

ATM Dokumentation

Die Panzerknacker

Die Panzerknacker

© 2022 Die Panzerknacker

Inhaltsverzeichnis

1. ATM Dokumentation Startseite	4
1.1 Abstract	4
1.2 Das Team	4
1.3 Die Auftraggeber	4
2. Anforderungsdokumentation	6
2.1 Produktvision und Produktziele	6
2.2 Rollen und Personas	6
2.3 User Stories	10
2.4 Aufgaben	11
2.5 Begriffslexikon	11
2.6 Mengengerüst	12
2.7 Use Cases	12
3. Architekturdokumentation	14
3.1 Beschreibung der Systemarchitektur	14
3.2 Systementwurf	15
3.3 Mensch-Maschine-Schnittstelle	16
4. Testdokumentation	27
5. Abnahmedokumentation	30
5.1 System Under Test	30
5.2 Bereitstellung zur Abnahme	30
6. Benutzerdokumentation	0
6.1 Allgemeine Funktionen	0
6.2 Benutzerspezifische Funktionen	0
6.3 Kunde	0
6.4 Geld abheben	0
6.5 Geld einzahlen	0
6.6 Kontostand anzeigen	0
6.7 Logout	0
6.8 Administrator	0
6.9 Account hinzufügen	0
6.10 Account löschen	0
6.11 Account bearbeiten	0
7. Projektdokumentation	0
7.1 Lessons Learned	0
7.2 Mapping zu individuellen Leistungen	0

8. Codedokumentation	0
8.1 Code Ist-Dokumentation	0
8.2 Delta-Dokumentation	0



1. ATM Dokumentation Startseite

1.1 Abstract

Das Ziel dieses Projekts war es, den [bestehenden Java-Code](#) eines Geldautomaten zu dokumentieren und zu verbessern. [Das Team](#) bestand aus fünf Personen, die sich auf ein bestimmtes Thema spezialisiert hatten. Ein [GitHub-Repository](#) diente als zentraler Ort für die Dokumentation und den Programmcode. Zusätzlich konnten mit Hilfe eines [SCRUM-Boards](#) die Aufgaben der Teammitglieder aufgeteilt und deren Fortschritt überwacht werden.

Das Team beschloss zu Beginn des Projekts, den bestehenden Code zu verwerfen und noch einmal von vorne zu beginnen. Dies führte zu einer neuen Architektur, die nun modularer gestaltet und dadurch leichter erweiterbar war. Details können in der [Architekturdokumentation](#) nachgelesen werden.

Die Dokumentation wurde in Form einer Website realisiert, die unter der Adresse atm.node5.de eingesehen werden kann. Zusätzlich steht eine automatisch generierte [PDF-Datei](#) zum Download zur Verfügung.

1.2 Das Team

Wir sind die Panzerknacker.

Mitglied	Spezialisierung
Michel Franz	UX
Juri Kaemper	Text & QS
Christian Lopéz	Programmierung
Felix Möhler	Requirements Engineering
Julian Thiele	UML/Kollab.-Werkzeug, Entwicklungsumgebung

1.3 Die Auftraggeber

Katharina Franz
katharina.franz@th-ab.de

Prof. Dr. Timea Illes-Seifer
timea.illes-seifert@th-ab.de

Technische-Hochschule Aschaffenburg
Würzburger Straße 45, 63743 Aschaffenburg, Deutschland



2. Anforderungsdokumentation

2.1 Produktvision und Produktziele

2.1.1 Produktvision

Eine regionale Bank hat unser externes Software-Entwicklerteam für einen Auftrag eingestellt. Bei dem uns übertragenem Projekt handelt es sich um die fehlerhafte Software einer ATM (Automated Teller Machine) zu deutsch Bankautomat. Der bereits existente Programmcode wurde von einem externen Unternehmen entwickelt, so dass der Kunde kein Expertenwissen zum Programm verfügt. Außerdem fehlt auch die Dokumentation vollständig.

Um dem Bankunternehmen nun die Verwendung des Systems zu ermöglichen, muss das Programm komplett überarbeitet werden, darüber hinaus soll eine detaillierte Dokumentation (vollständig in deutsch) für die Bank erstellt werden. Das fehlerfreie Programm mit den bereits integrierten Features und einer strukturierten Dokumentation ist unser Basisfaktor. Das Programm ist für die Bankautomaten der Bank in Deutschland vorgesehen. Die Dokumentation soll die Entwicklung sowie die Funktionen der Software zusammenfassen und für den zuständigen Mitarbeiter verständlich machen.

2.1.2 Produktziele

Die Aufgabe unseres Teams ist es, den bereits vorhandenen Code so zu überarbeiten, dass dieser voll funktionsfähig ist und eine sichere Laufzeit gewährleistet werden kann. Zur Entwicklung der Software ist eine vollständig deutsche Dokumentation vorgesehen mit **Anforderungs-, Architektur-, Test-, Abnahme-, Benutzer-, Projekt-, und Codedokumentation.**

2.2 Rollen und Personas

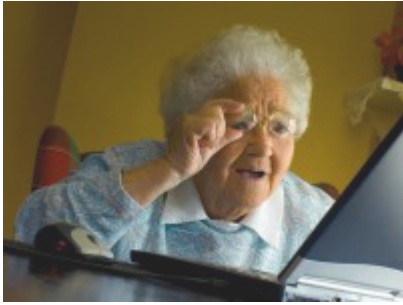
2.2.1 Rollen

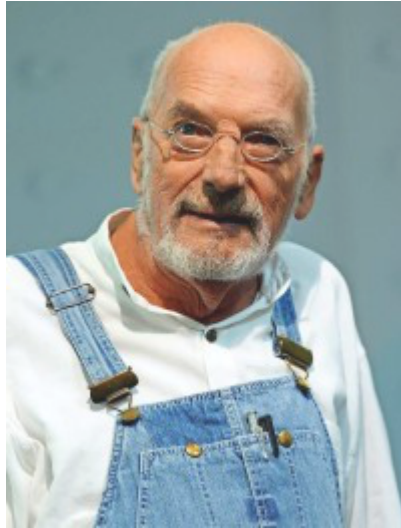
Hier werden die Rollen beschrieben, denen ein Benutzer angehören kann.

Rollen	Beschreibung
Benutzer	Die Benutzer sind Kunden der Bank, die den Geldautomaten zur Verfügung stellt
Administrator	Administratoren des Bankautomatensystems, die Verwaltungsrechte über alle Benutzer besitzen

2.2.2 Personas

Personas veranschaulichen typische Vertreter Ihrer Zielgruppe.

Gertrude Gabel	
	
Rolle	Benutzer
Alter	65
Geschlecht	weiblich
Tätigkeit	Rentnerin
Familienstand	verheiratet
Bildung	Mittelschule
Computerkenntnisse	Keine
Interessen und Hobbys	Wandern, Kaffee trinken
Einstellung zum Produkt	"Eine tolle Maschine, tut was sie soll"
Wünsche	Einfache Bedienung, wenig zum Merken

Peter Lustig

Rolle	Benutzer
Alter	38
Geschlecht	männlich
Tätigkeit	Handwerker
Familienstand	verheiratet
Bildung	Realschule
Computerkenntnisse	Grundkenntnisse
Interessen und Hobbys	Autos, Actionfilme, Fahrradfahren
Einstellung zum Produkt	"Hoffentlich werden die neuen Geldautomaten besser"
Wünsche	Nützliche Funktionen, Schnelle Bedienbarkeit

Andy Auman

Rolle	Administrator
Alter	29
Geschlecht	männlich
Tätigkeit	Systemadministrator
Familienstand	ledig
Bildung	Abitur
Computerkenntnisse	Fachkenntnisse
Interessen und Hobbys	Programmierung, Netzwerke, Gaming
Einstellung zum Produkt	""
Wünsche	Viele Funktionen, Wenig Konfigurationsaufwand

Mathias Jung

Rolle	Benutzer
Alter	19
Geschlecht	männlich
Tätigkeit	Student
Familienstand	ledig
Bildung	Abitur
Computerkenntnisse	Grundkenntnisse
Interessen und Hobbys	BWL / Wirtschaft
Einstellung zum Produkt	""
Wünsche	Schnelle und einfache Transaktionen

2.3 User Stories

User Stories sind Wünsche an eine Software, die aus Sicht des Endbenutzers verfasst wurden.

Als **[Rolle]** möchte ich **[Ziel/Wunsch]**, um **[Nutzen]**

1. Als **Benutzer** möchte ich **verschiedene Geldbeträge eingeben**, um diese abzuheben
2. Als **Benutzer** möchte ich **sehen, wie viel Geld auf meinem Konto** ist, um zu wissen, wie viel ich noch abheben kann
3. Als **Benutzer** möchte ich einen **maximal Debit Betrag pro Tag festlegen** können, um bei Diebstahl den Verlust zu minimieren
4. Als **Benutzer** möchte ich eine **vierstellige Pin zu meiner Karte eingeben** müssen, um Gelddiebstahl von meinem Konto zu vermeiden
5. Als **Benutzer** möchte ich die **Ziffern meiner Pin ändern** können, um sie mir besser merken zu können
6. Als **Benutzer** möchte ich die **Länge meiner Pin ändern** können, um die Sicherheit zu verbessern
7. Als **Benutzer** möchte ich eine **Stückelung auswählen** können, um gewünschte Scheine zu erhalten
8. Als **Benutzer** möchte ich mich **in mein Konto einloggen** können, um getätigte Transaktionen zu sehen
9. Als **Mitglied einer anderen Bank** möchte ich **gegen Gebühren Geld abheben** können, um örtlich flexibel zu sein
10. Als **Administrator** der Bank möchte ich eine **vollständige und detaillierte Dokumentation**, um im Fehlerfall schnell handeln zu können

2.4 Aufgaben

Auflistung aller Aufgaben dieses Projektes.

- Anfertigen einer Ist-Dokumentation des Codes
- Funktionen aus User Stories implementieren
- Codeverbesserungen in Delta-Dokumentation beschreiben
- Anfertigen einer Anforderungsdokumentation
- Anfertigen einer Systemdokumentation
- Anfertigen einer Testdokumentation
- Anfertigen einer Abnahmedokumentation
- Anfertigen einer Benutzerdokumentation
- Anfertigen einer Projektdokumentation

2.5 Begriffslexikon

Hier werden wichtige fachspezifische Begriffe aufgelistet, die in diesem Projekt verwendet werden.

Begriff	Bedeutung
Cash Dispenser	Bargeld im ATM-Dispenser
Deposit Slot	Geldfach zum Ein- und Auszahlen
Balance	Ist-Saldo auf einem Account
Withdrawal	Geld abheben
Account Pin	Geheimpin eines Accounts (unique)
Account number	Nummer eines Accounts (unique)
Credit	Gutschrift
Debit	Lastschrift
UI	Benutzeroberfläche
GUI	Grafische Benutzeroberfläche
JUnit	Java Bibliothek zum Testen
JSwing	Grafisches Toolkit für Java
ATM	Geldautomat (Automated Teller Machine)
Mockup	Digitales Modell einer Anwendung

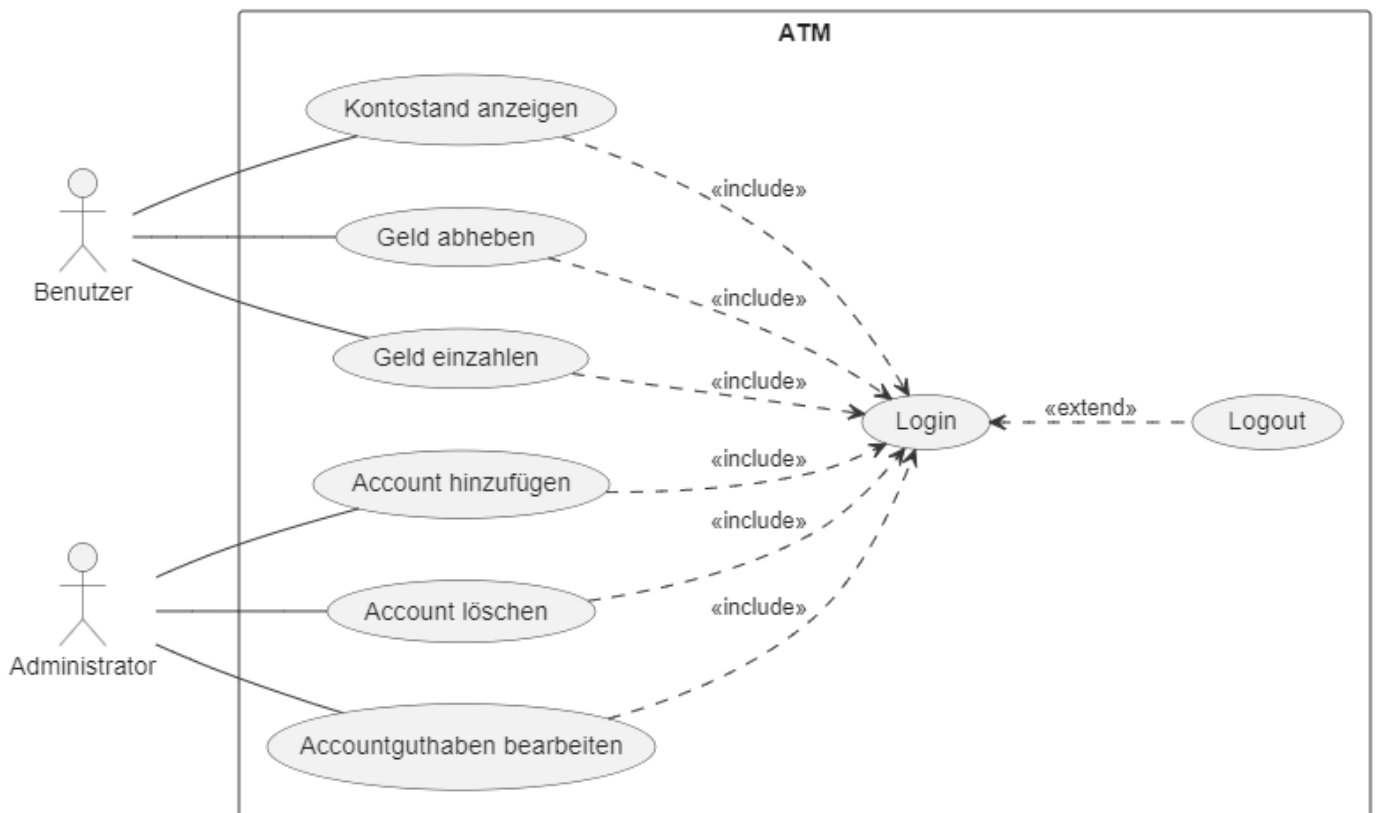
2.6 Mengengerüst

Das Mengengerüst beschreibt quantitativ die Komponenten eines Projektes.

Bezeichnung	Beschreibung	Menge	Einheit
Pin	Stellenanzahl der Pin	4	Stellen
Geldautomaten	Anzahl Geldautomaten in Aschaffenburg	43	Stück
Debit	Maximale Auszahlung pro Tag	1000	Euro
Nutzer	Maximale Nutzer gleichzeitig	1	Person
Nutzer	Maximal registrierte Nutzer	>1000	Person
Transaktion	Maximale Transaktion pro Minute	~100	Transaktion

2.7 Use Cases

In diesem Anwendungsfalldiagramm wird das nach außen sichtbare Verhalten des Systems aus Sicht der Nutzer beschrieben.



2.7.1 Use-Case Beschreibung

Name	ATM - Anwendungsfälle des Geldautomaten
Kurzbeschreibung	Dieser Use-Case beschreibt alle Funktionen, die ein Benutzer oder ein Administrator an dem Geldautomaten tätigen kann.
Akteure	Benutzer, Administrator
Vorbedingungen	Der Benutzer hat ein Konto bei der Bank.
Auslösendes Ereignis	Der Benutzer möchte eine Funktion des Geldautomaten benutzen. Der Administrator möchte Änderungen vornehmen
Hauptszenario	<ol style="list-style-type: none"> 1. Der Benutzer meldet sich an. 2. Das System zeigt die möglichen Funktionen an. 3. Der Benutzer wählt eine Funktion. 4. Die Funktion wird ausgeführt. 5. Der Benutzer meldet sich ab.
Alternativszenarien	<ol style="list-style-type: none"> 1. Ein Administrator meldet sich an. 2. Das System zeigt die Admin-Ansicht mit den Funktionen für den Administrator an. 3. Der Administrator wählt eine Funktion. 4. Die Funktion wird ausgeführt. 5. Der Administrator schließt die Admin-Ansicht und meldet sich ab.
Nachbedingungen	Hauptszenario: Funktion des Benutzers wurde erledigt und der Benutzer hat sich abgemeldet. Alternativszenario: Funktion des Administrators wurde erledigt und der Administrator hat sich abgemeldet.

3. Architekturdokumentation

3.1 Beschreibung der Systemarchitektur

3.1.1 Priorisierung der nicht funktionalen Anforderungen

Nichtfunktionale Anforderungen werden vielfach als Randbedingungen und Qualitätseigenschaften verstanden.

Qualitätsanforderungen

Änderbarkeit und **Wiederverwendbarkeit** waren uns besonders wichtig, da wir zu Beginn Schwierigkeiten hatten, uns einen Überblick über den bestehenden Code zu verschaffen. Aus diesem Grund entschieden wir uns, den Code noch einmal von Grund auf neu zu erstellen. Dadurch verbessert sich vor allem die **Brauchbarkeit** und **Wartbarkeit** des Codes.

Anforderungen an Lieferbestandteile

Eine vollständige Dokumentation in Form eines PDF Dokumentes und die Software bilden die Lieferbestandteile.

Anforderungen an die Benutzerschnittstelle

Eine weitere wichtige nicht funktionale Anforderung ist die **Bedienbarkeit** oder **Benutzerfreundlichkeit** des Programms. Da diese Anwendung für eine sehr große Menge an Benutzern ausgelegt ist, wurde die Bedienbarkeit und Benutzerfreundlichkeit des Programms auf eine höhere Priorität gesetzt. So wird gewährleistet, dass Benutzer aller Altersgruppen gut mit der Anwendung interagieren können.

3.1.2 Architekturprinzipien

Nach welchen Kriterien soll das System in Komponenten unterteilt werden? Wie sollen Komponenten strukturiert und verfeinert werden?

Das System wurde in verschiedene Komponenten unterteilt, die sich jeweils auf eine bestimmte Aufgabe beziehen, um eine enge Kopplung der Module untereinander zu reduzieren. Der verschachtelte Aufbau der UI Komponenten bildet eine Struktur, die leicht erweitert werden kann.

Welche Aspekte sollen in Komponenten zusammengefasst werden?

In der `ATM.java` Klasse werden die Änderungen von einem Modus in den Nächsten behandelt. Dem entsprechend wird die `Screen.java` Klasse angesteuert, um die UI Elemente zu aktualisieren.

Die Klasse `Screen.java` beinhaltet alle Funktionen, die zum Ändern der UI Elemente benötigt werden. In ihr werden die Klassen `Keypad.java` und `SidePanel.java` verwendet.

Welche Dienstleistungen sollen Komponenten nach außen an ihrer Schnittstelle anbieten? Wie sollen die Komponenten miteinander interagieren?

Die Komponente `Keypad.java` gibt über das `KeypadListener.java` Interface alle Events für Tastendrucke an die `Screen.java` Klasse weiter. Die Komponente `Screen.java` gibt über das Interface `ATMListener.java` Events wie z.B. einen Modus-Wechsel oder das Betätigen der Enter-Taste an die `ATM.java` Klasse weiter.

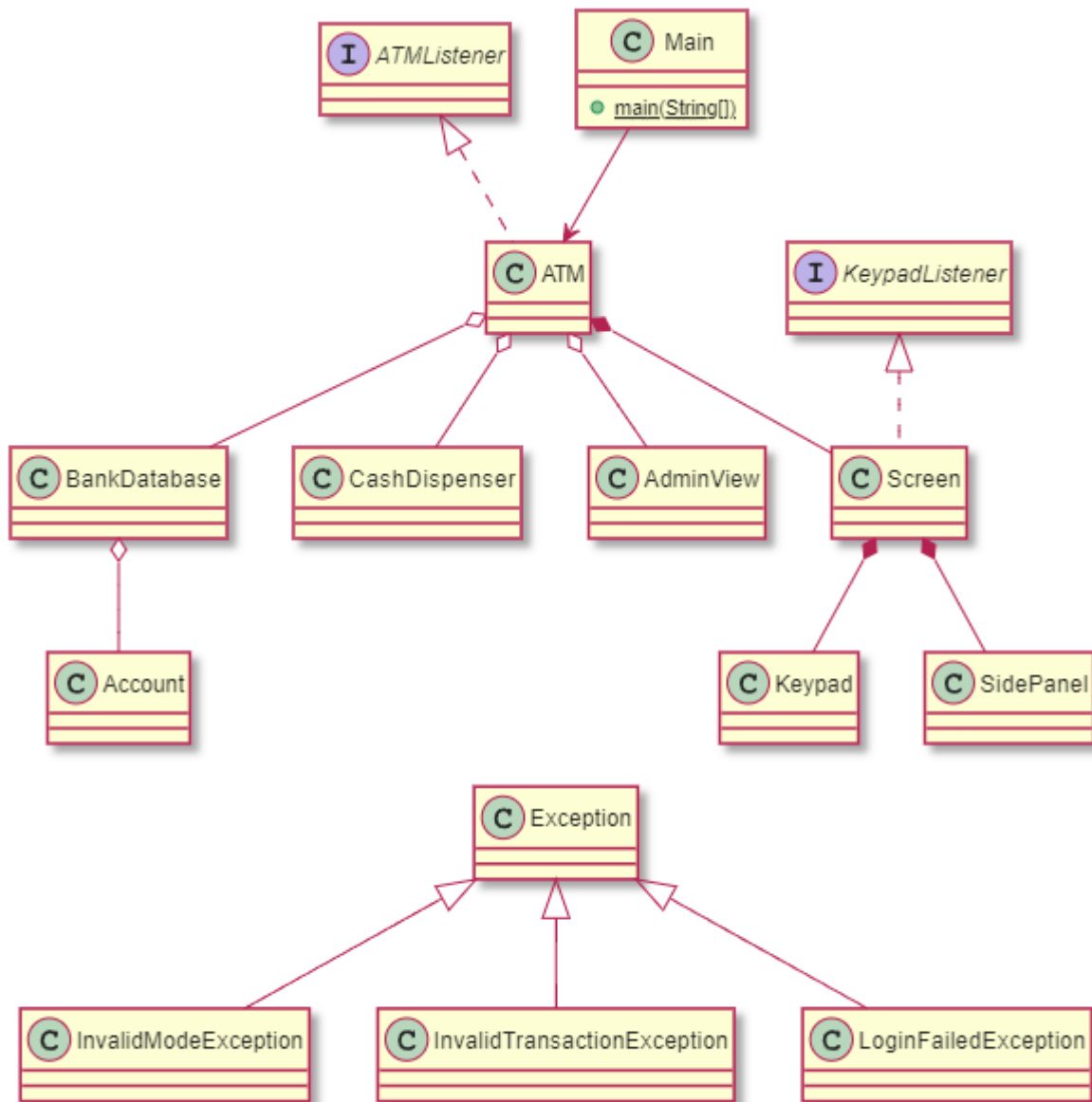
3.1.3 Schnittstellen

Hier werden alle Schnittstellen des Systems beschrieben.

- UI mit den Java-Swing GUI Bibliotheken
- `KeypadListener.java` für Kommunikationsschnittstelle zwischen dem Tastenfeld und dem Bildschirm Objekt
- `ATMListener.java` ist die Schnittstelle zum Haupt-ATM-Objekt, in der Aktionen, wie ein Wechsel in einen anderen Modus oder das Betätigen der Enter-Taste behandelt werden

3.1.4 Big Picture der Systemarchitektur

Der Aufbau der Systemarchitektur ist weitestgehend modular gestaltet und ist hier in einem Klassendiagramm dargestellt.



3.2 Systementwurf

3.2.1 Systemdekomposition

Im folgenden Abschnitt werden die einzelnen Komponenten des Systems und ihre Funktionen beschrieben.

Das System lässt sich hauptsächlich durch die Bestandteile Guthaben anzeigen, Geld abheben und Geld einzahlen beschreiben. Zusätzlich gibt es ein Menü, eine Admin-Ansicht und eine Login, sowie eine Logout Funktion.

Vom Menü aus, ist es einem Benutzer möglich alle relevanten Funktionalitäten durch das Drücken einer Zahl zu erreichen. Die Funktion `atmSwitchModeAction()` wechselt nun, je nach eingegebener Zahl, in den entsprechenden Modus. Eine weitere wichtige Komponente des Systems ist das **Keypad**, welches die verschiedenen Knöpfe darstellt. Dieses befindet sich immer in der linken Hälfte des Fensters und hilft dem Nutzer bei der Bedienung des Automaten. Es wird in dem Konstruktor der Klasse `Screen.java` zusammen mit dem `SidePanel` initialisiert.

Das `SidePanel` hat, wie das Keypad, eine eigene Klasse. Es befindet sich auf der rechten Hälfte des Fensters und beinhaltet unter anderem einen „Back-Button“. Mit diesem kann zurück in den „Menü-Modus“ gewechselt werden. In dem `SidePanel` befindet sich außerdem das Textfeld, in welchem die Benutzereingabe angezeigt wird, sowie ein `JLabel`. Dieses zeigt, je nach Modus, zum Beispiel das verfügbare Geld, oder die verschiedenen Optionen mit entsprechender Eingabe an.

Eine weitere Funktionalität ist die `Admin-Ansicht`. Loggt sich ein Admin ein, öffnet sich ein neues Fenster. In diesem können die Daten der Benutzer geändert und anschließend gespeichert werden.

3.2.2 Designalternativen und -Entscheidungen

Es wurde sich dazu entschieden die einzelnen Funktionalitäten mit Hilfe von verschiedenen Modi zu implementieren. Der Bankautomat befindet sich zu jedem Zeitpunkt in einem bestimmten Modus und reagiert, je nach Modus, unterschiedlich auf bestimmte Eingaben. Dieser Ansatz unterscheidet sich von der ursprünglichen Version des Automaten. Hier gab es keine Modi und die verschiedenen Funktionen, wie das Geldabheben, wurden von eigenen Klassen übernommen.

In der alten Version des Bankautomaten, konnte ein Admin mit Hilfe eines Iterators auf die einzelnen Benutzer zugreifen. In dem überarbeiteten Modell ist es möglich, aus einer Liste von Benutzern den gewünschten per Mausklick auszuwählen. Dies ermöglicht eine einfachere und schnellere Bearbeitung.

Zudem wird das Speichern der verschiedenen Benutzer nicht mehr innerhalb einer Java-Klasse übernommen, sondern außerhalb in einer JSON-Datei. Die Benutzerdaten werden mit Hilfe der Klasse `BankDatabase.java` in diese Datei übertragen.

3.2.3 Cross-Cutting-Concerns, NFRs

Nun werden kurz die Cross-Cutting-Concerns des Systems, sowie der Umgang mit diesen, vorgestellt.

Ein Benutzer soll in jedem Modus eine Eingabe tätigen können. Daher wurde das Keypad und ein entsprechendes Textfeld so implementiert, dass diese Komponenten stets sichtbar und verfügbar sind. Andere Komponenten werden teilweise unsichtbar gemacht, da diese nicht in jedem Modus gebraucht werden.

Ein weiterer Cross-Cutting-Concern ist das Geben von passendem Feedback an den Benutzer. Hier soll dem Benutzer, unabhängig von dem aktuellen Modus, stets mitgeteilt werden, wenn er eine ungültige Eingabe getätigt hat. Für diese Art von Fehlermeldungen wurde im untersten Bereich des Fensters ein Textfeld angelegt, welches die jeweilige Nachricht in roter Farbe anzeigt.

Außerdem ist die Validierung des Inputs bei einem Bankautomaten äußerst wichtig. Deshalb werden die Eingaben stets auf Richtigkeit überprüft. So wird beispielsweise sichergestellt, dass das eingezahlte Geld keinen Maximalwert überschreitet. Ebenso muss sichergestellt werden, dass ein Benutzer nicht mehr Geld abheben kann, als gerade für ihn verfügbar ist.

Bezüglich der Nicht-funktionalen-Anforderungen wurde auf eine hohe Performance und Bedienbarkeit geachtet. Dem Benutzer wird das Bedienen des Automaten durch ein intuitives Interface leichtgemacht. Die Wartezeiten sind kurz, da die Funktionen zur Berechnung von Überweisungen und Kontoständen eine geringe Laufzeit aufweisen.

3.3 Mensch-Maschine-Schnittstelle

3.3.1 Anforderungen an die Mensch-Maschine-Schnittstelle

Die Mensch-Maschine-Schnittstelle, oder auch Benutzerschnittstelle, bezieht sich auf die Kommunikation zwischen einem Nutzer (Mensch) und dem Geldautomaten (Maschine). Der Mensch gibt mit seinen Aktoren (Händen) eine Eingabe-Information an die Peripherieeinheiten des Geldautomaten, welche eine digitale Information an die Recheneinheit des Geldautomaten weiterleiten. Die von der Recheneinheit entgegengenommene Information wird mittels der aufgespielten Software verarbeitet und eine Ausgabe-Information wird erzeugt. Die Recheneinheit steuert digital die Peripherieeinheiten des Geldautomaten an, welche eine

optische (Bildschirm-Ausgabe) und mechanische Ausgabe Information (Geldauszahlung) erzeugen. Die Rückgabe-Informationen werden vom Menschen visuell (Bildschirm-Information) und haptisch (Annahme des ausgezahlten Geldes) verarbeitet.

Ein-/Ausgabe	Mensch Schnittstelle	Hardware Schnittstelle	Software Schnittstelle
Eingabe	Hände	Encrypting PIN Pad	Tastenabfrage
	Augen	ID-Kartenleser, Softkeys oder Touchscreen	Touchbildschirm Abfrage
Ausgabe	Hände	Bildschirm	Grafikausgabe
	Augen	Auszahlmodul	Peripherie Ansteuerung

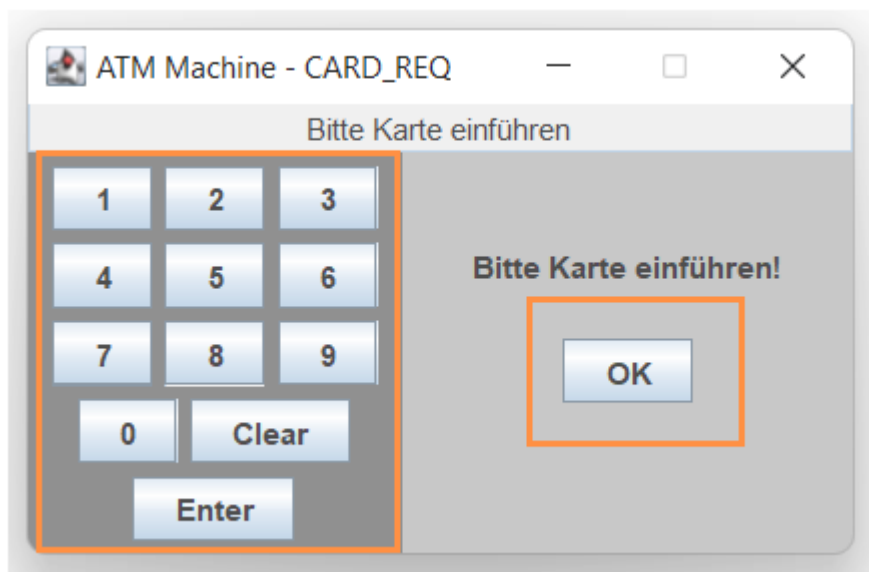
3.3.2 Gestaltungsprinzipien

Gestaltungsprinzipien bzw. Gestaltungsgesetze je nach Literatur, helfen ein ansprechendes und verständliches Design zu erstellen. Es sind psychologische Ansätze, wie das menschliche Gehirn visuelle Informationen wahrnimmt und ordnet. Folgende Prinzipien wurden in diesem Projekt beachtet:

Benutzeransicht

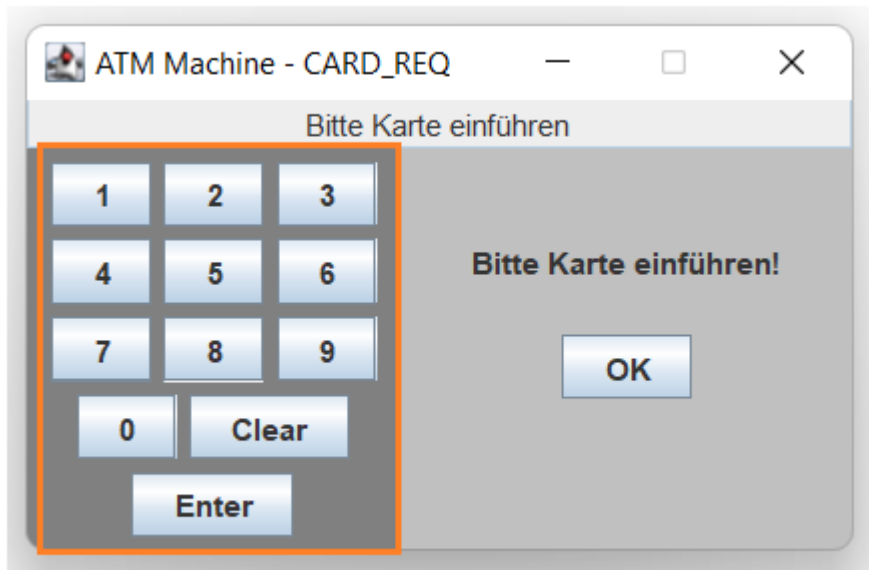
Prinzip der Ähnlichkeit

Elemente die ähnliche bzw. gleiche Funktionen haben wurden gleich gestaltet. Wie zum Beispiel die Eingabe Tastenfeld Null bis Neun. Diese haben alle das gleiche Design. Die Tasten "Enter", "Clear" und "OK" haben das gleiche Design, aber führen unterschiedliche Funktionen aus.



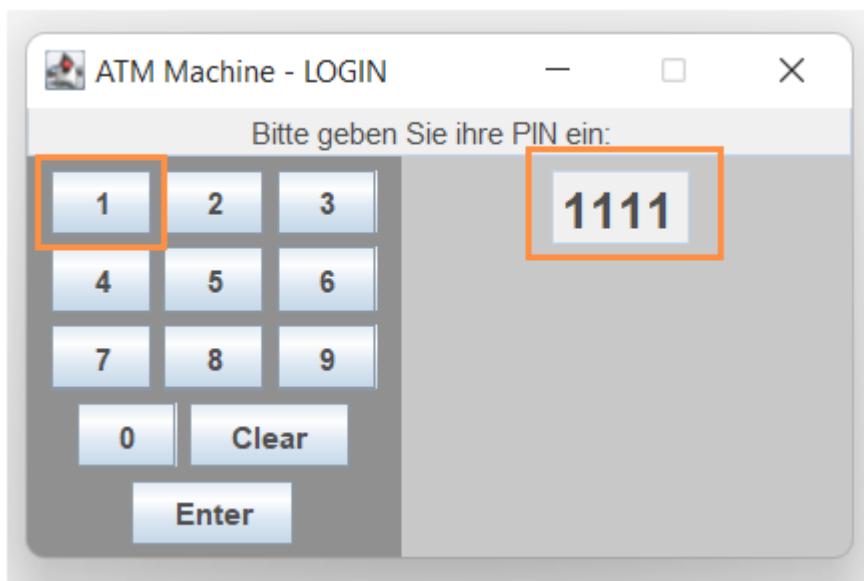
Prinzip der Nähe

Elemente, welche die gleichen Funktionen ausführen, oder helfen, dass diese Funktion ausgeführt werden können, wurden räumlich nah platziert. Wie zum Beispiel die Tasten Null bis Neun. Sie wurden nah aneinander, links im Display platziert. Auch die Knöpfe "Enter" und "Clear" wurden im Tastenfeld platziert, da sie die Eingabe bestätigen oder löschen.



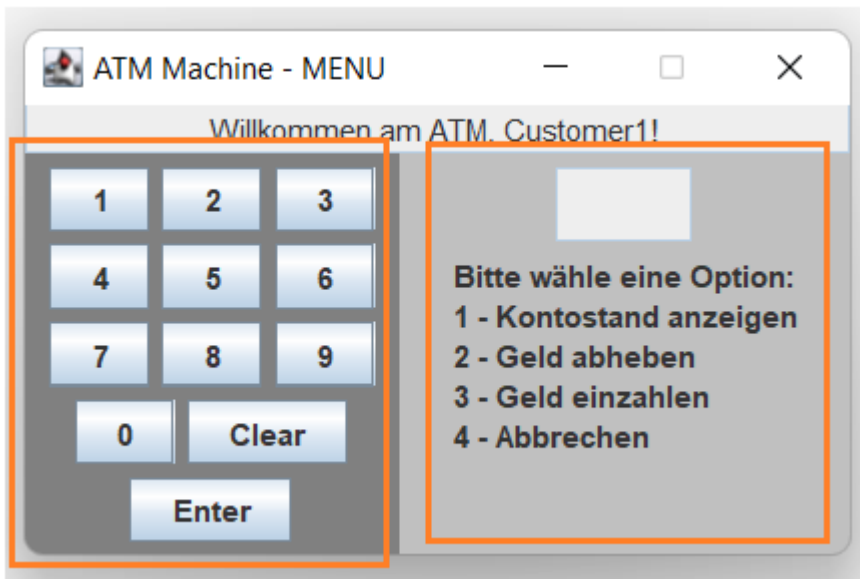
Prinzip der Prägnanz

Interaktionselemente wurden hervorgehoben. Wie im Falle der Eingabeknöpfe Null bis Neun. Aber auch das Ausgabefenster wurde quadratisch und mit weißem Hintergrund gestaltet.



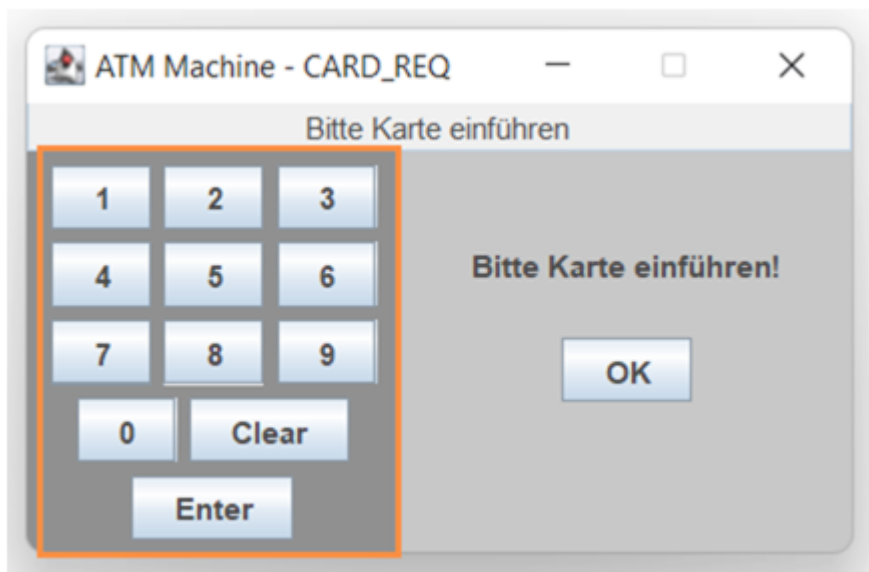
Prinzip der gemeinsamen Region

Eingabe Elemente werden links vom Display platziert und das Ausgabe-Fenster rechts vom Display. Die Regionen sind mit einer unterschiedlichen Hintergrundfarbe optisch getrennt.



Prinzip der Erfahrung

Elemente werden so gestaltet, wie sie sich zuvor in der Praxis schon bewiesen haben, oder es aus kultureller Sicht angenommen wurde. In unserem Fall wurde kulturell standardisiert, dass numerische Tastenfelder in einer 3x4 Matrix angeordnet werden.



Admin-Ansicht

Prinzip der Ähnlichkeit

Elemente, die ähnliche bzw. gleiche Funktionen haben, wurden gleich gestaltet. Wie zum Beispiel die Buttons „Neuer Account“, „Account löschen“ und „Speichern“, die alle benötigt werden um Accountmanagement auszuüben. Die Eingabefelder zum Erstellen neuer Accounts wurden ebenfalls gleich gestaltet.

Admin-Ansicht

Willkommen, Manager1!

Customer1	Account-Nummer	12345
Customer2	Benutzername	Customer1
Customer3	Verfügbares Guthaben	1050.0
Manager1	Gesamtes Guthaben	1200.0
Benutzer1	Pin	1111
Benutzer2		

Neuer Account Account löschen Speichern

Prinzip der Nähe

Elemente, welche die gleichen Funktionen ausführen, oder dazu helfen, dass diese Funktion ausgeführt werden können, wurden räumlich nah platziert. Die Buttons für das Accountmanagement wurden alle unten platziert. Die Eingabefelder zum Erstellen eines neuen Accounts wurden alle rechts am Bildschirmrand zusammengefasst und die Auswahl eines Kunden an der linken Seite des Bildschirmrandes.

Admin-Ansicht

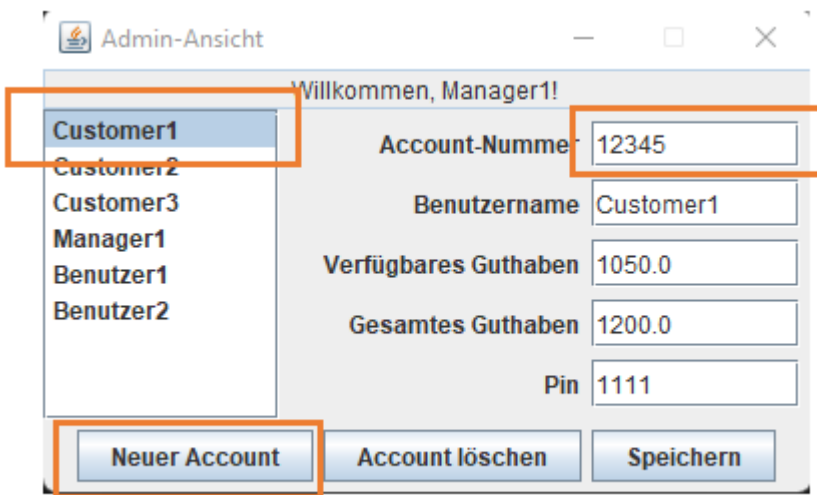
Willkommen, Manager1!

Customer1	Account-Nummer	12345
Customer2	Benutzername	Customer1
Customer3	Verfügbares Guthaben	1050.0
Manager1	Gesamtes Guthaben	1200.0
Benutzer1	Pin	1111
Benutzer2		

Neuer Account Account löschen Speichern

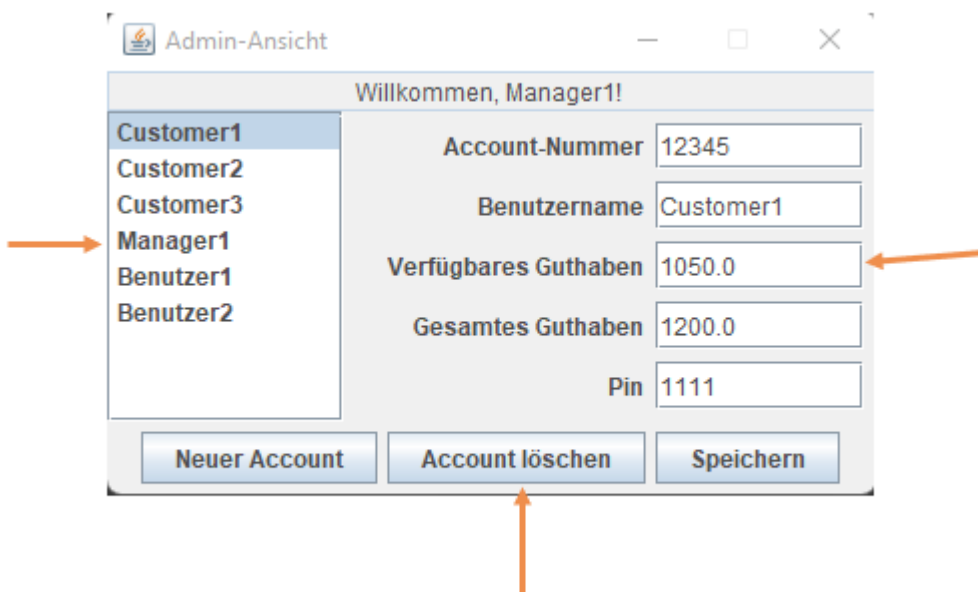
Prinzip der Prägnanz

Interaktionselemente wurden hervorgehoben. Die Buttons wurden bläulich gefärbt und umrandet, um sie hervorzuheben. Ebenso wurde bei der Auswahl eines Kunden, die Auswahl bläulich hinterlegt um zu markieren, welcher Kunde ausgewählt wurde. Zusätzlich wurden die Eingabefelder alle mit weißem Hintergrund versehen.



Prinzip der gemeinsamen Region

Die Auswahl der Kunden befindet sich rechts am Bildschirm und die dazugehörigen Daten werden links im Bildschirm angezeigt. Die verfügbaren Optionen wurden unten am Bildschirmrand platziert.



3.3.3 Styleguide

Im Folgenden wurden Design-Mockups erstellt, welche die Ansichten für den Benutzer und den Administrator repräsentieren.

Mockup für die Standardansicht des Automaten

The mockup shows a rectangular frame representing the ATM screen. At the top, a horizontal bar contains the text "ATM". Below this bar, the screen is divided into two main sections. On the left is a numeric keypad with buttons for digits 1 through 9, 0, a "Clear" button, and an "Enter" button. On the right is a larger area containing the text "Bitte geben Sie ihre PIN ein" (Please enter your PIN) above a single-line text input field.

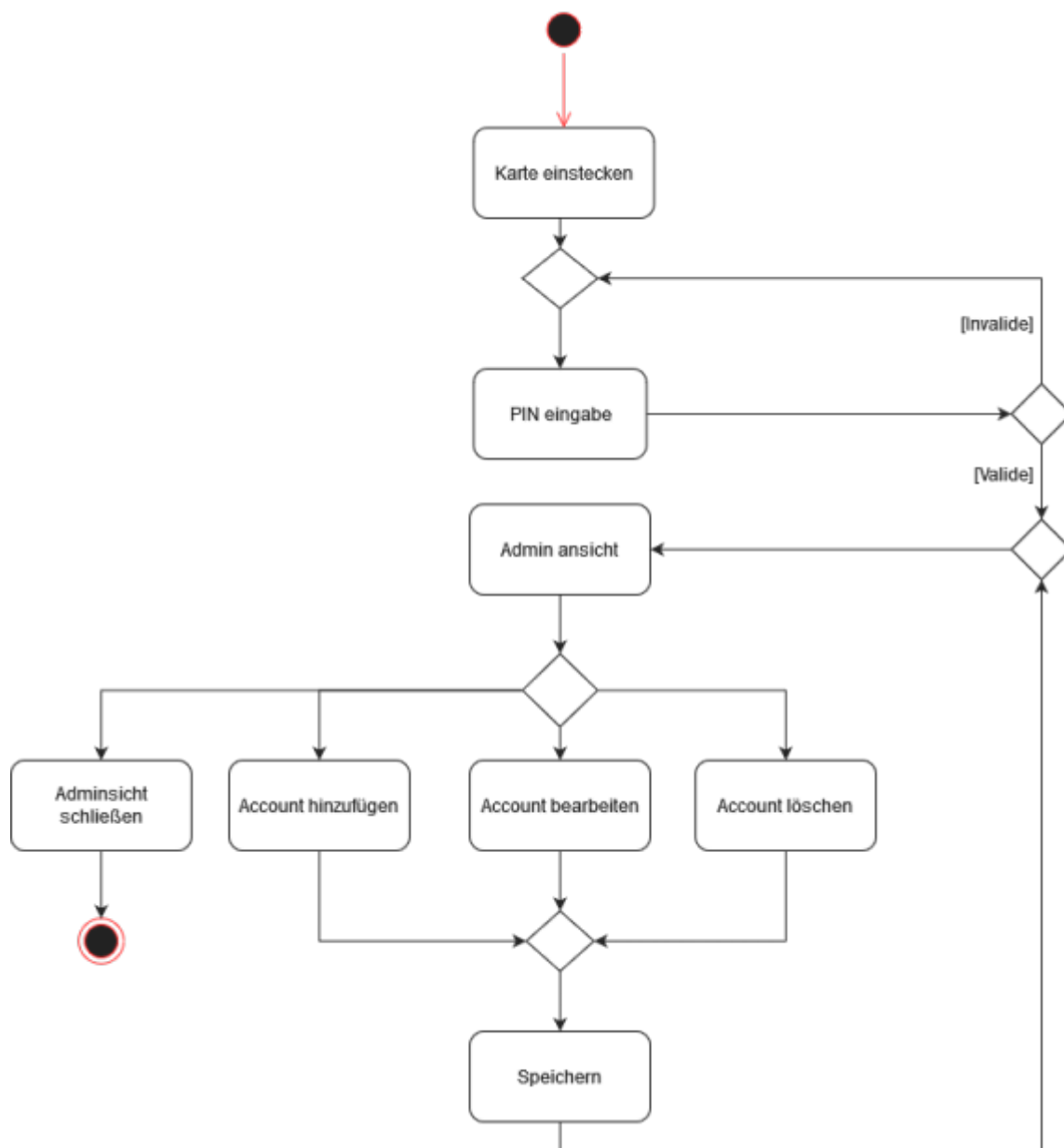
Mockup für die Administratoransicht des Automaten

The mockup shows a rectangular frame representing the ATM screen. At the top, a horizontal bar contains the text "ATM". Below this bar, the screen is divided into two main sections. On the left is a list area with the text "Customer 1" and "Customer 2". On the right is a form area with five labels and corresponding input fields: "Account nummer", "Benutzername", "Verfügbares Guthaben", "Gesamtes Guthaben", and "PIN". At the bottom of the right section are three buttons: "Neuer ACC", "ACC löschen", and "Speichern".

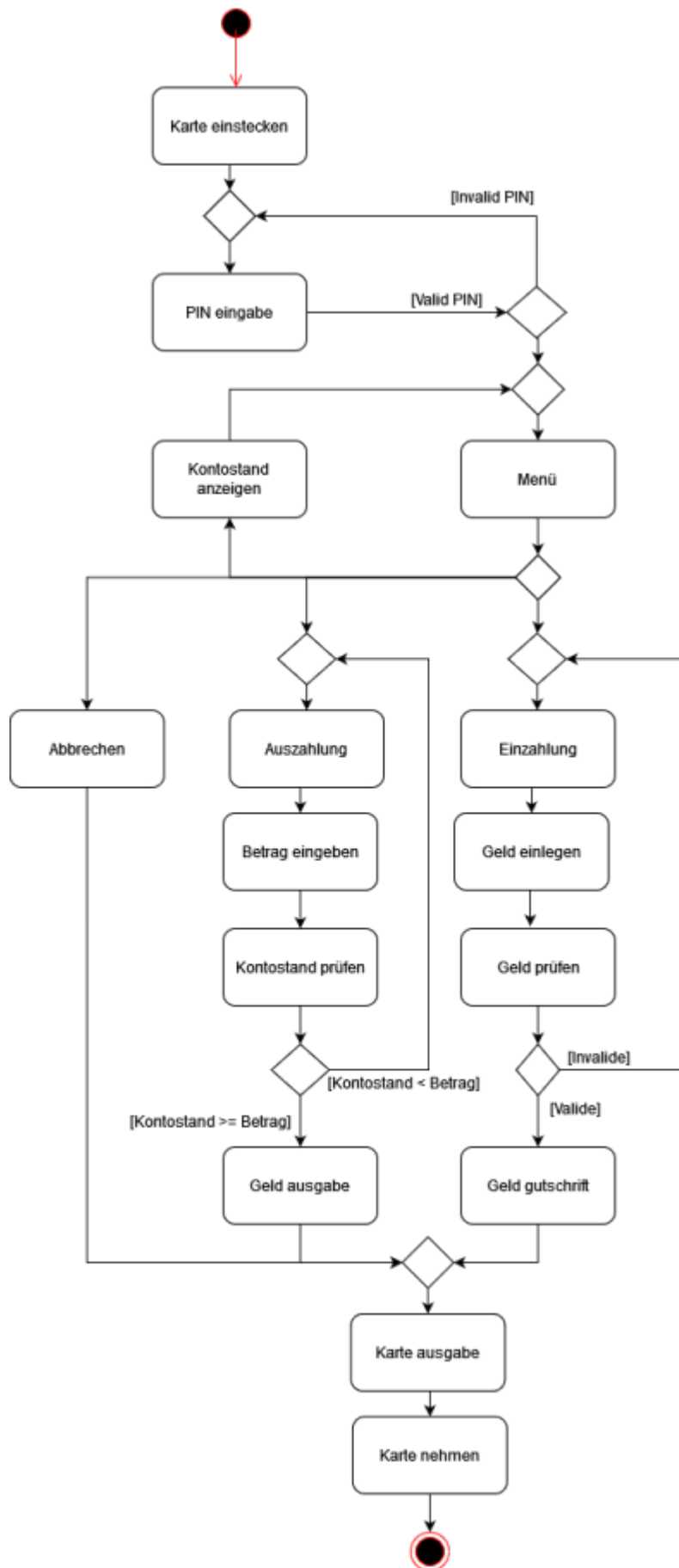
3.3.4 Interaktionsmodellierung

Im Folgenden wurde die Interaktion zwischen den Benutzern und dem Geldautomaten modelliert und in einem UML-Aktivitätsdiagramm dargestellt.

Admin-Ansicht



Benutzeransicht



4. Testdokumentation

In der folgenden Dokumentation werden die für das Projekt durchgeführten Test beschrieben. Diese sind entweder manuell oder mit Hilfe von JUnit ausgeführt worden.

Name	Sind Komponenten initialisiert
Anforderung	Die ATM-Instanz soll einen screen und eine bankDatabase haben
Vorbedingung	ATM-Instanz ist erzeugt
Nachbedingung	Screen und bankDatabase des ATM sind initialisiert
Testschritte	Stelle sicher, dass Komponenten nicht null sind

Name	Wechsel in BALANCE Modus
Anforderung	Mit dem Input "1" soll in den BALANCE Modus gewechselt werden
Vorbedingung	ATM-Instanz ist erzeugtUser ist eingeloggtMomentan im MENU Modus
Nachbedingung	Guthaben wird angezeigtATM im BALANCE Modus
Testschritte	Funktion atm.atmEnterAction() wird mit Input "1" aufgerufen

Name	Falscher Input in Menü
Anforderung	Bei falschem Input soll ATM im selben Modus bleiben
Vorbedingung	ATM-Instanz ist erzeugtUser ist eingeloggtMomentan im MENU Modus
Nachbedingung	ATM gibt Fehlermeldung, resettet das Textfeld und bleibt im selben Modus
Testschritte	Funktion atm.atmEnterAction() wird mit falschem Input aufgerufen

Name	"Back" Button
Anforderung	Der "Back" Button, soll den Modus zu MENU wechseln
Vorbedingung	ATM-Instanz ist erzeugtUser ist eingeloggtMomentan im MENU Modus
Nachbedingung	ATM befindet sich wieder im MENU Modus
Testschritte	Wechsel in BALANCE Modus, Drücken auf "Back" ButtonWechsel in WITHDRAWAL Modus, Drücken auf "Back" ButtonWechsel in DEPOSIT Modus, Drücken auf "Back" Button

Name	"Clear" Button
Anforderung	Bei Drücken auf den "Clear"-Button soll das Textfeld resettet werden
Vorbedingung	ATM-Instanz ist erzeugtUser ist eingeloggtMomentan im MENU Modus
Nachbedingung	Das Textfeld ist leer
Testschritte	Beliebiger Input wird in Textfeld eingegeben"Clear"-Button wird gedrückt

Name	Ungültiger Pin Input
Anforderung	Bei falscher Pin soll eine LoginFailedException geworfen werden
Vorbedingung	ATM-Instanz ist erzeugtEin neuer Account ist angelegt
Nachbedingung	ATM hat keinen Pin akzeptiert, da Pins aus 4 Ziffern bestehen müssenATM befindet sich noch im LOGIN Modus
Testschritte	Anmeldungsversuche mit verschiedenen ungültigen PinsZuerst ein Pin mit Buchstaben, dann ein Pin mit 5 Ziffern und zuletzt ein Pin mit 3 Ziffern

Name	Neuen Account erstellen
Anforderung	In der AdminView soll ein neuer Account erstellt werden können
Vorbedingung	ATM-Instanz ist erzeugt
Nachbedingung	Neuer Account wurde angelegtATM im ADMIN Modus
Testschritte	Neuer Admin-Account wird erstellt und der Datenbank hinzugefügtDer Admin loggt sich mit seiner Pin einÜberprüfen, ob die Länge der Account Liste sich um 1 erhöht hat

Name	Credit und Debit Funktion
Anforderung	Credit Funktion soll das Guthaben um mitgegebenen Betrag erhöhenDebit Funktion soll das Guthaben um mitgegebenen Wert verringern
Vorbedingung	ATM-Instanz ist erzeugtNeuer Account "a1" ist angelegt
Nachbedingung	Guthaben ist gleich hoch wie vor der Durchführung des Tests
Testschritte	a1.credit(5) wird aufgerufenÜberprüfen, ob sich Guthaben um 5 erhöht hata1.debit(5) wird aufgerufenÜberprüfen, ob sich Guthaben um 5 verringert hat

Name	Geldschein-Menge überprüfen
Anforderung	Nach dem Einzahlen von Geld soll sich die Menge der jeweiligen Euro-Scheine entsprechend verändern
Vorbedingung	Ein Objekt der Klasse CashDispenser ist erzeugt
Nachbedingung	Anzahl der verschiedenen Geldscheine hat sich erhöht
Testschritte	Es werden 875€ in den Automaten gezahltÜberprüfen, dass acht 100€-Scheine, ein 50€-Schein, ein 20€-Schein und ein 5€-Schein mehr im CashDispenser sind

Name	Ungültiger Einzahlungs-Betrag
Anforderung	Es soll eine InvalidTransactionException geworfen werden, wenn versucht wird einen ungültigen Betrag einzuzahlen
Vorbedingung	Ein Objekt der Klasse CashDispenser ist erzeugt
Nachbedingung	Es wurde 3 mal eine InvalidTransactionException geworfen
Testschritte	Es wird überprüft, ob bei folgenden ungültigen Eingaben eine Exception geworfen wird:- Eingabe: -4€ (negativ)- Eingabe: 7€ (nicht durch 5 teilbar)- Eingabe: 1100€ (mehr als 1000€ auf einmal)

Zusätzlich zu den automatisch durchlaufenen Unit-Tests wurden noch einige Tests manuell durchgeführt. Jegliche Funktionen der Software wurden durch das direkte Benutzen dieser überprüft. Der gesamte Prozess, vom Login, zum Geldabheben, bis zum Logout wurde mehrmals mit verschiedenen möglichen Inputs und Reihenfolgen ausgeführt. Hierbei wurde auch auf die korrekte Anordnung der UI-Komponenten geachtet. Die verschiedenen Textbeschreibungen und Überschriften wurden ebenfalls auf ihre Richtigkeit überprüft.

Zusammenfassend lässt sich sagen, dass alle geschriebenen Tests erfolgreich durchlaufen wurden und die Software wie erwünscht funktioniert. Es wurden keine Fehler gefunden, welche die Abnahme der Software verhindern würden. Natürlich kann nicht ausgeschlossen werden, dass kleinere Fehler beim Benutzen der Software auftreten könnten, jedoch sind beim Nutzen und Testen der Software keine solche Fehler aufgefallen.

5. Abnahmedokumentation

5.1 System Under Test

System Under Test bezieht sich auf die Validierung des Systems. Das System wird unter verschiedenen Szenarien getestet. Die Anforderungsspezifikationen werden den Testfällen zugeordnet, um zu überprüfen, ob alle Anforderungen erfüllt sind. Die folgende Tabelle beinhaltet die Testfälle und Testergebnisse. Für detaillierte Testspezifikationen siehe [Testdokumentation](#).
Bestanden: Testergebnisse wie erwartet Nicht bestanden: Testergebnisse nicht wie erwartet

Testfall	Testergebnis
Sind Komponenten initialisiert	Bestanden
Wechsel in BALANCE Modus	Bestanden
Falscher Input in Menü	Bestanden
"Back" Button	Bestanden
"Clear" Button	Bestanden
Ungültiger Pin Input	Bestanden
Neuen Account erstellen	Bestanden
Credit und Debit Funktion	Bestanden

Die folgende Tabelle beinhaltet die User Stories und deren Ergebnisse. Für detaillierte User Stories siehe [Anforderungsdokumentation](#). Implementiert: User Stories erfolgreich implementiert Nicht implementiert: User Stories nicht erfolgreich implementiert

Nr.	User Stories	Testergebnis
1	...Verschiedene Geldbeträge eingeben...	Implementiert
2	...Sehen, wie viel Geld auf Konto ist...	Implementiert
3	...maximal Debit Betrag pro Tag festlegen...	Implementiert
4	...vierstelligen Pin zu meiner Karte eingeben...	Implementiert
5	...Ziffern meiner Pin ändern...	Implementiert (Administrator)
6	...Länge meiner Pin ändern...	Nicht implementiert
7	...Stückelung auswählen...	Nicht implementiert
8	...in mein Konto einloggen...	Implementiert
9	...gegen Gebühren Geld abheben...	Nicht implementiert
10	...vollständige und detaillierte Dokumentation...	Implementiert

5.2 Bereitstellung zur Abnahme

Das Abnahmeprotokoll kann [hier](#) als PDF-Datei heruntergeladen werden.

Abnahmeprotokoll

In diesem Absatz finden sie die Abnahmeerklärung zwischen Auftraggeber und Auftragnehmer.

Machen sie ein Kreuz zwischen den eckigen Klammern, wenn die Aussage korrekt ist [X].

Abnahmeprotokoll



Projektname: ATM Dokumentation

Projektnummer: 0001

Auftraggeber: Katharina Franz, Technische Hochschule Aschaffenburg

Auftragnehmer: Panzerknacker

Abnahmeumfang: Gesamtabnahme [] Teilabnahme []

Projektbeginn: 25.04.2022

Abgabetermin: 20.06.2022

Lieferanhang

Bestätigung	Lieferartikel
[]	1. Anforderungsdokumentation
[]	2. Architekturdokumentation
[]	3. Testdokumentation
[]	4. Abnahmedokumentation
[]	5. Benutzerdokumentation
[]	6. Projektdokumentation
[]	7. Ist-Dokumentation
[]	8. Delta-Dokumentation
[]	Neuer Delta-Code
[]	Dokumentierter Ist-Code

Offene Fehler

Nr.	Beschreibung	Schwere	Zu beheben bis
1			
2			
3			