

# Morik Password Manager

## Programmentwurf

von

**Moritz Gutfleisch**

und

**Erik Zimmermann**

Abgabedatum:	01. Februar 2018
Bearbeitungszeitraum:	01.10.2017 - 31.01.2018
Matrikelnummer, Student:	0000000, Moritz Gutfleisch
Matrikelnummer, Student:	0000000, Erik Zimmermann
Kurs:	TINF19B1
Gutachter der Dualen Hochschule:	Daniel Lindner

## **Abstract**

*- English -*

This is the starting point of the Abstract. For the final bachelor thesis, there must be an abstract included in your document. So, start now writing it in German and English. The abstract is a short summary with around 200 to 250 words.

Try to include in this abstract the main question of your work, the methods you used or the main results of your work.

## **Abstract**

- *Deutsch* -

Dies ist der Beginn des Abstracts. Für die finale Bachelorarbeit musst du ein Abstract in deinem Dokument mit einbauen. So, schreibe es am besten jetzt in Deutsch und Englisch. Das Abstract ist eine kurze Zusammenfassung mit ca. 200 bis 250 Wörtern.

Versuche in das Abstract folgende Punkte aufzunehmen: Fragestellung der Arbeit, methodische Vorgehensweise oder die Hauptergebnisse deiner Arbeit.

# Inhaltsverzeichnis

Abkürzungsverzeichnis	IV
Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
Quellcodeverzeichnis	VII
1 Einleitung	1
2 Clean Architecture	2
2.1 Plugins . . . . .	2

# Abkürzungsverzeichnis

**SQL**     Structured Query Language

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Quellcodeverzeichnis



# **1 Einleitung**

## 2 Clean Architecture

### 2.1 Plugins

#### 2.1.1 Datenbank

Die Datenbank stellt eines der Plugins dar. Als Technologie wurde hier eine SQLite3 Datenbank verwendet. Die konkrete Implementierung der Schnittstelle zur SQLite-Datenbank befindet sich in der Klasse *SQLiteDatabase*. Diese Klasse erbt von der abstrakten Klasse *AbstractDatabase*, die sich in der Applikationsschicht befindet. Hierdurch wird eine Dependency Inversion erreicht, da nun eine *AbstractDatabase* von anderen Klassen verwendet werden kann, um SQL-Befehle auf der Datenbank auszuführen, statt eine konkrete *SQLiteDatabase* zu verwenden, was die Abhängigkeit von innen nach außen laufen lassen würde. Der Klasse *DbInterface* wird im Konstruktor eine solche *AbstractDatabase* übergeben, was die Dependency Injection umsetzt. Des Weiteren implementiert die Klasse *DbInterface* die eigentliche Funktionalität in Form der SQL-Anweisungen und führt diese lediglich über die konkrete Implementierung der *AbstractDatabase* in der Datenbank aus. Somit geht keine Funktionalität verloren, wenn das Plugin durch eine andere Datenbanktechnologie ausgetauscht wird.

Auf die Benutzung von Prepared Statements wurde hier verzichtet, da die Datenbank lokal ist und nur der Benutzer Befehle auf ihr ausführt. Würde die Datenbank über eine öffentliche Schnittstelle angesteuert werden, so wäre dies nicht zu vernachlässigen.