Software Grobdesign Dokument

**Projekt:**

Passwortmanager

**Autor:**

Ekaterina Karavaeva

**Datum:**

20.01.2025

**Version:**

1.0

**Prüfer:**

Sascha Wanninger

**Changelog:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | **Datum** | **Version** | **Änderung** |
| Ekaterina Karavaeva | 20.01.2025 | 1.0 |  |
|  |  |  |  |
|  |  |  |  |

**Software Grobdesign Dokument**

Dieses Dokument beschreibt den Prozess des Grobdesigns in der Softwareentwicklung. Es behandelt die Übertragung von Anforderungen in die Sprache der Entwickler, die Verwendung von UML-Diagrammen zur Modellierung, die Definition der Systemarchitektur, die Auswahl von Frameworks und das Design der Benutzeroberfläche. Der Schwerpunkt liegt auf der iterativen Entwicklung von Klassen und Objekten, der Erstellung von Klassendiagrammen und der Validierung des Modells mithilfe von Sequenzdiagrammen.

**1. Systemarchitektur**

**Hardware und Software:**

* Das Projekt wurde auf Windows-Betriebssystemen durchgeführt.
* Die Entwicklung erfolgte auf existierenden Rechnern der Entwickler.

**Middleware:**

* Es wurde keine spezifische Middleware verwendet, da die Kommunikation direkt über die Anwendung erfolgte.

**Schnittstellen und Programmiersprachen:**

* Die Programmiersprache für die Entwicklung war Python.
* Die Entwicklungsumgebung war Visual Studio Code.

**Persistenzframework:**

* Für die langfristige Speicherung der Daten wurde ein CSV-Format verwendet, um Benutzerdaten und Passwörter zu speichern.

**1.1 Ergänzungen:**

**Entwicklungsumgebung:**

* Visual Studio Code wurde als Hauptentwicklungsumgebung genutzt, da es eine benutzerfreundliche Oberfläche und umfangreiche Erweiterungen für Python bietet.

**Versionskontrolle:**

* Git wurde für die Versionskontrolle und Zusammenarbeit im Team verwendet.

**Sicherheitsmaßnahmen:**

* Die Passwörter wurden mit SHA-256 gehasht, um die Sicherheit der Benutzerdaten zu gewährleisten.

**Dokumentation:**

* Die Dokumentation des Projekts wurde in Markdown-Dateien geführt, die ebenfalls in Visual Studio Code bearbeitet wurden.

## **2. Systemübersicht**

Der Passwortmanager besteht aus vier Hauptklassen:

* **Main**
* **User**
* **Entry**
* **Logic**

## **3. Hauptklassenbeschreibungen**

### **3.1 Main**

**Beschreibung:** Diese Klasse ist die Hauptklasse für den Passwortmanager. Sie initialisiert die Logik, den angemeldeten Benutzer und den generierten Eintrag.

**Attribute:**

* **functions:** Instanz der Klasse Logic, die die Funktionalitäten für das Benutzermanagement und die Passwortverwaltung bereitstellt.
* **loggedUser:** Der aktuell angemeldete Benutzer.
* **generatedEntry:** Der generierte Eintrag.
* **usersFilePath:** Pfad zur Datei, in der die Benutzerdaten gespeichert sind.

### **3.2 User**

**Beschreibung:** Diese Klasse repräsentiert einen Benutzer in der Passwort-Manager-Anwendung.

**Attribute:**

* **\_username:** Der Benutzername des Benutzers.
* **\_passwordHash:** Der gehashte Passwort des Benutzers.
* **\_password:** Das Klartext-Passwort des Benutzers (optional).
* **myEntrys:** Liste der Einträge des Benutzers.
* **oldPasswords:** Liste der alten Passwörter des Benutzers.

**3.3 Entry**

**Beschreibung:** Diese Klasse repräsentiert einen Eintrag in der Passwort-Manager-Anwendung. Jeder Eintrag enthält ein Passwort, einen Benutzernamen, eine URL und eine optionale Notiz.

**Attribute:**

* **\_password:** Das Passwort für den Eintrag.
* **\_username:** Der Benutzername für den Eintrag.
* **\_url:** Die URL für den Eintrag.
* **\_notice:** Zusätzliche Notizen für den Eintrag.

### **3.4 Logic**

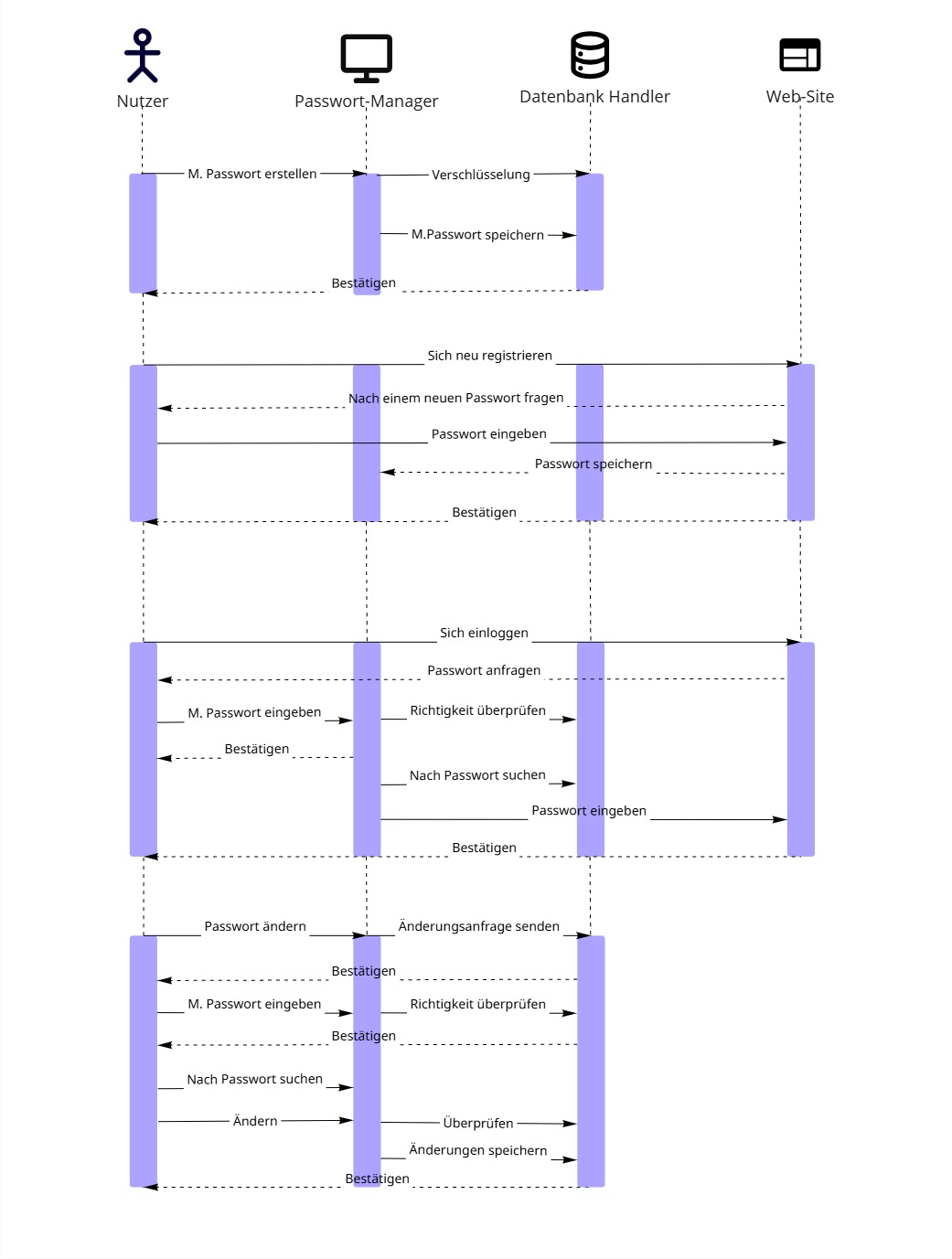
**Beschreibung:** Diese Klasse bietet Funktionalitäten für das Benutzermanagement und die Passwortverwaltung.

**Attribute:**

* **allUsers:** Liste aller Benutzer.
* **alphabetLowercase:** Liste der Kleinbuchstaben.
* **alphabetUppercase:** Liste der Großbuchstaben.
* **numberList:** Liste der Zahlen.
* **symbolsList:** Liste der Symbole.
* **completeList:** Liste, die alle oben genannten Listen enthält.

**3.5 Sequenz-Diagramm**

Das folgende Diagramm zeigt ein Sequenzdiagramm, das die Interaktionen zwischen verschiedenen Akteuren und Systemkomponenten in einem Passwort-Manager beschreibt.



**Hauptakteure und Systemkomponenten:**

## **Nutzer:** Der Benutzer des Passwort-Managers.

## **Passwort-Manager:** Die Hauptanwendung zur Verwaltung von Passwörtern.

## **Datenbank Handler:** Die Komponente, die für die Speicherung und Verwaltung der Datenbank verantwortlich ist.

## **Web-Site: D**ie Webseite, mit der der Passwort-Manager interagiert.

## **Verschlüsselung:** Die Komponente, die für die Verschlüsselung und Entschlüsselung von Passwörtern zuständig ist.

## **Hauptinteraktionen:**

## **Passwort erstellen:** Der Nutzer erstellt ein neues Passwort.

## **Passwort eingeben:** Der Nutzer gibt das Passwort ein.

## **Bestätigen:** Der Nutzer bestätigt das eingegebene Passwort.

## **Passwort ändern:** Der Nutzer ändert das Passwort.

## **Nach Passwort suchen:** Der Nutzer sucht nach einem Passwort.

## **Passwort speichern:** Das Passwort wird in der Datenbank gespeichert.

## **Sich neu registrieren:** Der Nutzer registriert sich neu.

## **Passwort anfragen:** Der Nutzer fragt nach einem neuen Passwort.

## **Richtigkeit überprüfen:** Die Richtigkeit des Passworts wird überprüft.

## **Änderungsanfrage senden:** Der Nutzer sendet eine Anfrage zur Änderung des Passworts.

## **Änderungen speichern:** Die Änderungen werden in der Datenbank gespeichert.

## Das Diagramm zeigt die Reihenfolge der Nachrichten und Aktionen, die zwischen den Akteuren und Systemkomponenten stattfinden, um verschiedene Funktionen des Passwort-Managers zu realisieren. Es hilft, die Abläufe und Interaktionen im System zu visualisieren und zu verstehen.

**4. Iterative Entwicklung der Klassen**

**Ziel:** Die iterative Entwicklung der Klassen und Objekte ermöglicht eine schrittweise Verfeinerung und Anpassung des Modells basierend auf den Anforderungen und der Analyse.

**Prozess:**

1. **Anforderungsanalyse:**
2. Jede Anforderung wird geprüft, um festzustellen, ob sie die Beschreibung eines Objekts oder einer Klasse enthält.
3. Nomen werden für Objekte und Adjektive für Exemplarvariablen verwendet.
4. **Erste Iteration:**
   1. **Klasse User:**

**Attribute:** \_username, \_passwordHash, \_password, myEntrys, oldPasswords.

**Methoden:** \_\_init\_\_, addPassword, hashPw, getPasswordHash, setPasswordHash, getUsername, setUsername, getPassword, setPassword.

* 1. **Klasse Entry:**

**Attribute:** \_password, \_username, \_url, \_notice.

**Methoden:** \_\_init\_\_, getPassword, setPassword, getUsername, setUsername, getUrl, setUrl, getNotice, setNotice.

* 1. **Klasse Logic:**

**Attribute:** allUsers, alphabetLowercase, alphabetUppercase, numberList, symbolsList, completeList.

**Methoden:** tryToLogin, alreadyUsed, checkPasswordLength, addNewUser, deleteUser, updateUser, addOldPassword, isOldPassword, addEntry, deleteEntry, updateEntry, goThroughEntry, generatePassword, checkIfSimmilarEntryExists, getSafetyLevel, getAmmountOfDifferentChars, combinationPoints, containsLower, containsUpper, containsNumber, containsSymbol.

* 1. **Klasse Main:**

**Attribute:** functions, loggedUser, generatedEntry, usersFilePath.

**Methoden:** \_\_init\_\_, start, login, createAccount, openMainPage, createEntry, searchForEntry, generatePassword, openPersonalSite, listAllEntrys, writeFile, encrypt, decrypt, readFile.

1. **Zweite Iteration:**
   1. **Verfeinerung der Klassen:**
      1. Hinzufügen neuer Attribute und Methoden basierend auf weiteren Anforderungen und der Analyse.
      2. Beispiel: Klasse User könnte zusätzliche Methoden zur Verwaltung von Sicherheitsfragen oder Zwei-Faktor-Authentifizierung erhalten.
2. **Dritte Iteration:**
   1. **Optimierung und Validierung:**
      1. Überprüfung der Klassen auf Redundanzen und Konsistenz.
      2. Beispiel: Entfernen von Klassen ohne Methoden und Exemplarvariablen.
      3. Validierung der Klassen und Methoden durch Sequenzdiagramme.
3. **Kontinuierliche Verbesserung:**
   1. **Feedback und Anpassung:**
      1. Basierend auf Benutzerfeedback und Tests werden weitere Anpassungen vorgenommen.
      2. Beispiel: Verbesserung der Methode generatePassword in der Klasse Logic, um stärkere Passwörter zu generieren.