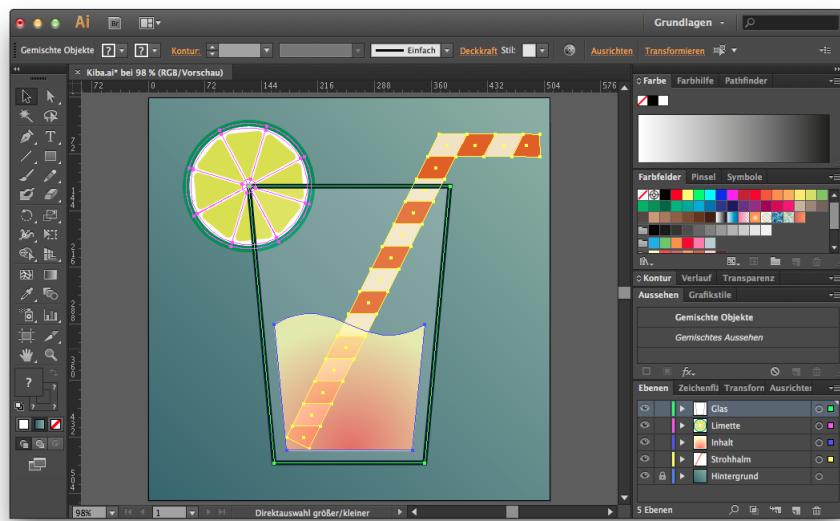


Abbildung 23: Designen des KiBa-Icons mit Adobe Illustrator

9 Qualitätssicherung von Konstantin S. M. Möllers

Für große Projekte wird es zunehmend unabdinglich, über Planung und Ausführung hinaus auch ein Monitoring und Controlling zu besitzen, auf welche ich in Form der Qualitätssicherung in diesem Abschnitt eingehen werde.¹⁰

9.1 Entwicklungsprozess

Um Planungssicherheit zu haben benötigt man einen kontinuierlichen Prozess, damit man weiß, an wen welche Aufgaben zu verteilen sind und jeder Projektteilnehmer den Vorgangsablauf kennt. In unserem Fall war die Aufgabenverteilung bereits durch Telekom anhand der Rollen Projektmanager, Konzepter, Berater und Entwickler vorgegeben, sodass wir uns auch versuchten daran zu orientieren.

Unsere Arbeitsprozesse waren wichtig für die Qualität der Entwicklung. Die effektive Ar-

¹⁰PROJEKT MANAGEMENT INSTITUTE: *A Guide to the Project Management Body of Knowledge*. Fifth Edition. New-ton Square, 2008. 589 Seiten.

beitsteilung half uns, sich auf seinen Bereich fokussieren zu können und mit entsprechender Kenntnis voranschreiten zu können. Auch wenn die Umsetzung der Prozesse nicht immer absolut problemfrei ablief, so haben wir es doch geschafft, unsere beiden Hauptfeatures Finder und Self-Service gut umsetzen zu können.

9.2 *Qualität des Designs*

Damit sichergestellt ist, dass wir ein konsistentes, qualitatives Design haben, versuchten wir uns grundlegend an den iOS7-Design-Richtlinien zu orientieren.¹¹ Da leider kein Mitglied von uns weitreichende Erfahrungen mit Design-Werkzeugen hatte, war die Umsetzung deren für uns von größerem Problem.

9.3 *Umgang mit Feedback*

Ein effektives Instrument zur Qualitätssicherung, dass uns durch die Veranstalter zur Verfügung gestellt wurde, ist das Feedback von potentiellen Kunden. Umso mehr ist es für uns daher eine große Hilfe gewesen, dass auf Probleme und Wünsche bei der App in wöchentlichen Plenarveranstaltungen eingegangen wurde. Wir haben jedes Feedback umgehend in unserer Facebook-Gruppe dokumentiert und Lösungen ausdiskutiert.

Dieser Vorgang hat uns sehr in der Einhaltung der oben geschilderten Prozesse geholfen. Da wir uns jeden Dienstag und Donnerstag im Mac-Arbeitsraum getroffen hatten, konnten wir erst neue Storys in PivotalTracker anhand des Feedbacks anlegen, diese dann unter uns aufteilen und bis zur nächsten Plenarveranstaltung abarbeiten. Man kann sagen, dass wir also die Plenarveranstaltung als „Motor“ für unsere Scrum-Iterationen benutzt haben.

Wir entwickelten somit über die Zeit ein immer besseres Gefühl für die Umsetzung von Feedback und auch über die Möglichkeit, Scrum als Methode des agilen Projektmanagements anzuwenden.

¹¹<https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/>

9.4 Ausblick

Über die vorangegangen Punkte hinaus haben wir uns außerdem überlegt, wie wir auch entwicklungstechnischer Sicht Qualitätssicherung umsetzen können. Unser nächster Schritt in der Entwicklung wäre daher auch das Einbinden von Unitests gewesen, um auch in Hinblick auf die oben beschriebene Architektur eine Absicherung über die Prozesse geben.

Programmausdruck 2 zeigt ein Beispiel für so einen Unitest, wie er die Arbeitsweise der Dependency Injection verifizieren kann. Er zeigt, wie beispielsweise das Einbinden der Authentifizierung getestet werden kann.

Programmausdruck 2: Beispiel für einen Unitest mit iOS

```
@implementation KibaTests

/**
 * Diese Methode wird vor jedem Test ausgeführt.
 */
- (void)setUp
{
    [super setUp];
    // Bootstrap der App
    [KBABootstrap bootstrap];
}

/**
 * Diese Methode wird nach jedem Test ausgeführt.
 */
- (void)tearDown
{
    [super tearDown];
}

/**
 * Testet, ob die Abhängigkeit für die
 * Authentifizierung korrekt eingebunden wurde.
 */
- (void)testAuthDependency
{
    // Ist die Abhängigkeit vorhanden?
    XCTAssert([KBADependencyInjector hasDependency:@auth],
              @"Auth dependency is missing!");

    // Handelt es sich auch um ein KBAAuth-Objekt?
    id auth = [KBADependencyInjector getByKey:@auth];
    XCTAssert([auth isKindOfClass:[KBAAuth class]],
              @"Auth dependency is not a valid Auth object!");
}

@end
```

10 Fazit von Michael Schaarschmidt

Das Projekt objektorientierte Softwareentwicklung hat für uns in gewisser Hinsicht eine Zäsur im Studium markiert. Bisherige Module oder Praktika legten den Fokus auf eine technisch möglichst saubere Implementierung, bei der im Zweifel das Interaktionsdesign oder der Funktionsumfang zurückgestellt wurden. Im Projekt wurde die Beherrschung objektorientierter Techniken bereits vorausgesetzt und erstmalig ein Software-Produkt entwickelt, das in seiner Gesamtheit auch industriellen Anforderungen genügen musste.

Mehrere Teilnehmer unserer Gruppe hatten bereits ein erstes Maß an Arbeitserfahrung vorzuweisen, etwa als Werkstudent oder freiberuflicher Webentwickler. Trotzdem wurde in den ersten Wochen des Projekts deutlich, dass im Bezug auf die Arbeitsorganisation als Auftragnehmer eines Produktes noch Raum für Verbesserungen war.

Dies galt insbesondere für den Umgang mit Designfragen und der Reihenfolge der Entwicklung. In einer Gruppe, die bis auf eine Ausnahme aus Informatikstudenten bestand, gingen wir zunächst in aus anderen Modulen vertrauter Manier vor. Um die Eigenheiten von iOS und Objective C kennenzulernen, unternahmen wir erste Gehversuche in Funktionen wie einer Überweisungsansicht, die für das Endprodukt sekundär waren. Ebenso behandelten wir in den ersten Überlegungen das Design noch etwas stiefmütterlich. Eine Mentalität, der im Informatikstudium nicht selten Vorschub geleistet wird, ließe sich auf die Aussage „Erst wird programmiert, dann das Design nachgeschoben“ reduzieren. Im Plenum wurde uns aufgezeigt, dass ein solcher Ansatz nicht nur weniger effizient ist, sondern auch das eigene Produkt schlechter dastehen lässt.

Denn einerseits ist es in einem engen Zeitrahmen, der ohnehin eher auf die Entwicklung eines Prototyps abzielt, sinnvoller, sich voll und ganz auf die Funktionen zu konzentrieren, die Alleinstellungsmerkmale darstellen und den innovativen Kern des Konzepts bilden. Denn nur anhand dieser lässt sich ein Kunde von einem Produkt überzeugen. Zudem erfordert ein benutzerzentriertes Design, das erst nachträglich hinzugefügt wird, umständliche Änderungen an der Codebasis. Gleichzeitig können bereits wenige, minimalistische Designelemente in konsistenter Farbe und Anordnung einen enormen Unterschied in der Wahrnehmung der Produktqualität bedeuten.

Entsprechend haben wir unsere Arbeitsweise adjustiert und ein neues Verständnis von zeitgemäßer Softwareentwicklung gewonnen. Während innovative Funktionalität von technischer Seite weiter geboten ist, hat sich der Anspruch der Benutzer an die Benutzbarkeit stark gewandelt; Apps, die dies nicht berücksichtigen, haben auf einem derart umkämpften Markt wenig Erfolgsaussicht.

Abkürzungsverzeichnis

DAO Data Access Object

REST Representational State Transfer

SOAP Simple Object Access Protocol

VCS Version Control System (Versionsverwaltung)

Abbildungsverzeichnis

Abb. 1:	Endfassung des Umfragebogens	12
Abb. 2:	Ergebnisse der Kontexterkundung	13
Abb. 3:	Mockup des Dashboards, der Filial-Seite und der Überweisung	15
Abb. 4:	App-Icon mit Farbskala	16
Abb. 5:	Dashboard	26
Abb. 6:	Googlemaps mit Filialmarkierungen	27
Abb. 7:	Filialseite mit Sortenanfrage Pop-Over	27
Abb. 8:	Die Authentifizierungsstati	28
Abb. 9:	Self-Service Startseite. Die App ist verbunden.	28
Abb. 10:	Die verschiedenen Dokumenttypen	29
Abb. 11:	Übersicht der Kontoauszüge	29
Abb. 12:	Eine Scheckgrafik als Formular für die Umbuchung	30
Abb. 13:	Finanzierungsrechner mit individuellen Konditionen	31
Abb. 14:	Ein minimalistischer Posteingang des Nutzers	31
Abb. 15:	Eigenkreationen für das Design	32
Abb. 16:	Entitäten-Relationen-Diagramm	35
Abb. 17:	Klassendiagramm unserer Dependency Injection	37
Abb. 18:	Stories in PivotalTracker	39
Abb. 19:	Burndown-Chart in PivotalTracker	40
Abb. 20:	Quelltext-Hosting bei GitHub	41

Abb. 21:	GitHub-Wiki für den Wissensaustausch	41
Abb. 22:	Entwerfen mit Balsamiq	42
Abb. 23:	Designen des KiBa-Icons mit Adobe Illustrator	43

Programmausdrucke

Prog. 1:	Der Bootstrapping-Vorgang	37
Prog. 2:	Beispiel für einen Unitest mit iOS	45