Die „KiBa“-App

Projektbericht

von

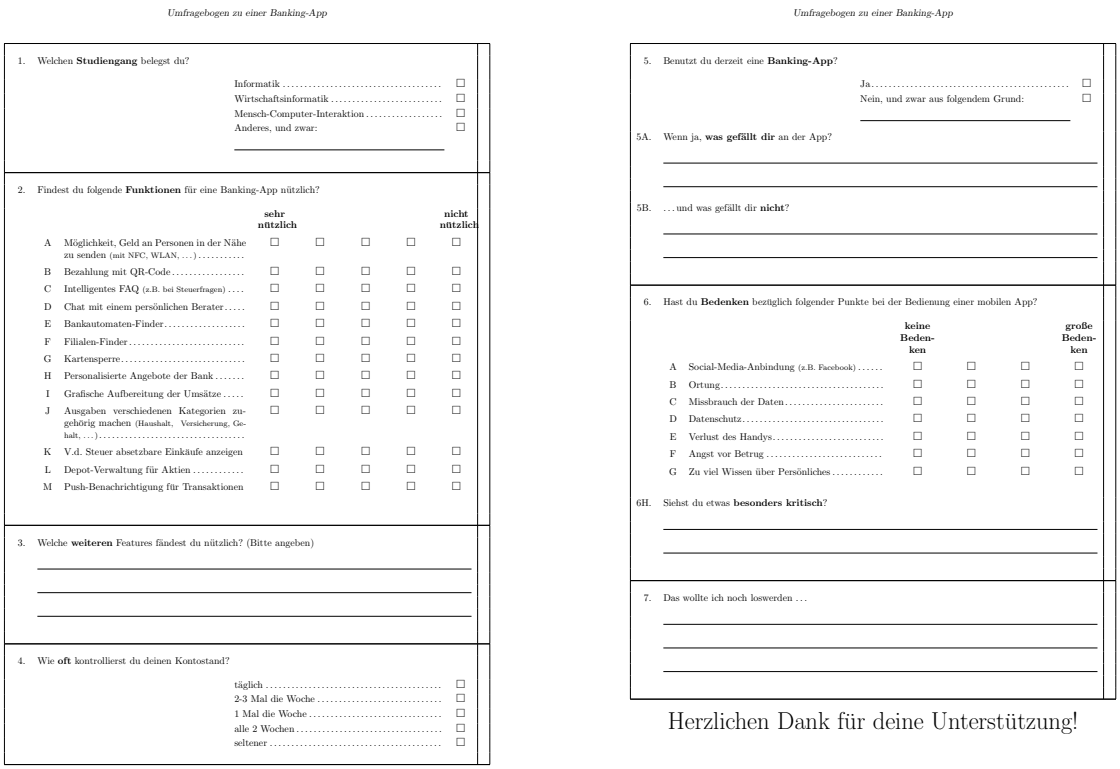
Alexander Droste
Markus Fasselt
Marco F. Jendryczko
Konstantin S. M. Möllers
Michael Schaarschmidt
Julius Wulk

Veranstalter

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

Universität Hamburg
Fachbereich Informatik

Inhaltsverzeichnis



Umfragebogen zu einer Banking-App

1. Welchen **Studiengang** belegst du?

Informatik ☐
Wirtschaftsinformatik ☐
Mensch-Computer-Interaktion ☐
Anderes, und zwar: ☐

2. Findest du folgende **Funktionen** für eine Banking-App nützlich?

		sehr nützlich				**nicht nützlich**
A	Möglichkeit, Geld an Personen in der Nähe zu senden (mit NFC, WLAN, ...)	☐	☐	☐	☐	☐
B	Bezahlung mit QR-Code	☐	☐	☐	☐	☐
C	Intelligentes FAQ (z.B. bei Steuerfragen)	☐	☐	☐	☐	☐
D	Chat mit einem persönlichen Berater	☐	☐	☐	☐	☐
E	Bankautomaten-Finder	☐	☐	☐	☐	☐
F	Filialen-Finder	☐	☐	☐	☐	☐
G	Kartensperre	☐	☐	☐	☐	☐
H	Personalisierte Angebote der Bank	☐	☐	☐	☐	☐
I	Grafische Aufbereitung der Umsätze	☐	☐	☐	☐	☐
J	Ausgaben verschiedenen Kategorien zugehörig machen (Haushalt, Versicherung, Gehalt, ...)	☐	☐	☐	☐	☐
K	V.d. Steuer absetzbare Einkäufe anzeigen	☐	☐	☐	☐	☐
L	Depot-Verwaltung für Aktien	☐	☐	☐	☐	☐
M	Push-Benachrichtigung für Transaktionen	☐	☐	☐	☐	☐

3. Welche **weiteren** Features fändest du nützlich? (Bitte angeben)

4. Wie **oft** kontrollierst du deinen Kontostand?

täglich ☐
2-3 Mal die Woche ☐
1 Mal die Woche ☐
alle 2 Wochen ☐
seltener ☐

Umfragebogen zu einer Banking-App

5. Benutzt du derzeit eine **Banking-App**?

Ja ☐
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	Social-Media-Anbindung (z.B. Facebook)	☐	☐	☐	☐
B	Ortung	☐	☐	☐	☐
C	Missbrauch der Daten	☐	☐	☐	☐
D	Datenschutz	☐	☐	☐	☐
E	Verlust des Handys	☐	☐	☐	☐
F	Angst vor Betrug	☐	☐	☐	☐
G	Zu viel Wissen über Persönliches	☐	☐	☐	☐

6H. Siehst du etwas **besonders kritisch**?

7. Das wollte ich noch loswerden ...

Herzlichen Dank für deine Unterstützung!

Abbildung 1: Umfragebogen für erste Ist-Analyse

1 Aufgabenstellung und Vision von Michael Schaarschmidt

Die unser Gruppe übertragene Aufgabe bestand darin, eine iOS-App zu entwickeln, die die Bindung zwischen einer fiktiven Filialbank und ihren Kunden erhöht. In den ersten Überlegungen wurde schnell deutlich, dass nahezu jede Dienstleistung einer Filiale auf die ein oder andere Weise von Direktbanken abgebildet werden kann. Das Alleinstellungsmerkmal besteht im direkten menschlichen Kontakt vor Ort und in der persönliche Bindung zu einem möglicherweise vertrauten Berater.

Insofern haben wir uns nach einem Prozess von abwechselnden Gruppendiskussion und externen Feedback dazu entschlossen, diejenigen Features zu betonen, die Berater und Kunden möglichst in der Filiale zusammenbringen, aber auch unabhängig davon genutzt werden können. Auch Direktbanken versuchen inzwischen, anonyme Hotlines zu vermeiden und Kunden einzelnen Beratern zuzuordnen. In Kombination mit niedrigeren Gebühren, kostenlosen Kredit-

karten und einfacher Kontoführung ist dies ein schwer zu schlagendes Angebot. Eine App, die Neukunden für eine Filialbank gewinnen soll, muss also einerseits eine ähnlich einfache Kontoverwaltung bieten, außerdem aber noch einen Mehrwert schaffen, der die höheren Gebühren einer Filialbank rechtfertigt.

Ein weiterer, für die Akzeptanz einer App wichtiger Aspekt ist die (vielleicht auch nur vom Kunden gefühlte) Sicherheit. Eine von uns durchgeführte Umfrage unter Studenten technischer Fächer, primär Informatik, ergab, dass die meisten Teilnehmer unabhängig von der Funktionalität schon aus Sicherheitsgründen keine Bankgeschäfte mit einer App erledigen wollten. Die besondere Sensibilität der genannten Zielgruppe für Sicherheitsaspekte darf nicht außer Acht gelassen werden; dennoch ergab sich für uns die Frage, ob hier durch Interaktion in der Filiale nicht zugleich Kundenbindung und Vertrauen in die Sicherheit der Anwendung erhöht werden könnten.

Insgesamt entstand also die Vorstellung einer Anwendung, die klassische Bankgeschäfte wie Überweisungen beherrscht, gleichzeitig aber versucht, Lokalität und Interaktion mit dem eigenen Berater herzustellen.

2 Kontext und App-Idee von Michael Schaarschmidt

2.1 Name

Der Name der App, „**K**i**B**a“, 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.

2.2 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, je-

denfalls 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. Die Vorstellung, ein Kreditangebot auf dem Bildschirm eines iPhones durchzusprechen, ist hingegen eher absurd.

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.

2.3 Funktionen

Die Kernfunktionen der App sollen sich auf genau die Bereiche konzentrieren, die einer Direktbank nicht zur Verfügung stehen und somit nicht trivial nachgebaut werden können. Dabei haben wir im Zuge der Plenumsdiskussionen und anschließenden internen Debatten insbesondere den zeitlichen Aspekt als zentrale Komponente identifziert. 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) 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 einen Multifunktionsdrucker und per App können verschiedenste Bescheinigungen ausgedruckt werden.

Ein wesentliches Produkt von Filialbanken ist das klassische Sparbuch. Umbuchungen 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 ganztägig zugänglich. Auf diese Weise bleibt einerseits das Sparbuch als vertrauenswürdige Marke erhalten, die in der Wahrnehmung misstrauischer Benutzer von den Verwerfungen der Online-Kriminalität unberührt bleibt, andererseits ist die Verfügbarkeit erheblich verbessert.

Ebenfalls auf die Bereitstellung von Service-Dienstleistungen zielt der interaktive Filialfinder ab. Aus der Karte heraus sollen Anfragen an eine bestimmte Filiale in der Umgebung er möglicht werden, indem etwa ein Beratungstermin reserviert wird und dann ohne Anstehen erfolgen kann. Eine Funktionalität dieser Art ist

Als Startbildschirm für die App ist 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. Ist der Benutzer nicht eingeloggt, erscheint an dieser Stelle ein Mockup mit der Aufforderung, sich einzuloggen.

Das Kerngeschäft 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 möhten 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.

Eng zusammenhängend mit dem Finanzierungsprofil steht die Aktivierung („Authentifikation“) der App in einer Filiale. Ohne eine solche Aktivierung 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. Ähnlich einer Kreditkarte soll dann bei Gerätverlust auch die App selbst jederzeit gesperrt werden können. Insbesondere dient die Aktivierung aber auch da-

zu, den Kunden in einem Beratungsgespräch besser kennenzulernen und in einer bankseitigen Datenbank einen persönlichen Ansprechpartner festzuhalten. Aus der App kann dann direkt ein Termin mit dem eingetragenen Berater vereinbart werden. Auch ein Nachrichten-System ist denkbar, um einzelne Fragen direkt zu klären.

3 Mockups von Julius Wulk

4 Vorgehen von Alexander Droste

4.1 Scrum

Im Rahmen des Projekts wurden wir mit der Srum-Methodik vertraut gemacht und hatten Gelegenheit,eine einführende Zertifizierung zu absolvieren. Während der erste 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 Gegegebenheiten 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 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-Dummyßu 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 2 zu sehen.

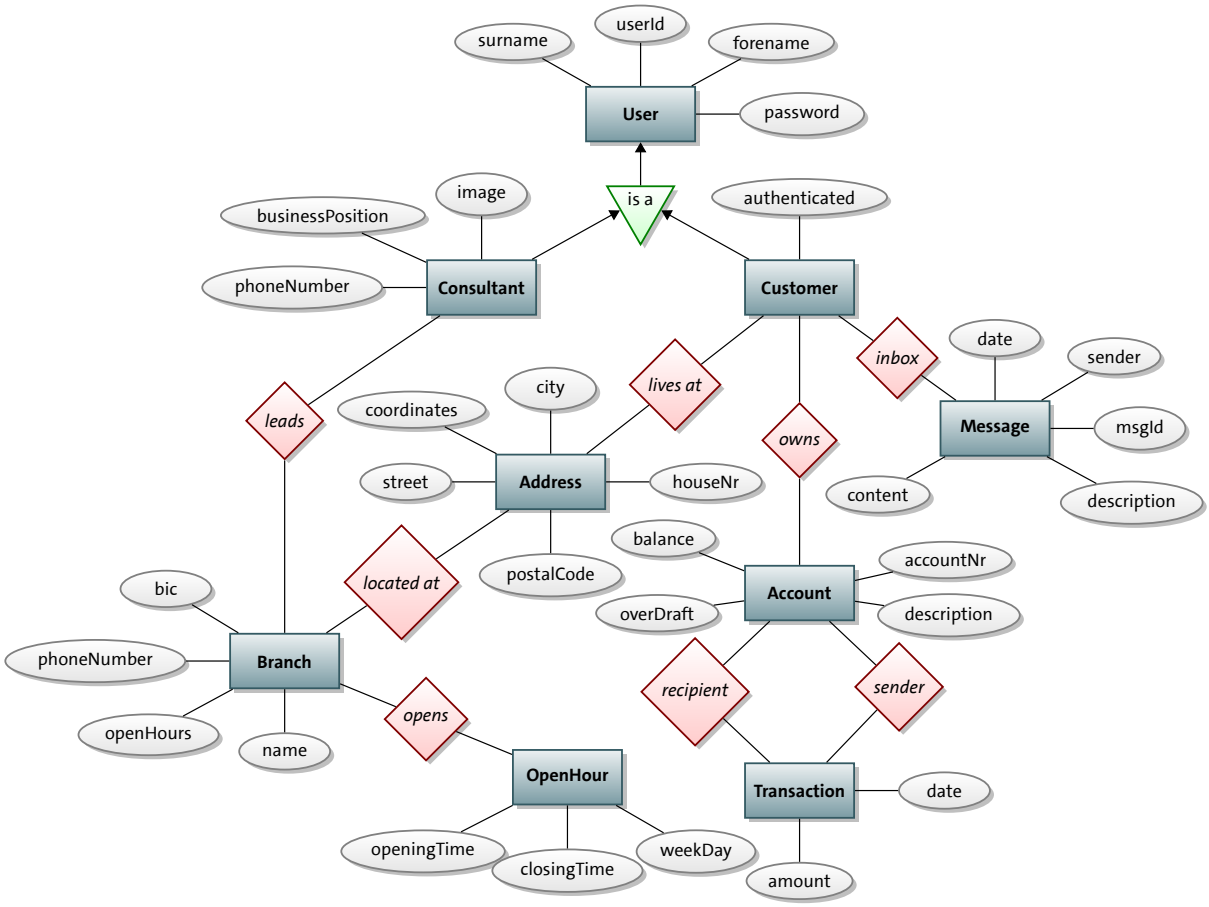


Abbildung 2: Entitäten-Relationen-Diagramm

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)initDependencyInjectorWithMode:(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

PivotalTracker + Scrum

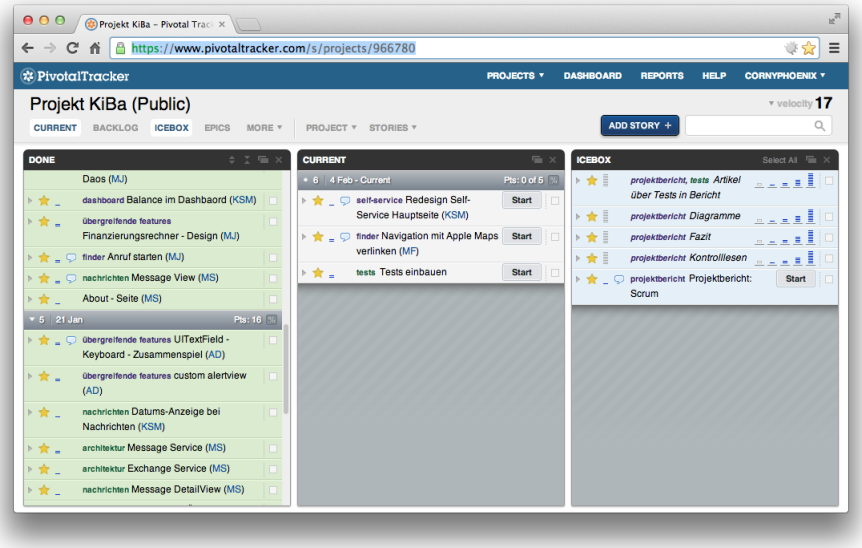


Abbildung 3: Stories in PivotalTracker

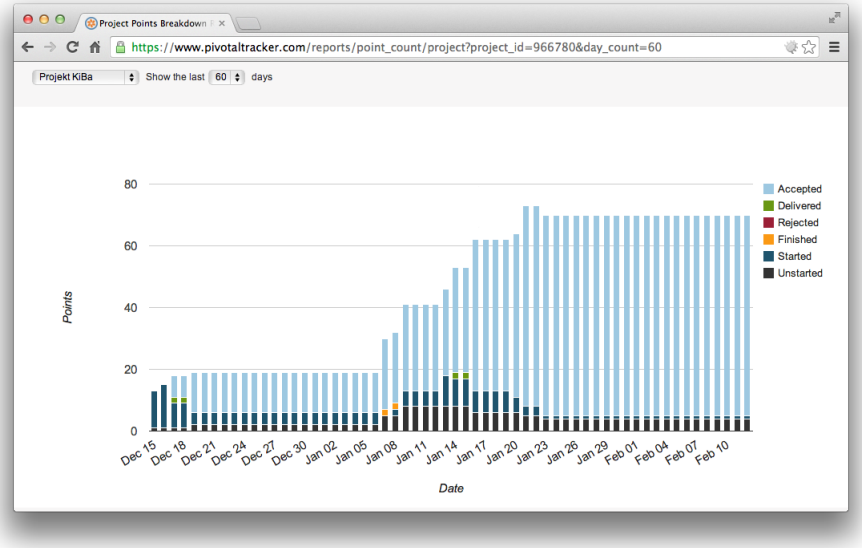


Abbildung 4: Burndown-Chart in PivotalTracker

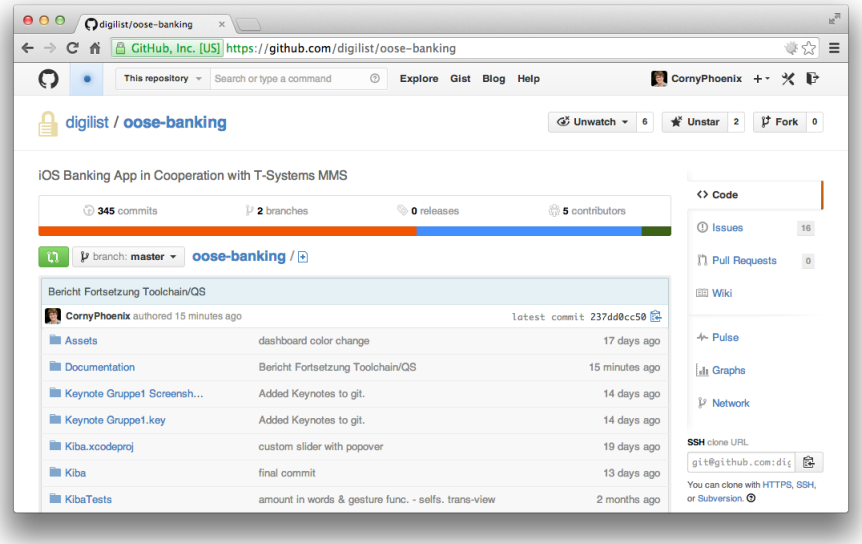


Abbildung 5: Quelltext-Hosting bei GitHub

8.2 Quelltextversionierung

GitHub + Git

8.3 Konzept und erster Entwurf

Balsamiq

8.4 Design- und Textsatztools

LuaLaTeX, Adobe Illustrator/Photoshop

9 Qualitätssicherung von Konstantin S. M. Möllers

- Gestaltungs-/Entwicklungsprozess (Mock-ups → Design → GUI-Bau → Logik)
- Design und iOS7-Guidelines
- Weeklies
- Feature-Improvement nach Plenarveranstaltungen

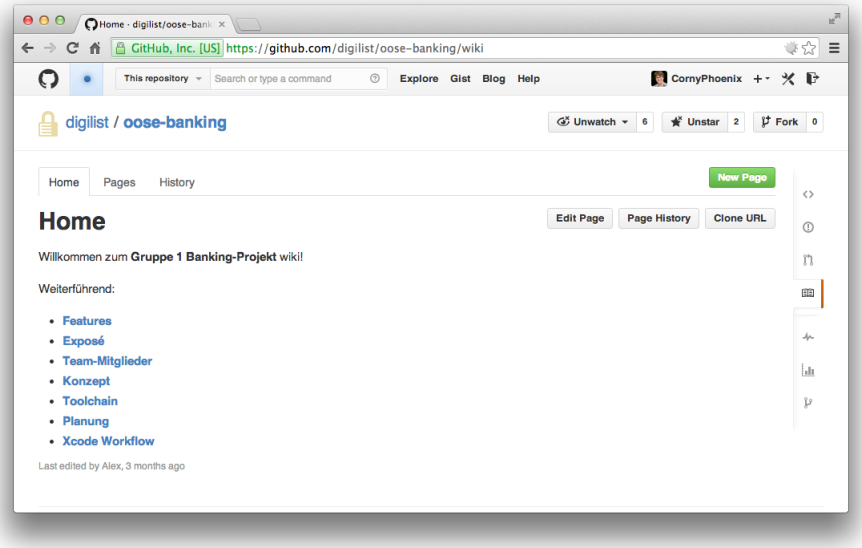


Abbildung 6: GitHub-Wiki für den Wissensaustausch

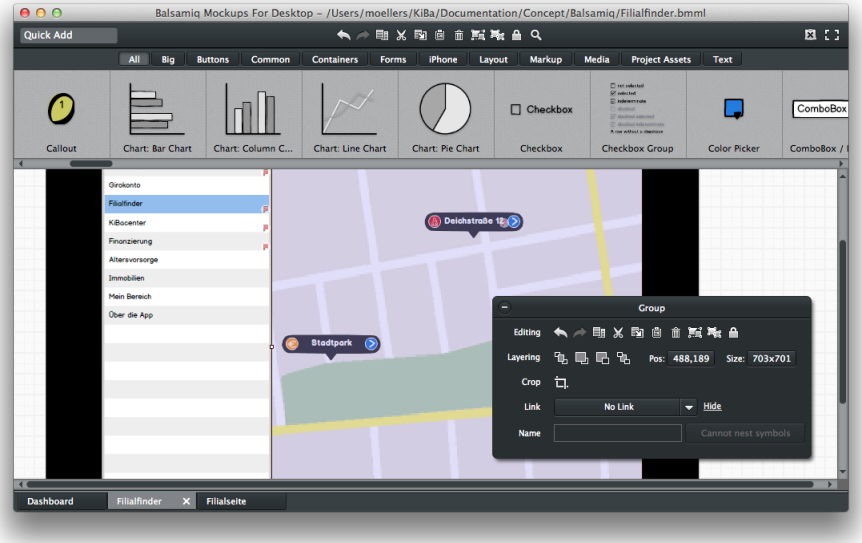


Abbildung 7: Entwerfen mit Balsamiq

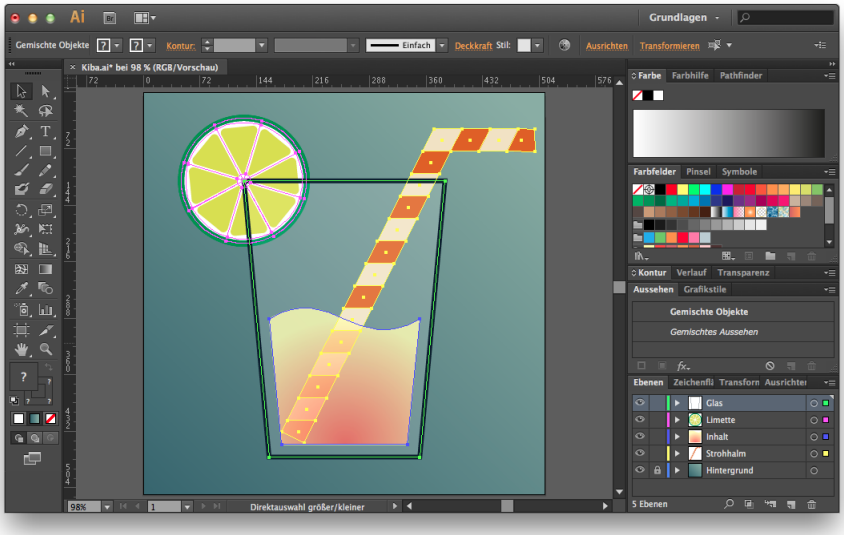


Abbildung 8: Designen des KiBa-Icons mit Adobe Illustrator

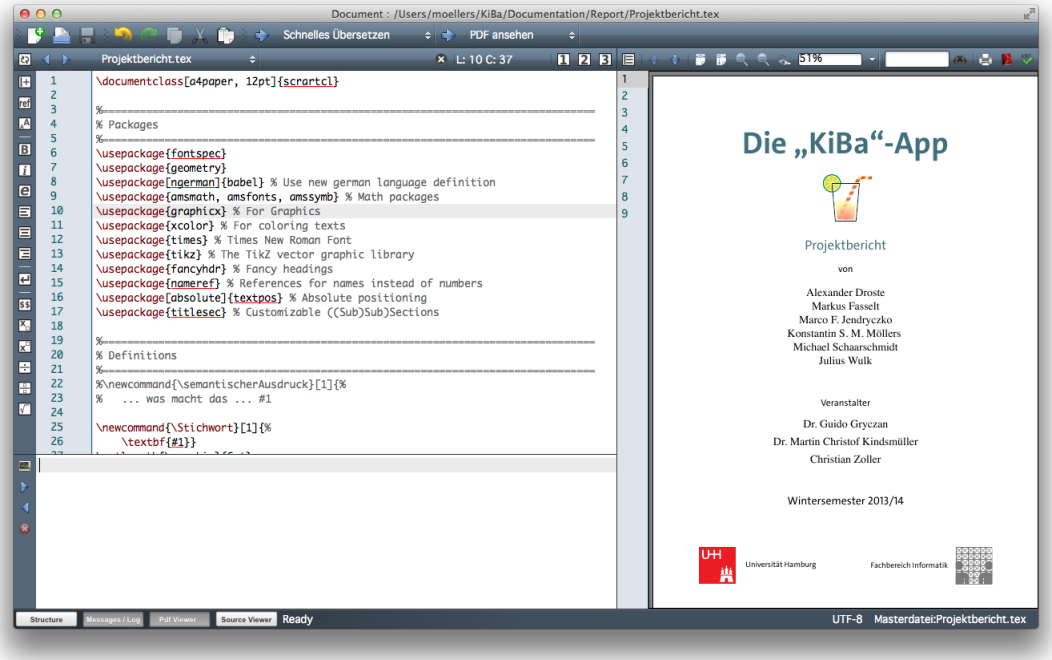


Abbildung 9: Setzen des Projektberichts mit LuaLaTeX

- Ausblick auf Unit Tests
- Club Mate

10 Fazit von Michael Schaarschmidt

Abkürzungsverzeichnis

DAO Data Access Object

REST Representational State Transfer

SOAP Simple Object Access Protocol

Abbildungsverzeichnis