



# Die „KiBa“-App

## Projektbericht

von

Alexander Droste

Markus Fasselt

Marco F. Jendryczko

Konstantin S. M. Möllers

Michael Schaarschmidt

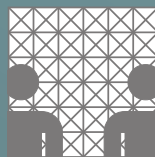
Julius Wulk

Veranstalter

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



Universität Hamburg



Fachbereich Informatik

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung und Projektkontext</b>	<b>4</b>
<b>2</b>	<b>App-Idee</b>	<b>5</b>
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

<b>3</b>	<b>Mockups</b>	<b>10</b>
<b>4</b>	<b>Vorgehen</b>	<b>11</b>
4.1	Scrum	11
<b>5</b>	<b>Design</b>	<b>12</b>
<b>6</b>	<b>Ergebnis</b>	<b>12</b>
<b>7</b>	<b>Architektur</b>	<b>12</b>
7.1	Umsetzung des MVC-Ansatzes	12
7.2	Datenmodell	12
7.3	Sicherheitsaspekte	13
7.4	Dependency Injection	14
<b>8</b>	<b>Toolchain</b>	<b>16</b>
8.1	Agiles Vorgehen im Projekt	16
8.2	Versionsverwaltung und Informationsaustausch	17
8.3	Konzept und erster Entwurf	19
8.4	Design- und Textsatztools	19
<b>9</b>	<b>Qualitätssicherung</b>	<b>19</b>
9.1	Fester Entwicklungsprozess	21
9.2	Umgang mit Feedback	21
9.3	Ausblick	21
<b>10</b>	<b>Fazit</b>	<b>22</b>
	<b>Abkürzungsverzeichnis</b>	<b>24</b>
	<b>Abbildungsverzeichnis</b>	<b>25</b>

## 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 software-technischer 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 Abschwung <sup>1</sup>. Es erscheint insofern zunächst widersinnig, ein Geschäftsmodell, welches zunehmend effizient durch Softwareprodukte 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 finden <sup>2</sup>, 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.

---

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

<sup>2</sup><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 foursquare<sup>3</sup>, 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

---

<sup>3</sup><https://de.foursquare.com/>

Banken zukünftigen prägen könnten, ergab jedoch einen ähnlichen Tenor<sup>4</sup>.

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 Sparbuch<sup>5</sup>.

---

4

<sup>5</sup>vgl. 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 erwerben<sup>6</sup>; 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.

---

<sup>6</sup><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 Geräteverlust 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 Wertpapierzeitschriften oder Bonität - per Kontaktformular abzuschicken und dann einige Tage auf die entsprechenden Ausdrücke 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 Sparbuches 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 graphischen 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 Nachrichtbereich, 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 Mockups von Julius Wulk

## 4 Vorgehen von Alexander Droste

### 4.1 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 Vervollendung 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 Burndown-Charts dann nicht besonders indikativ dafür, wie sich bisher Erledigtes zu verbleibenden Aufgaben mit Blick auf die Einarbeitungsschwierigkeiten

verhält.

## 5 Design von Julius Wulk

## 6 Ergebnis von Marco F. Jendryczko

## 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, welche Teile der Logik, der GUI oder anderer

### 7.2 Datenmodell

Zur Modellbildung der App fiel unsere Entscheidung auf ein klassisches Entitäten-Beziehungssystem. 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 1 zu sehen.

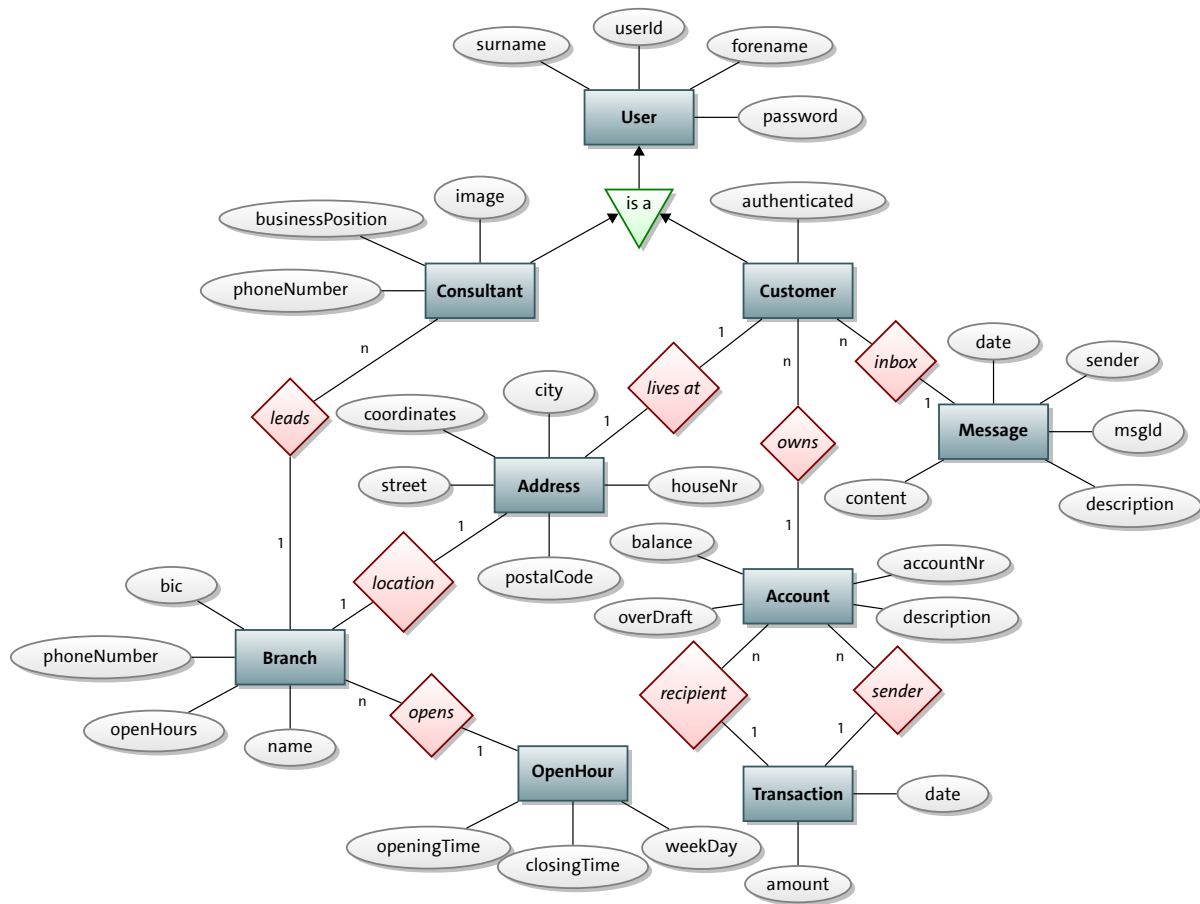


Abbildung 1: Entitäten-Relationen-Diagramm

### 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 Schlüsse, 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.

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.

Daher ist für uns Dependency Injection als Entwicklungsmuster die Lösung dieses Problems. Wir haben es in einer reduzierter Form selber versucht zu implementieren. Dabei wird an einer zentralen Stelle im Quelltext, nämlich im *Bootstrapping*, in den `KBADependencyInjector` 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 Programmausdruck 1 wiedergegeben.

Programmausdruck 1: Der Bootstrapping-Vorgang

```
+ (void) initWithDependencyInjectorWithMode:(NSString *)mode {
    id<KBABranchDao> branchDao;
    id<KBAExchangeRateDao> exchangeRateDao;
    id<KBACustomerDao> customerDao;
    id<KBAAccountDao> accountDao;
    id<KBATransactionDao> transDao;
    id<KBACreditRatingDao> 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 = [KBAAccountDaoDummy new];
    }
}
```

```

        transDao = [KBATransactionDaoDummy new];
        creditDao = [KBACreditRatingDaoDummy 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 withKey:@"branchDao"];
    [KBADependencyInjector setObject:exchangeRateDao withKey:@"rateDao"];
    [KBADependencyInjector setObject:customerDao withKey:@"customerDao"];
    [KBADependencyInjector setObject:accountDao withKey:@"accountDao"];
    [KBADependencyInjector setObject:auth withKey:@"auth"];
    [KBADependencyInjector setObject:transDao withKey:@"transDao"];
    [KBADependencyInjector setObject:creditDao withKey:@"creditDao"];
    [KBADependencyInjector setObject:messageDao withKey:@"messageDao"];
    [auth login: @"max" withPassword:@"test"];
} // end of initDependencyInjectorWithMode:

```

Wie man sehen kann, besitzt jedes Data Access Object (DAO) ein Protocol, welches auf zwei Arten instanziiert 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 zugemünzt werden kann. Dies hat für unseren Kunden den Vorteil, ihre App unter viele Kunden bringen zu können.

Wir haben festgestellt, dass sich das frühe Auseinandersetzen mit seiner 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 2 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“)<sup>7</sup> und unseren Vorhersagen für den Aufwand einer Story angegeben durch Punkte.

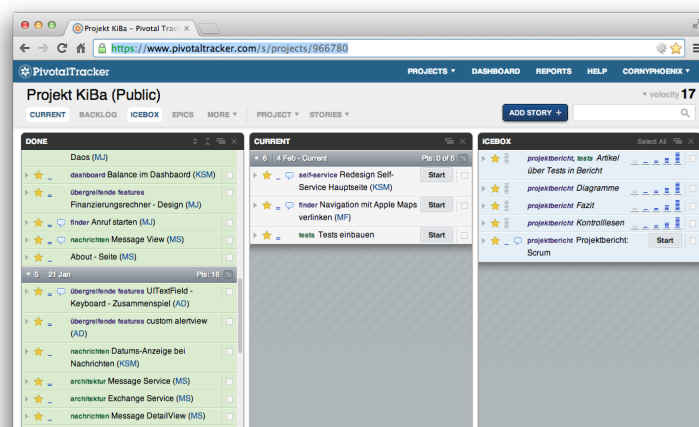


Abbildung 2: Stories in PivotalTracker

<sup>7</sup><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 3 zu sehen ist.

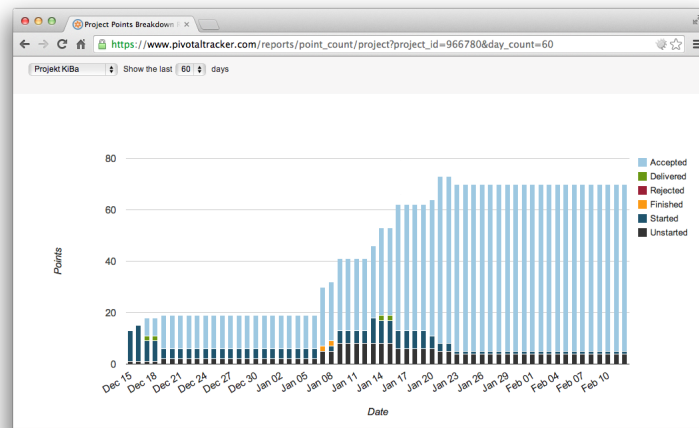


Abbildung 3: Burndown-Chart in PivotalTracker

## 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 Git<sup>8</sup> aus dem Universitäts- und Arbeitsumfeld am größten war, fiel unsere Wahl für ein VCS darauf. Entscheidend dabei war auch der größere Komfort gegenüber Subversion und die Möglichkeit, die Plattform GitHub<sup>9</sup> als Repository-Hoster und anfangs auch als Issue-Tracker zu verwenden. Wie unser Repository bei GitHub aussieht ist zu sehen in Abbildung 4.

Ein weiteres nützliches Feature von GitHub für uns war außerdem ein Wiki, dargestellt in Abbildung 5. 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.

<sup>8</sup><http://git-scm.com/>

<sup>9</sup><https://github.com/>

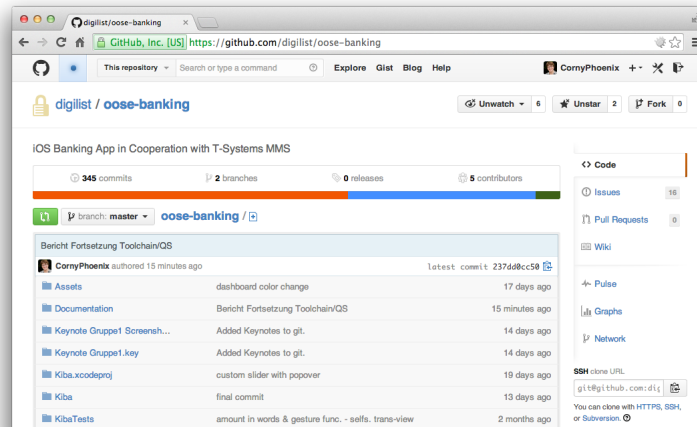


Abbildung 4: Quelltext-Hosting bei GitHub

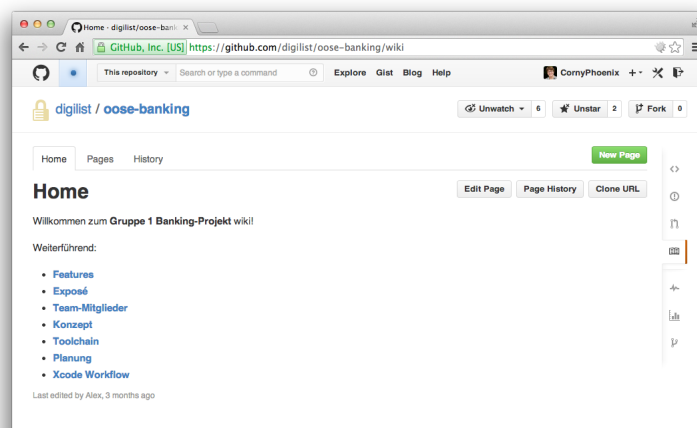


Abbildung 5: GitHub-Wiki für den Wissensaustausch

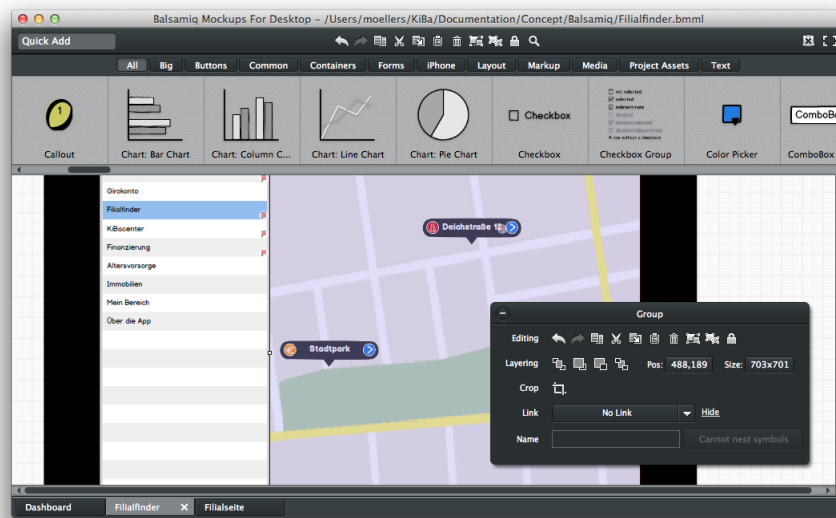


Abbildung 6: Entwerfen mit Balsamiq

### 8.3 Konzept und erster Entwurf

Balsamiq

### 8.4 Design- und Textsatztools

Lua<sub>La</sub>T<sub>E</sub>X, Adobe Illustrator/Photoshop

## 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.<sup>10</sup>

<sup>10</sup>PROJEKT MANAGEMENT INSTITUTE: *A Guide to the Project Management Body of Knowledge*. Fifth Edition. Newton Square, 2008. 589 Seiten.

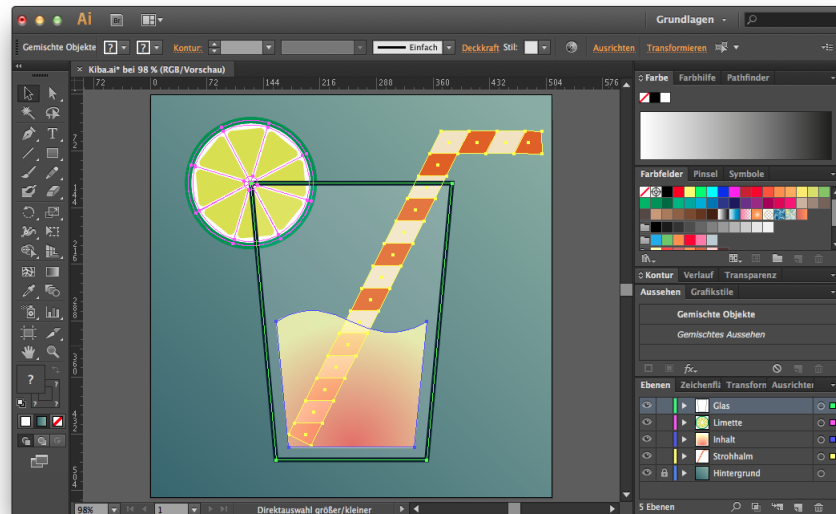


Abbildung 7: Designen des KiBa-Icons mit Adobe Illustrator

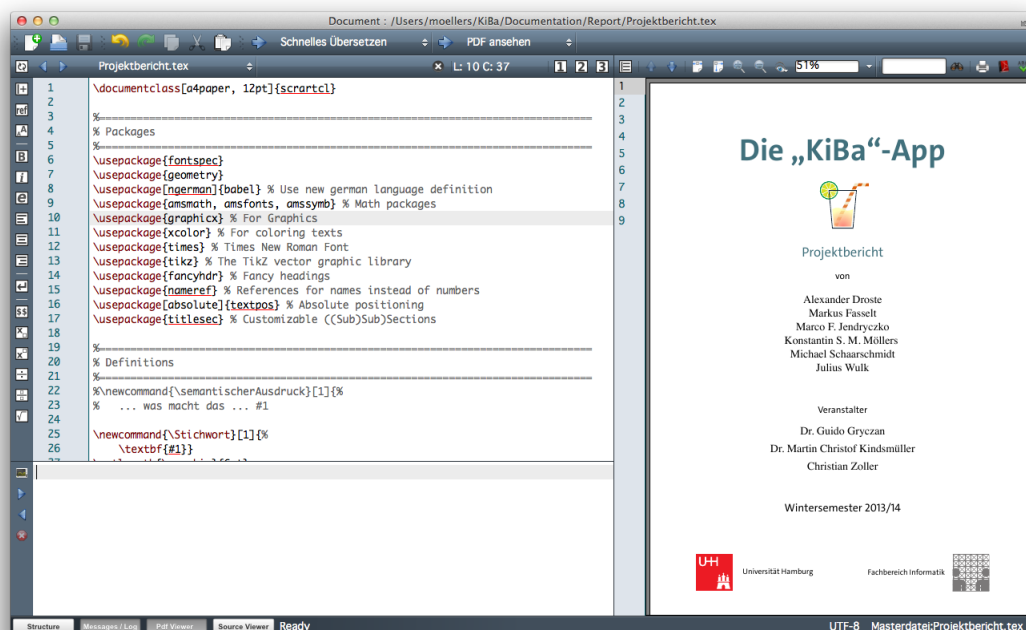


Abbildung 8: Setzen des Projektberichts mit Lua $\text{\LaTeX}$

## 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 versuchen daran zu orientieren.

Unsere Arbeitsprozesse waren wichtig für die Qualität der Entwicklung. Die effektive Arbeitsteilung 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.<sup>11</sup> 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

---

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

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.

## 9.4 Ausblick

Über die vorangegangenen 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 Unittests gewesen, um auch in Hinblick auf die oben beschriebene Architektur eine Absicherung über die Prozesse geben.

Programmausdruck 2 zeigt ein Beispiel für so einen Unittest, 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 Unittest 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:	Entitäten-Relationen-Diagramm . . . . .	13
Abb. 2:	Stories in PivotalTracker . . . . .	16
Abb. 3:	Burndown-Chart in PivotalTracker . . . . .	17
Abb. 4:	Quelltext-Hosting bei GitHub . . . . .	18
Abb. 5:	GitHub-Wiki für den Wissensaustausch . . . . .	18
Abb. 6:	Entwerfen mit Balsamiq . . . . .	19
Abb. 7:	Designen des KiBa-Icons mit Adobe Illustrator . . . . .	20
Abb. 8:	Setzen des Projektberichts mit Lua <sup>A</sup> TeX . . . . .	20