|  |
| --- |
| Florian Fritz |
| Lego-Set-Verwaltungssystem |
| Pflichten und Lastenheft |

|  |
| --- |
| Florian Emilio Fritz  22.1.2025 |

Inhalt

[1. Einführung 2](#_Toc188374948)

[2. Ist-Situation 2](#_Toc188374949)

[3. Soll-Situation 2](#_Toc188374950)

[3.1 Soll-Zustand 2](#_Toc188374951)

[3.2 FunktionaleAnforderungen 3](#_Toc188374952)

[3.3 Nicht-Funktionale Anforderungen 3](#_Toc188374953)

[3.4 Schnittstellen 3](#_Toc188374954)

[3.5 Risiken 3](#_Toc188374955)

[4. Abnahmekriterien 4](#_Toc188374956)

[5. Use-Case-Diagramme 5](#_Toc188374957)

[6. Projektplan 7](#_Toc188374958)

[7. Produktumgebung 7](#_Toc188374959)

[8. Skizze von GUI oder Webseite 8](#_Toc188374960)

[9. DB-Entwurf 10](#_Toc188374961)

[10. Link zu einem gehosteten Git-Repository 10](#_Toc188374962)

[11. Testplan 10](#_Toc188374963)

## Einführung

Das Projekt wird im Rahmen eines internen Entwicklungsprojekts durchgeführt.

Das Projekt soll die Verwaltung von Lego-Sets vereinfachen. Das Programm ist grundsätzlich für jeden Lego-Fan mit besonderer Zielgruppe von Sammlern, die ihre Sammlung/Inventar verwalten wollen, um eine besseren Übersicht zu haben. Durch eine zusätzliche Datenbank lassen sich Daten perfekt abspeichern.

## Ist-Situation

Als großer Lego-Fan mit einem größeren Inventar kann es schnell passieren das man den überblick seiner Sets verliert. Hier kommt das Projekt zum Einsatz, es soll dem Nutzer die Verwaltung seiner Sammlung vereinfachen. Vorher haben viele es mit einer Tabelle oder auf Papier von Hand zu Fuß gemacht, was bei größeren Sammlungen schnell zu Fehlern führen kann. Dies soll das Programm erleichtern.

## Soll-Situation

3.1 Soll-Zustand

Nach Abschluss des Projekts wird eine einfache Software bereitstehen, mit der Lego-Sammler ihre Sets verwalten können. Die Software soll den Wunsch erfüllen, alle Informationen zu einer Sammlung an einem zentralen Ort zu speichern und leicht abrufbar zu sein.

Mit der Anwendung können Nutzer ihre Sets erfassen, Informationen wie Name, Nummer, Thema und Preis(UVP) speichern sowie den Gesamtwert der Sammlung automatisch berechnen lassen. Durch die Anbindung an eine externe Datenbank (Rebrickable) wird es möglich sein, Sets direkt zu suchen und hinzuzufügen, was Zeit spart, und die manuelle Eingabe reduziert.

Die Vorteile der Software liegen vor allem darin, dass Sammler ihre Sammlung besser im Blick haben und schneller Änderungen vornehmen können.

Insgesamt bietet die Software eine sinnvolle Lösung, um die Verwaltung einer Lego-Sammlung zu erleichtern und übersichtlicher zu gestalten.

### 3.2 FunktionaleAnforderungen

|  |  |  |
| --- | --- | --- |
| *Funktion* | *Beschreibung* | *Aufwand (Stunden)* |
| Set hinzufügen | Nutzer können neue Lego-Sets manuell anlegen oder über eine API-Suche hinzufügen. | 12 |
| Set löschen | Sets können aus der Sammlung entfernt werden. | 8 |
| Sets durchsuchen | Sets per Nummer suchen | 12 |
| Inventarwert berechnen | Gesamtwert des Inventars Berechnen | 8 |
| Integration einer API | Integration von API(Rebrickable) für Daten abruf | 20 |
| Datenbankanbindung | Verbindung zur Datenbank zum Speichern und Abrufen von Sets. | 15 |

### 3.3 Nicht-Funktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| *Funktion* | *Beschreibung* | *Aufwand (Stunden)* |
| GUI erstellen | Eine Intuitive GUI erstellen | 25 |

3.4 Schnittstellen

**-Rebrickable API**: Sucht Lego-Sets anhand eines Namens oder einer Nummer.

3.5 Risiken

|  |  |  |
| --- | --- | --- |
| Risiko | Verantwortlicher | Alternative Lösung |
| API nicht verfügbar | Entwickler | Manuelles Hinzufügen von Sets als Alternative anbieten.  API Daten in der Datenbank speichern |
| Datenbankzugriffsprobleme | Entwickler | Lokale Backups erstellen |

## Abnahmekriterien

**Muss-Kriterien**

1. **Sets verwalten**
   * Sets können hinzugefügt und gelöscht werden.
2. **Inventarwert anzeigen**
   * Der Gesamtwert der Sammlung wird korrekt berechnet.
3. **Filterfunktion nach Nummer**
   * Sets können gezielt anhand ihrer Set-Nummer gefiltert werden.
4. **Benutzeroberfläche**
   * Die Software verfügt über eine funktionsfähige und intuitive grafische Benutzeroberfläche.
5. **Login Service**
   * Man kann sich registrieren und Anmelden

**Kann-Kriterien**

1. **Exportfunktion**
   * Die Sammlung kann als PDF exportiert werden.
2. **Zulätzt hinzugefügt**
   * Eine Fenster das die Zuletzt hinzugefügten Sets anzeigt
3. **Erweiterte Filterfunktionen**
   * Weitere Filtermöglichkeiten wie nach Namen, Thema oder Jahr.
4. **Dynamische GUI**
   * Die GUI passt sich unterschiedlichen Bildschirmgrößen an.
5. **Passwort zurücksetzen**
   * Das Passwort via. Email zurück setzen

## Use-Case-Diagramme

Ein Bild, das Diagramm, Zeichnung, Entwurf, Lineart enthält.

Automatisch generierte Beschreibung

Lego Set Verwaltungssystem

Tabellarische Beschreibung der Use-Cases

|  |  |
| --- | --- |
| Use Case | Beschreibung |
| Set hinzufügen | Der Benutzer fügt ein Lego-Set über die API-Suche hinzu. Hierfür werden die Daten über den Use-Case „API-Daten abrufen“ abgerufen und gespeichert. |
| Set manuell hinzufügen | Der Benutzer gibt die Informationen zu einem Set manuell ein, ohne die API zu nutzen. Die eingegebenen Daten werden über „Daten speichern“ gespeichert. |
| Set aus dem Inventar löschen | Der Benutzer kann ein Set aus seinem Inventar entfernen. Dabei werden die Änderungen über den Use-Case „Daten speichern“ in der Datenbank aktualisiert. |
| Inventar Gesamtwert anzeigen | Der Benutzer kann den Gesamtwert seines Inventars basierend auf den gespeicherten Sets anzeigen lassen. |
| API-Daten abrufen | Für „Set hinzufügen“ werden die Daten über die API-Schnittstelle abgerufen, um automatisch Informationen wie Name, Nummer und Preis einzutragen. |
| Daten speichern | Die zentralen Datenänderungen (Hinzufügen, Löschen, Bearbeiten) werden in der Datenbank gespeichert. |
| Login mit Registrierung | Man kann sich anmelden und Registrieren |
| (optional) zuletzt hinzugefügt | Man kann die letzten gespeicherten Einträge anschauen |
| Inventar Anzeigen | Man kann sich das Inventar in einer Tabelle anzeigen |
| (Optional) Exportfunktion via PDF | Der Benutzer kann seine Sammlung in einem PDF-Format exportieren. |

Beziehungen zwischen Use-Cases

|  |  |
| --- | --- |
| Beziehung | Beschreibung |
| Set hinzufügen => API-Daten abrufen | Set hinzufügen ruft immer die API-Daten auf, um die benötigten Set-Informationen zu laden. |
| API-Daten abrufen => Daten Speicher | API-Daten abrufen ruft Daten Speicher auf um die API daten langfristig zu speichern das bewirkt keine langen API-Abfragen so wie ein kleines Backup falls die API mal nicht verfügbar ist |
| Set hinzufügen => Daten speichern | Nach dem Hinzufügen eines Sets werden die Daten in der Datenbank gespeichert. |
| Set manuell hinzufügen => Daten speichern | Set manuell hinzufügen ruft Daten speichern auf, um die eingegebenen Daten dauerhaft zu speichern. |
| Set aus dem Inventar löschen => Daten speichern | Beim Löschen eines Sets aus dem Inventar wird Daten speichern verwendet, um die Änderungen zu aktualisieren. |
| Login mit Registrierung => Daten speichern | Login und Registrierung rufen Daten Speicher auf um bei Registrierung Daten zu speichern und beim Login Daten Vergleichen |
| Inventar anzeigen => Daten speichern | Um zu schauen welche Sets in seinem Inventar sind |
| Inventar Gesamtwert anzeigen => Daten speichern | Der Gesamtwert wird anhand der gespeicherten Daten in der Datenbank berechnet. |

## Projektplan

Ein Bild, das Text, Screenshot, Reihe, Diagramm enthält.

Automatisch generierte Beschreibung

## Produktumgebung

* **C#**:

Das Projekt wird in C# programiert

* **WPF**:

WPF wird benutzt, um eine benutzerfreundliche GUI zu schaffen

**Datenbank**

* **SQLite**:

Wird verwendet, um die Daten dauerhaft zu speichern.

* **Rebrickable API**:

Eine externe Schnittstelle, die genutzt wird, um Daten zu Lego-Sets wie Name, Nummer, Jahr und Thema abzurufen. Dies erleichtert das Hinzufügen von Sets und reduziert manuelle Eingaben.

* **Visual Studio**:
  + Das ganze Projekt wird in Visual Studio geschrieben, getestet und gedebugget
* **Git/GitHub**:
  + Zur Versionierung des Projekts wird Git/GitHub benutzt. Das Projekt wird in einem GitHub-Repository gehostet.

.

## Skizze von GUI oder Webseite

Ein Bild, das Text, Screenshot, Software, Display enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Display enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Display enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

Automatisch generierte Beschreibung

## DB-Entwurf

Ein Bild, das Entwurf, Zeichnung, Lineart, Diagramm enthält.

Automatisch generierte Beschreibung

.

## Link zu einem gehosteten Git-Repository

https://github.com/FloFritz/Lego-Set-Verwaltungssystem

## Testplan

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Schrift, Reihe, Zahl enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Reihe, Schrift, Zahl enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Reihe, Schrift enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Reihe, Schrift enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

UnitTests  
**1. Registrierung**

* **Ziel**: Sicherstellen, dass die Registrierung fehlerfrei funktioniert.
* **Testfälle**:
  + Erfolgreiche Registrierung eines neuen Benutzers.
  + Fehler bei der Registrierung mit bereits existierendem Benutzernamen oder E-Mail.

**2. Login**

* **Ziel**: Prüfen, ob Benutzer erfolgreich eingeloggt werden und Fehler korrekt behandelt werden.
* **Testfälle**:
  + Erfolgreicher Login mit gültigen Benutzerdaten.
  + Fehler bei falschem Passwort.
  + Fehler bei nicht existierendem Benutzer.

**3. Inventarverwaltung**

* **Ziel**: Sicherstellen, dass Sets korrekt hinzugefügt, gelöscht und angezeigt werden.
* **Testfälle**:
  + Hinzufügen eines neuen Sets, sowohl über die API als auch manuell.
  + Verhindern von doppelten Einträgen im Inventar.
  + Löschen eines Sets aus dem Inventar.

**4. Berechnung des Inventarwerts**

* **Ziel**: Überprüfen, ob der Gesamtwert des Inventars korrekt berechnet wird.
* **Testfälle**:
  + Korrekte Berechnung des Wertes bei mehreren Sets.
  + Fehlerfreies Verhalten bei leerem Inventar.

**5. (Optional) Anzeige der zuletzt hinzugefügten Sets**

* **Ziel**: Sicherstellen, dass die letzten 5 hinzugefügten Sets korrekt angezeigt werden.
* **Testfälle**:
  + Anzeige der Sets in der korrekten Reihenfolge (nach Hinzufügedatum).
  + Korrekte Anzeige bei weniger als 5 vorhandenen Sets.

**6. (Optional) Passwort zurücksetzen**

* **Ziel**: Prüfen, ob Benutzer ihr Passwort sicher zurücksetzen können.
* **Testfälle**:
  + Senden einer E-Mail mit einem Reset-Link.