

Modul 326 - KKK

Projekt - KKK

Fabian Kurt

2018

Inhaltsverzeichnis

[Projektbeschrieb 2](#_Toc510779155)

[Use Case 3](#_Toc510779156)

[Klassendiagramm 4](#_Toc510779157)

[Kontextdiagramm 5](#_Toc510779158)

[Anforderungen 6](#_Toc510779159)

[Funktionale Anforderungen 6](#_Toc510779160)

[Nicht funktionale Anforderungen 6](#_Toc510779161)

[Verwendete Libraries 7](#_Toc510779162)

[Java 1.8 7](#_Toc510779163)

[groovy 2.4.15 7](#_Toc510779164)

[mysql connection java 5.1.43 7](#_Toc510779165)

[Testszenarien 8](#_Toc510779166)

[Implementierungsentscheidung 9](#_Toc510779167)

[In Bezug auf Abstrakte Klassen 9](#_Toc510779168)

[In Bezug auf Klassenhierarchien 9](#_Toc510779169)

[In Bezug auf Interfaces 9](#_Toc510779170)

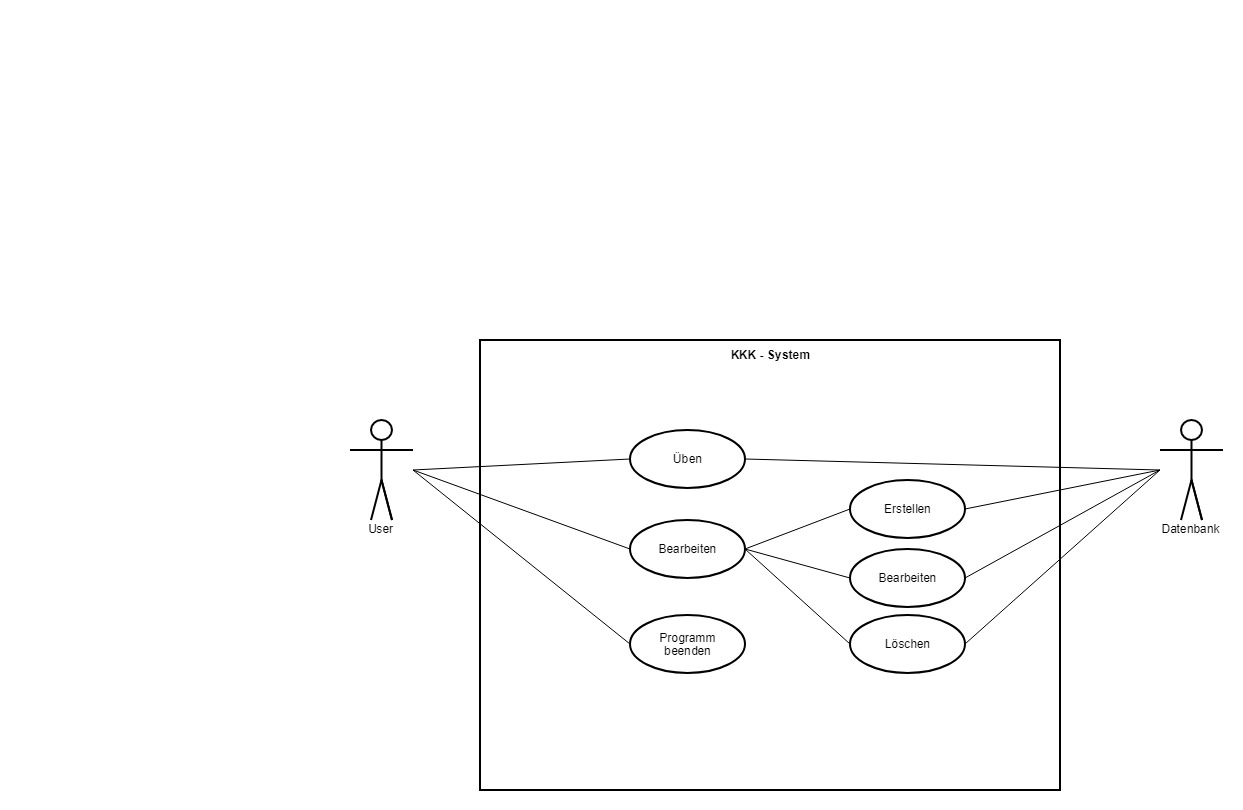
[Zugriff auf das Datenbanksystem 10](#_Toc510779171)

[Datensicherheit & Applikationssicherheit 11](#_Toc510779172)

[Fazit 12](#_Toc510779173)

# Projektbeschrieb

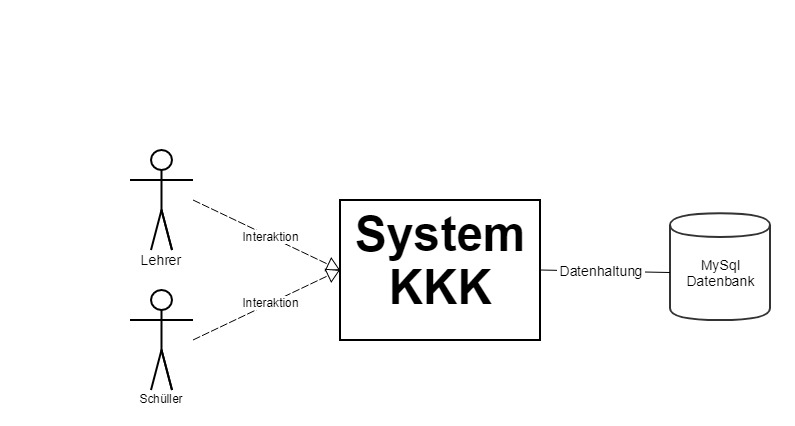
# Use Case



# Klassendiagramm

<KlassenDiagram.pdf>

# Kontextdiagramm



# Anforderungen

## Funktionale Anforderungen

|  |
| --- |
| **Programm / Applikation** |
| Fehlerlogging mittels Output-Stream korrekt umgesetzt |
| Korrekte Implementierung & Verwendung des Erzeugungsmusters: Singleton |
| Korrekte Implementierung & Verwendung des Erzeugungsmusters: Factory |
| Korrekte Implementierung & Verwendung des Verhaltensmuster: Observer |
| Korrekte Implementierung des Architekturmusters: Model-View-Controller / oder MVVM |
| Verwendung von mindestens 3 selbst erstellten Interfaces korrekt und sinngemäss für das Projekt |
| Verwendung von mindestens 3 abstrakten Klassen korrekt und sinngemäss für das Projekt |
| Verwendung von mindestens 3 Basisklassen korrekt und sinngemäss für das Projekt |
| Abstrahierung der Datenbankschnittstelle auf einzelne Entitäten korrekt und sinngemäss umgesetzt (Siehe Arbeitsblatt 02 – Datenbankanbindung) |
| Programm ist Lauffähig und stürzt nicht ab |
| Testszenarien gemäss Dokumentation implementiert und sind Funktionsbereit |
| Umsetzung mindesten zweier Entitäten mittels CRUD-Operationen |
| GUI ansprechend und sinngemäss gestaltet und umgesetzt |

## Nicht funktionale Anforderungen

|  |
| --- |
| **Programm / Applikation** |
| Applikation wird auf allen Desktopgeräten von 1280x1024 – 1920x1080 korrekt dargestellt und kann frei vergrössert & verkleinert werden |

# Verwendete Libraries

## Java 1.8



Wird verwendet für das Programmieren der Applikation.

## groovy 2.4.15



Wird verwendet für das Testen (JUnit-Tests) der Applikation.

## mysql connection java 5.1.43



Wird verwendet um eine Verbindung zur Datenbank herzustellen und mit ihr zu arbeiten.

# Testszenarien

|  |  |
| --- | --- |
| Testszenarien | |
| Testszenario 01 | Programm startet und Stürzt nicht ab |
| Testszenario 02 | Man kann zwischen den einzelnen Scenen wechseln ohne dass das Programm abstürzt. |
| Testszenario 03 | Man kann den die CRUD-Funktionalität(Button) Update(Bearbeiten) und Delete(Löschen) im der Scene ArbeitenAuswählen nur benutzten wen man eine Kartei ausgewählt hat. |
| Testszenario 04 | Man kann nur eine Übung starten wen man eine Kartei auswählt. |
| Testszenario 05 | Der User kann sich die Karteien anzeigen lassen(immer nur sie Ausgewählte). |
| Testszenario 06 | Der User kann eine neue Kartei Erstellen. |
| Testszenario 07 | Der User kann einer Kartei einen Namen geben. |
| Testszenario 08 | Einer neu erstellten Kartei können Karten hinzugefügt werden. |
| Testszenario 09 | Einer schon erstellten Kartei können später noch beliebig oft Karten hinzugefügt werden. |
| Testszenario 10 | Eine Kartei samt allen dazugehörigen Karten können gelöscht werden |
| Testszenario 11 | Alle benötigten Daten(aus MySql-Server) können korrekt aus/eingelesen werden. |
| Testszenario 12 | Die Frage wird angezeigt und nach der Beantwortung wird die nächste angezeigt wen vorhanden. |
| Testszenario 13 | Dem User wird angezeigt(MessageBox), wen er die richtige Antwort eingegeben hat. |
| Testszenario 14 | Dem User wird angezeigt(MessageBox) das die Antwort falsch sei und was die Richtige Antwort wäre, wen er eine falsche Antwort eingegeben hat. |
| Testszenario 15 | Am Ende einer Übung wird dem User angezeigt wie viel er richtig und wie viel er falsch hatte. |

# Implementierungsentscheidung

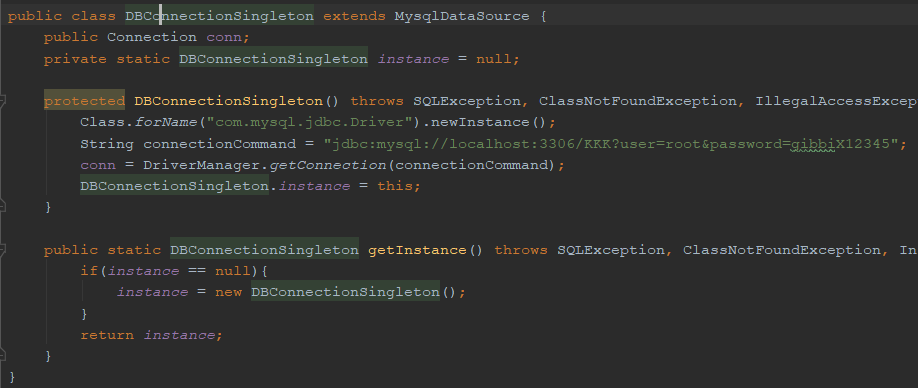
## In Bezug auf Abstrakte Klassen

## In Bezug auf Klassenhierarchien

## In Bezug auf Interfaces

# Zugriff auf das Datenbanksystem

**2.**



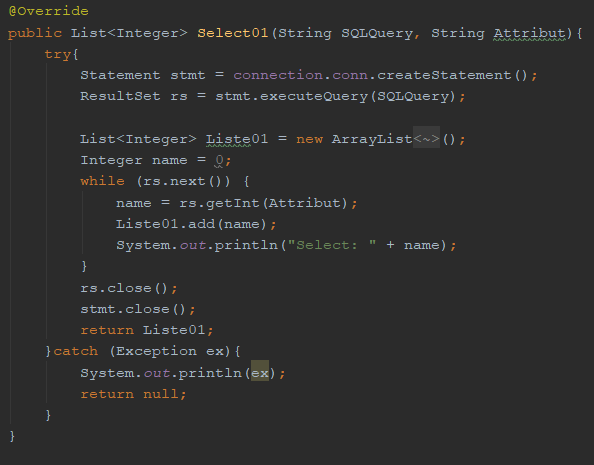
**1.**

**4.**

**3.**



**5.**



**10.**

**9.**

**8.**

**7.**

**6.**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Es wird eine Klasse instanziiert mit einem Methoden Aufruf der eine Variable festlegt. | 6. | Es wird ein Statement deklariert und initialisiert mit der conn(Datenbank Verbindung) Variable. |
| 2. | Es werden die nötigen Treiber geladen für eine DBConnektion. | 7. | Der Resultset wird deklariert und mir der Ausgabe von dem ausgeführten stmt initialisiert. |
| 3. | Eine String Variable wird deklariert und Initialisiert mit allen wichtigen Angaben für eine Verbindung zur Datenbank | 8. | Eine String Liste „Liste01“ und eine Integer Variable „name“ werden deklariert und Initialisiert. |
| 4. | Die Instanz die = null ist, bekommt den Wert der aktuellen Instanz | 9. | In der schleifen werden alle werte von rs in die Liste „Liste01“ geladen. |
| 5. | Die Methode Slecet01 wird aufgerufen. | 10. | Beendet rs & stmt und gib Liste01 zurück. |

# Datensicherheit & Applikationssicherheit

Habe keine Datensicherheit & Applikationssicherheit integriert weil es gibt keine schützenswerte Daten. es sind alle für jeden zugänglich.

# Fazit