

ABSCHLUSSPRÜFUNG SOMMER 2022

AUSBILDUNG ZUM FACHINFORMATIKER

FACHRICHTUNG: ANWENDUNGSENTWICKLUNG

PRÜFLING

Eugen Maksymenko Schillerstraße3

90409 Nürnberg

Identnummer: 3608904

AUSBILDUNGSBETRIEB

SUSE Software Solutions Germany GmbH Maxfeldstraße 5

90409 Nürnberg



SUSE

Dokumentationstools für die Verwaltung und Erstellung von wöchentlichen Berichten

ABGABEDATUM: 15. MAI 2022

Inhaltsverzeichnis

[1. Einleitung 2](#_Toc103288114)

[1.1 Vorwort 2](#_Toc103288115)

[1.2 Projektname 2](#_Toc103288116)

[2. Definitionsphase 2](#_Toc103288117)

[2.1 Projektumfeld 2](#_Toc103288118)

[2.2 Auftraggeber 2](#_Toc103288119)

[2.3 Problemstellung 2](#_Toc103288120)

[2.4 Projektziel 2](#_Toc103288121)

[2.5 Projektabgrenzung 3](#_Toc103288122)

[2.6 IST-Analyse 3](#_Toc103288123)

[2.7 Soll-Konzept 3](#_Toc103288124)

[3. Planungsphase 3](#_Toc103288125)

[3.1 Zeitplan 3](#_Toc103288126)

[3.2 Ressourcenplanung 4](#_Toc103288127)

[3.2.1 Hardware 4](#_Toc103288128)

[3.2.2 Software 4](#_Toc103288129)

[3.2.3 Personal 4](#_Toc103288130)

[3.3 Projektkosten 5](#_Toc103288131)

[3.4 Amortisation/Vorteile 5](#_Toc103288132)

[4. Entwurfsphase 5](#_Toc103288133)

[4.1 Kommunikation zwischen Benutzter und dem Tool 5](#_Toc103288134)

[4.1.1 Interaktion ohne Kommandozeilenargumente 5](#_Toc103288135)

[4.1.2 Interaktion mit Kommandozeilenargumenten 7](#_Toc103288136)

[4.2 Datenbank 7](#_Toc103288137)

[4.3 Layout des Berichts 7](#_Toc103288138)

[4.4 Module 7](#_Toc103288139)

[4.4.1 Kommandozeilen-Modul 7](#_Toc103288140)

[4.4.2 Datenbank-Modul 7](#_Toc103288141)

[4.4.3 Berichts-Modul 7](#_Toc103288142)

[4.5 Planung der Testszenarien 7](#_Toc103288143)

[4.6 Pflichtenheft 8](#_Toc103288144)

[Konfiguration von Nutzerdaten 8](#_Toc103288145)

[5. Implementierung 8](#_Toc103288146)

[5.1 Kommandozeilen Modul 8](#_Toc103288147)

[5.1.1 Kommandozeilen Interaktion mit Menü ohne Unterbefehle 9](#_Toc103288148)

[5.1.2 Kommandozeilen-Interaktion ohne Menü, mit Unterbefehl 9](#_Toc103288149)

[5.2 Datenbank Modul 10](#_Toc103288150)

[5.3 Berichts-Modul 10](#_Toc103288151)

[5.4 Vorlage für das Bericht-Modul 11](#_Toc103288152)

[6. Projektkontrolle 11](#_Toc103288153)

[6.1 Zeitlicher Soll-Ist-Vergleich 11](#_Toc103288154)

[6.2 Ziel-Vergleich 12](#_Toc103288155)

[6.3 Kostenvergleich 12](#_Toc103288156)

[6.4 Testdurchführung 13](#_Toc103288157)

[6.4.1 Code Review als White-Box-Test 13](#_Toc103288158)

[6.4.2 Systemtest als Black-Box-Test 13](#_Toc103288159)

[7. Projektabschluss 13](#_Toc103288160)

[7.1 Übergabe an Kunden/Team 13](#_Toc103288161)

[7.2 Fazit 13](#_Toc103288162)

[7.2.1 Projekt umgesetzt 13](#_Toc103288163)

[7.2.2 Zufriedener Auftraggeber 13](#_Toc103288164)

[7.2.3 Gelerntes Wissen 13](#_Toc103288165)

[7.3 Ausblick 13](#_Toc103288166)

[8. Glossar 15](#_Toc103288167)

[Anhang A: Mockup Erstes Berichts Layout 17](#_Toc103288168)

[Anhang B: Mockup Endgültiges Berichts Layout 18](#_Toc103288169)

[Anhang C: Detaillierte Zeitplanung 18](#_Toc103288170)

[Anhang D: Datenbank Modell 19](#_Toc103288171)

[Anhang E: Klassendiagramm 20](#_Toc103288172)

[Anhang F: Zusätzliche Bearbeitungszeit 20](#_Toc103288173)

**Abbildungsverzeichnis**

[Abbildung 1 Grobe Zeitplanung 4](#_Toc103284743)

[Abbildung 2 Mockup Hauptmenü 6](#_Toc103284744)

[Abbildung 3 ER Modell Datenbank 8](#_Toc103284745)

[Abbildung 4 Implementierung des Menü Interfaces 10](#_Toc103284746)

[Abbildung 5 Implementierung eines Unterbefehls mit zwei Argumenten 11](#_Toc103284747)

[Abbildung 6 Implementierung der privaten Methode für die Tabellenerstellung 11](#_Toc103284748)

[Abbildung 7 Implementierung eines Dictionary für das Berichts-Modul 12](#_Toc103284749)

[Abbildung 8 HTML Vorlage 12](#_Toc103284750)

[Abbildung 9 Soll-IST-Vergleich 13](#_Toc103284751)

[Abbildung 10 Vorläufiges Berichtsmodell 18](#_Toc103284752)

[Abbildung 11 Endgültiges Berichtsmodell 19](#_Toc103284753)

[Abbildung 12 Datenbank Modell 20](#_Toc103284754)

[Abbildung 13 Database Klassendiagramm 21](#_Toc103284755)

**Tabellenverzeichnis**

Tabelle 1 Kategorien im wöchentlichen Bericht 3

Tabelle 2 Benutzte Software 5

Tabelle 4 Pflichtenheft 9

Tabelle 5 Ziel-Vergleich 13

Tabelle 6 Kostenvergleich 14

# 1. Einleitung

Das vorliegende Projekt wurde vom Autor dieser Dokumentation, Herrn Eugen Maksymenko im Rahmen der Abschlussprüfung zum Fachinformatiker für Anwendungsentwickler im Sommer 2022 durchgeführt.

## 1.1 Vorwort

Innerhalb dieser Dokumentation sind *kursiv* formatierte oder abgekürzte Begriffe zu finden, welche bei erster Nennung ausführlich und alphabetisch im Glossar aufgeführt werden

## 1.2 Projektname

Bei diesem Projekt handelt es sich um ein *Open Source* Projekt. Darunter ist zu verstehen das jeder Mensch dieses Projekts für jeden Zweck verwendet kann. Es ist erlaubt das Projekt zu inspizieren, modifizieren und weiterverbreiten. Nach der finalen Version dieses Projektes ist es geplant das Projekt auf GitHub zu veröffentlichen. Da jedes Projekt einen guten Namen erfordert und „Dokumentationstools für die Verwaltung und Erstellung von wöchentlichen Berichten“ nicht sprechend ist, wird das Projekt den Titel „Reportrick“ erhalten. Auch innerhalb dieses Projektdokumentation wird dieser Namen verwendet.

# 2. Definitionsphase

## 2.1 Projektumfeld

Die SUSE Software Solutions Germany GmbH, kurz SUSE, ist ein im Jahr 1992 in Fürth gegründetes Softwareunternehmen und entwickelt unter anderem eigene Linux Distributionen mit dem Namen SUSE Linux Enterprise (SLE) und der Community Version openSUSE. Eine Linux Distribution ist eine Zusammenstellung von über 5000 Programmen und Bibliotheken, die die alle unter einer Open Source Lizenz stehen und frei erhältlich sind. Diese aufeinander abgestimmten Komponenten bilden ein Allzweckbetriebssystem. Bei SUSE arbeiten weltweit ca. 2200 Mitarbeiter an verschiedenen Standorten auf unterschiedlichen Kontinenten, z.B. in Beijing, Vancouver und Prag. Die gesamte Projektumsetzung findet in der SUSE Software Solutions Germany GmbH im Standort Nürnberg statt.

## 2.2 Auftraggeber

Der Auftraggeber dieses Projektes ist das Dokumentation-Team der SUSE. Dieses Team entwickelt das DocBook Authoring and Publishing Suite (DAPS) Projekt und schreibt Dokumentation für SUSE-Produkte. Auftraggeber und Betreuer des Projekts ist ein Teammitglied des Dokumentationsteams, Herr Thomas Schraitle.

## 2.3 Problemstellung

Aktuell muss jeder Mitarbeiter eine eigene Lösung suchen, für die Erfassung seiner Aktivität im  
Team. Das hat den großen Nachteil, dass dadurch oft Details und Tätigkeiten verloren gehen, da  
die Berichte meistens erst gegen Ende der Woche, aus dem Gedächtnis angefertigt werden.

Uneinheitliche oder spät eingereichte Berichte erschweren und verlangsamen zusätzlich das Zusammenfassen und die Erstellung eines einheitlichen Berichtes für den Team-Leiter.

## 2.4 Projektziel

Aufgrund des Problems aus 2.3 Problemstellung entstand dieses Projekt.

Das Ziel dieses Projektes besteht darin, ein Kommandozeilentool für die Erfassung der  
wöchentlichen Tätigkeiten und Erstellung von Berichten zu entwickeln. Das Tool soll Einträge in eine lokale Datenbank abspeichern, damit auch bei Bedarf nachträglich auf die Berichte zugegriffen werden kann. Der Nutzer kann Beiträge hinzufügen, löschen und ausgeben. Die Beiträge werden in vordefinierte Kategorien für eine Kalenderwoche abgespeichert. Die so erfassten Daten werden am Ende der Woche zu einem Wochenbericht im gewünschten Format (HTML, PDF und Text) generiert.

## 2.5 Projektabgrenzung

* Die Funktion einer automatisierten E-Mail-Versendung nach Berichtserstellung ist nicht ein Teil dieses Projektes.
* Die Erstellung eines offiziellen RPM Paketes für openSUSE ist auch nicht ein Teil dieses Projektes.
* Das Veröffentlichen auf der Plattform GitHub ist durch zusätzlichen Aufwand kein Teil dieser Dokumentation.

## 2.6 IST-Analyse

Nach mehreren Gesprächen mit verschiedenen Teamleitern und der Auswertung eines ausgefüllten Fragebogens ergibt sich folgende Situation:

Mitarbeiter müssen in ihrem Team einen wöchentlichen Bericht für die aktuelle Kalenderwoche anfertigen. Der Bericht enthält Einträge mit ihren Tätigkeiten in verschiedenen Kategorien.

Die Kategorien, ihre Bedeutung und passende Beispiele sind in der Tabelle 1 dargestellt.

|  |  |  |
| --- | --- | --- |
| **Kategorie** | **Bedeutung** | **Beispiel** |
| RED | Problembehaftete Tätigkeiten und „Stopper“ | Festplatte defekt. Kein Release möglich |
| AMBER | Problematische Tätigkeiten | Internetausfall 2 Stunden |
| GREEN | Erledigte Tätigkeiten, die normal verlaufen sind. | Feature Erweiterung in Projekt XY implementiert |
| MEETING | Meetings, Team Events, Besprechungen | Code Review mit Mitarbeiter XY |
|  |  |  |

Tabelle 1 Kategorien im wöchentlichen Bericht

Der wöchentliche Bericht ist jeden Freitag bis 13 Uhr abzugeben. Der Inhalt bezieht sich immer auf die aktuelle Kalenderwoche von Montag-Freitag. Nach der Erstellung sendet der Mitarbeiter seinen Bericht per E-Mail an den Teamleiter. Dieser Teamleiter liest jeden einzelnen wöchentlichen Bericht. Alle Berichte der Mitarbeiter werden vom Teamleiter zu einem gemeinsamen Bericht zusammengefasst. Dadurch existiert ein schriftlicher Überblick über die aktuelle Situation im Team, inklusive aller aktuellen Komplikationen. Die Zusammenfassung verbraucht 0.5-2 Stunden, je nach Teamgröße und aktuelle Auslastung. Der sogenannte Teambericht (Zusammenfassung und Sortierung der Mitarbeiterberichte) wird an das höhere Management gesendet. Außerdem werden besondere Leistungen von Mitarbeitern in diesem Bericht hervorgehoben. Ein Tool für Mitarbeiter, der die Erfassung der Tätigkeiten erleichtert und die automatische Berichtserstellung ermöglicht, wird dringend gewünscht. Einheitliche Berichte eröffnen viele weiteren Optionen zur Automatisierung dieses Prozesses.

## 2.7 Soll-Konzept

Ein Kommandozeilen-Tool unter einer Open Source-Lizenz zum Erfassen der Tätigkeiten und Generierung von wöchentlichen Berichten soll realisiert werden.

Die Anpassung muss…

* ein Kommandozeilen-Menü anbieten.
* Interaktion direkt durch Kommandozeilen Argumente anbieten.
* eine Datenbankanbindung haben.
* eine HTML und CSS Vorlage besitzen.
* automatisiert ein Bericht erstellen können in mehreren Formaten (HTML/PDF/TEXT)
* muss modular aufgebaut sein, damit weitere Funktionalität „einfach“ erweiterbar sind.

# 3. Planungsphase

## 3.1 Zeitplan

Für das Abschlussprojekt stehen insgesamt 70 Stunden zur Verfügung. Um einen Überblick über den zeitlichen Aufwand zu erhalten, wurde das Projekt in folgende Phasen unterteilt, welche während des Projektes durchlaufen werden. Die grobe Zeitplanung lässt sich aus der folgen Grafik entnehmen.

Abbildung 1 Grobe Zeitplanung

Die genaue Aufteilung der Stunden ist Tabelle 3 Detaillierte Zeitplanung finden.

## 3.2 Ressourcenplanung

Eine Sammlung aller benötigten Ressourcen für eine erfolgreiche Durchführung des Projekts

### 3.2.1 Hardware

Büroarbeitsplatz mit Laptop, Monitor, Kopfhörer, Maus und Tastatur.

### 3.2.2 Software

|  |  |
| --- | --- |
| **Name** | **Verwendungszweck** |
| openSUSE Tumbleweed | Betriebssystem |
| Visual Studio Code | Quelltext-Editor |
| Git | Versionskontrolle |
| GitHub | Webbasierter Filehosting-Service für die Softwareentwicklung und das Teilen von Softwarecode |
| Python | Interpretierte Programmiersprache |
| Jinja2 | Web-Template-Engine für die Programmiersprache Python |
| HTML | Textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks |
| CSS | Stylesheet-Sprache für elektronische Dokumente und zusammen mit HTML |
| SQLite | Dateibasierte Datenbank |
| Draw.io | Erstellung von Diagrammen und Mockups |
| Microsoft Word | Erstellung der Dokumentation und Tabellen |

Tabelle 2 Benutzte Software

### 3.2.3 Personal

Mitarbeiter:

* Absprache
* Festlegung der Anforderungen
* Konzeption und Entwicklung
* Code Review

## 3.3 Projektkosten

In diesem Abschnitt werden die voraussichtlichen Kosten berechnet, die bei der Durchführung des Projektes anfallen. Da es sich um ein firmeninternes Projekt handelt, kann man keine pauschalen Kosten angeben, die normalerweise einem Kunden in Rechnung gestellt werden. An diesem Projekt soll ein Prüfling im dritten Lehrjahr voraussichtlich 70h arbeiten. Nach einer Anfrage in der Personalabteilung des Unternehmens verbraucht der Azubi 9.34 € pro Stunde. Darin sind pauschale Gemeinkosten wie Strom und Mietkosten ebenfalls enthalten. Nach der Umsetzung werden mehre *Code Reviews* von Herrn Schraitle durchgeführt. Der interne Stundensatz von einem Entwickler beträgt durchschnittlich 30 €.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vorgang | Mitarbeiter | Stundensatz | Zeit | Kosten |
| Entwicklungskosten | 1x Auszubildender | 9.34 € | 70 h | 653.80 € |
| Code Review | 1x Mitarbeiter | 30.00 € | 2 h | 60.00 € |
| **Gesamtkosten** | | | | 713.80 € |

Tabelle 1 Projektkosten

## 3.4 Amortisation/Vorteile

Da das Projekt nicht in direktem Bezug zum Umsatz steht, lässt sich über die Amortisierung der Kosten keine konkrete Aussage treffen. Es lassen sich jedoch offensichtliche Vorteile für das Unternehmen ableiten:

* Die Teamleiter können den aktuellen Status schneller einschätzen und in der Folge darauf präventiv auf Unzulänglichkeiten reagieren
* Die Mitarbeiter des Teams können das neue Tool viel einfacher in ihren Workflow einsetzten
* Die Einträge in den Bericht werden sofort erledigt. Dadurch gehen keine Details verloren.

Aufgrund dieser Vorteile ist abzusehen, dass sich die Verbesserung zügig amortisieren wird, weil sie Arbeitsabläufe beschleunigt und zu einer Vereinfachung des Workflows führt.

# 4. Entwurfsphase

In dieser Phase wird festgelegt, wie die Kommunikation zwischen Benutzter und dem Berichtstool stattfinden soll. Dabei unterscheiden wir nochmal die Interaktion des Anwenders mit und ohne Kommandozeilenargumente. Der nächste große Schritt ist der Entwurf einer relationalen Datenbank für die persistente Speicherung von Daten. Um einheitliche Berichte zu erfassen, ist eine Vorlage für das Layout erforderlich, welches mit den Werten aus der Datenbank befüllt wird. Zuletzt wurde eine Aufteilung in Module vorgenommen und die Schnittstellen zwischen den einzelnen Modulen entworfen. Die Ergebnisse der Entwurfsphase sind in den folgenden Abschnitten aufgeführt.

## 4.1 Kommunikation zwischen Benutzter und dem Tool

Der Nutzer kommuniziert mit dem Tool direkt per Kommandozeile. Wird das Programm ohne Kommandozeilenargumente gestartet, so wird das Hauptmenü aufgerufen.

Die Interaktion mit dem Tool bezieht sich standardmäßig immer auf die aktuelle Woche. Es ist aber möglich, auch auf Berichte aus der Vergangenheit zuzugreifen. Siehe 4.1.1.2 Wechsel der Kalenderwoche oder des Jahres. Alternativ soll eine direkte Übergabe an Befehlen mit Parametern möglich sein. Detaillierte Informationen sind im Abschnitt 4.1.2 Interaktion mit Kommandozeilenargumenten zu finden.

### 4.1.1 Interaktion ohne Kommandozeilenargumente

Im Kommandozeileninterface wird der Nutzer begrüßt. Das aktuelle Datum und die jetzige Kalenderwoche sollen angezeigt werden. Darunter ist ein Hauptmenü für die Interaktion zu sehen. Durch simple Nutzereingabe soll der Anwender durch das Menü navigieren können.

In der folgenden Abbildung ist ein Beispiel für das Hauptmenü dargestellt.

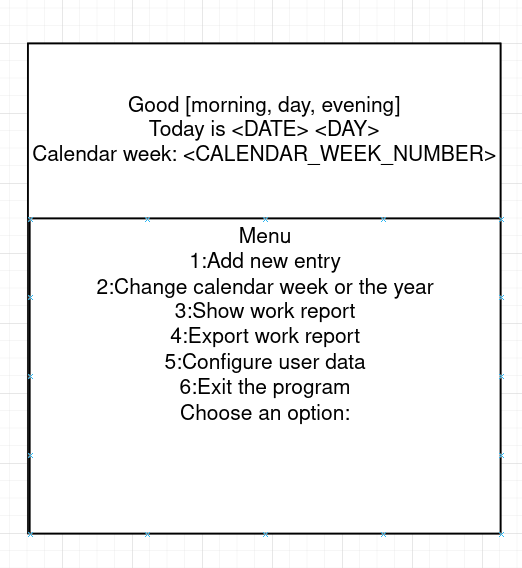


Abbildung 2 Mockup Hauptmenü

Die genauere Funktionsweise der einzelnen Punkte wird unten festgehalten

#### 4.1.1.1 Hinzufügen der neuen Einträge

Unter Nummer 1 kann die Option „Add new entry“ ausgewählt werden. Das beinhaltet das

Hinzufügen von neuen Beiträgen. Beim Auswählen von dieser Option wird der Nutzer aufgefordert dein Eintrag in die Kommandozeile einzutragen. Nach erfolgreicher Eingabe soll der Anwender entscheiden unter welcher Kategorie er den eingegebenen Eintrag abspeichern will. Nach dieser Eingabe wird der Eintrag für die eingestellte Kalenderwoche/Jahr in der Datenbank abgespeichert. Der Nutzer erhält eine Rückmeldung zu den abgespeicherten Daten. Danach kann der Benutzter zum Hauptmenü zurückkehren oder direkt das Programm beenden.

#### 4.1.1.2 Wechsel der Kalenderwoche oder des Jahres

Unter Nummer 2 kann die Option „Change calendar week or the year“ ausgewählt werden.

Damit kann der Nutzer bei Bedarf auf alte Berichte aus der Datenbank zugreifen.   
Der Nutzer soll ein Jahr und eine Kalenderwoche eingeben. Danach kehrt der Nutzer zurück ins Hauptmenü. Im Hauptmenü ist ein Hinweis zu sehen der betont, dass der Benutzer auf einen Bericht aus der Vergangenheit zugreift. Die normale Interaktion mit dem Tool kann weiterhin geführt werden mit dem Unterschied das man auf den alten Bericht zugreift und nicht auf die aktuelle Kalenderwoche. Das Beenden des Programms setzt immer das Programm auf das aktuelle Jahr und die aktuelle Kalenderwoche zurück.

#### 4.1.1.3 Einsicht in den aktuellen Wochenbericht

Unter Nummer 3 kann die Option („Show work report“) ausgewählt werden.

Beim Aufruf werden alle Einträge für diese Woche kategorisiert ausgegeben. Die Kategorien sollen farblich dargestellt werden, für eine bessere Lesbarkeit. Der Benutzer bekommt die Möglichkeit, dass Programm zu beenden und zum Hauptmenü zurückzukehren. Da der Nutzer die Ansicht aller Einträge vor sich hat, kann er als weitere Möglichkeit Einträge löschen.

Durch die Auswahl dieser Option im Untermenü wird der Anwender aufgefordert eine Kategorie auszusuchen, in welcher der zu löschende Eintrag sich befindet. Durch die Auswahl werden aus der Datenbank alle Einträge in der zugehörigen Kategorie angezeigt. Der Nutzer muss den zu löschenden Eintrag eingeben. Damit soll verhindert werden, dass ein unbeabsichtigter Eintrag gelöscht wird. Nach der Bestätigung dieser Eingabe wird der Eintrag gelöscht und der Anwender bekommt eine aktualisierte Ansicht aller Einträge. Bei Bedarf kann er diesen Vorgang wiederholen, das Programm beenden oder ins Hauptmenü zurückkehren.

#### 4.1.1.4 Erstellung des Wochenberichtes

Unter Nummer 4 kann die Option („Export work report“) ausgewählt werden. Diese Funktion dient dazu, den Wochenbericht für die aktuell ausgewählte Kalenderwoche und Jahr zu erstellen.

Nach der Auswahl dieser Option wird der Benutzer gefragt, in welchem Format dieser Bericht erstellt werden soll. Ein Untermenü wird auf der Kommandozeile ausgegeben, wo der Anwender zwischen drei benötigten Formaten, wie HTML, PDF und Text entscheiden. Nach der Selektierung des Formates werden alle Einträge aus der Datenbank zu einem Bericht erstellt. Eine Rückmeldung soll den Pfad angeben, in dem der Wochenbericht abgespeichert worden ist.

#### 4.1.1.5 Konfiguration der Nutzerdaten

Unter Nummer 5 kann die Option („Configure user data“) ausgewählt werden. Der Nutzer kann hier seine eigenen Daten wie Vornamen, Nachname und das Team, in dem er arbeitet, festhalten. Wenn der Nutzer eine Konfiguration vornimmt, werden diese Daten auch in den Bericht aufgenommen.

#### 4.1.1.6 Beenden des Programms

Unter Nummer 6 kann die Option („Exit the program“) ausgewählt werden. Das Programm wird beendet.

### 4.1.2 Interaktion mit Kommandozeilenargumenten

Die Befehle des Nutzers an das Tool müssen eine festgelegte Syntax einhalten, damit das Tool diese als Befehl interpretieren kann. Der Befehl „help“ zeigt dem Nutzer eine Bedienungsanleitung, die alle verfügbaren Befehle auflistet und beschreibt. Es soll möglich seine Einträge hinzuzufügen und zu löschen. Der Nutzer soll seine Daten, wie Name und Teamname konfigurieren können, damit sie im Bericht integriert werden. Aufruf von vergangenen Wochenberichten ist in dieser Art der Interaktion noch nicht geplant.

## 4.2 Datenbank

Das Tool benötigt eine eigene Datenbank, in welcher alle Nutzerdaten und Einträge abgespeichert werden.  
Abbildung 3 zeigt ein ER Diagramm ohne Attribute.

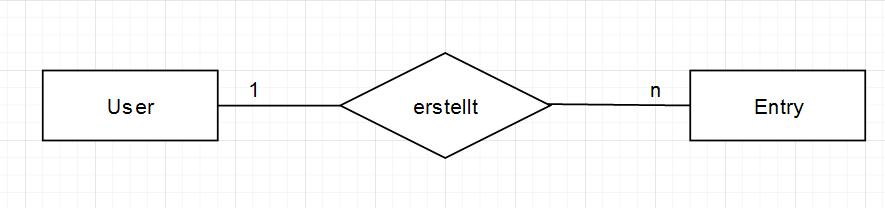


Abbildung 3 ER Modell Datenbank

Eine genaue Darstellung ist Abbildung 10 Datenbank Modell enthalten

## 4.3 Layout des Berichts

Für den wöchentlichen Bericht ist eine Vorlage notwendig. In dieser Struktur sollen Platzhalter fest definiert werden, die dynamisch mit Einträgen aus der Datenbank befüllt werden können.

Nach der Aufnahme der Kundenwünsche wurde ein Modell für das Layout angefertigt.   
Siehe Abbildung 10 Vorläufiges Berichtsmodell

Nach Rücksprache mit dem Auftraggeber wurden einige Änderungen vorgenommen. Dadurch entstand dann das finale Modell: Abbildung 11 Endgültiges Berichtsmodell

## 4.4 Module

Zur besseren Wart und Erweiterbarkeit wird Reportrick in Programmmodule aufgeteilt. Mehrere kleinere Module können leichter verstanden und verändert werden, als eine große Datei, in der die gesamte Programmlogik enthalten ist.

### 4.4.1 Kommandozeilen-Modul

Das „Main“-Modul beinhaltet die Logik für die Interaktion mit der Kommandozeile. Beim Programmstart wird dieses Modul aufgerufen.

### 4.4.2 Datenbank-Modul

Das Datenbank-Modul enthält die Logik der Datenbank in einer Klasse. Siehe Beispiel unter Abbildung 5 Database Klassendiagramm.

### 4.4.3 Berichts-Modul

Das Berichts-Modul ist für die Erstellung von Berichten in verschiedenen Formaten verantwortlich. Bei Bedarf ist die Erweiterung um zusätzliche Formate ohne Probleme möglich.

## 4.5 Planung der Testszenarien

Um möglichst effizient in der Testphase zu sein, werden die Testszenarien bereits in der Planungsphase entworfen. Dazu werden die erwarteten Funktionalitäten festgelegt. Diese Punkte werden in der Testphase als Black-Box-Test durchlaufen und die Ergebnisse werden im Feld „Ergebnis“ protokolliert. Ein Beispiel der geplanten Testszenarien lässt sich aus der folgenden Tabelle entnehmen

|  |  |
| --- | --- |
| **Anpassung** | **Erwartete Funktionalität** |
| Hinzufügen von Daten in der Datenbank | Die Daten werden korrekt abgespeichert. |
| Löschen der Daten in der Datenbank | Die Daten werden erfolgreich aus der Datenbank gelöscht |
| Erstellung des Berichtes | Die Datei wurde erstellt |
| Überprüfung der Einträge | Der Bericht hat alle Einträge aus der Datenbank eingespeichert. |
| Layout Format einhalten | Überprüfung der Schriftgröße, Zeilenumbrüche und des Layouts. |

Tabelle 2 Beispiel Testszenarien

## 4.6 Pflichtenheft

Im Pflichtenheft werden alle Punkte festgehalten, welche umgesetzt werden. Zusätzlich wird beschrieben, wie diese Punkte realisiert werden. Ein Auszug aus dem Pflichtenheft lässt sich aus der unterstehenden Tabelle entnehmen:

|  |  |
| --- | --- |
| **Funktionalität** | **Umsetzung** |
| Hinzufügen und Entfernen von Einträgen | Die Daten sollen über die Kommandozeile erfasst werden und in einer SQLite Datenbank abgespeichert. Einträge können gelöscht werden. |
| Zugriff auf alte Berichte | In der Kommandozeile können frühere Berichte aus der Datenbank aufgerufen werden. Hinzufügen, Löschen und Erstellen ist wiederholt möglich. |
| Anzeige aller Einträge für den aktuellen Bericht | Die Kommandozeile hat eine Unterfunktion zur Anzeige von Einträgen für die gewählte Woche. |
| Erstellung von Berichten in HTML/PDF oder Text | Die Kommandozeile hat eine Unterfunktion für die Erstellung eines Berichtes. Hierfür wird eine Vorlage verwendet, welche in das gewählte Format überführt wird. |
| Konfiguration von Nutzerdaten | Die Nutzerdaten sind konfigurierbar. Diese werden in der Datenbank abgespeichert. |

Tabelle 4 Pflichtenheft

# 5. Implementierung

Reportrick wird in der Programmiersprache Python umgesetzt, da diese frei und für alle gängigen Betriebssysteme verfügbar ist. Python ist universell, üblicherweise interpretiert und eine höhere Programmiersprache, die alle Werkzeuge mit sich bringt, um dieses Projekt erfolgreich durchzuführen. Zusätzlich ist Python sehr reich an Modulen die nach Bedarf importiert werden können. Eins der benötigten Werkzeuge ist das Modul SQLite, welches die Implementierung einer dateibasierten Datenbank ermöglicht. Durch die geringe Größe kann Reportrick dadurch auf eine ausgewachsene Datenbank zugreifen und deshalb lässt sich das Datenbanksystem vollständig in die Anwendung integrieren. Das Python-Modul Jinja2 welche ermöglicht aus einer, eigenständig erstellten, HTML Vorlage einen Bericht zu erzeugen. Dazu werden spezielle Platzhalter in der HTML-Vorlage genutzt, welche bei der Erstellung durch einen Wert ersetzt werden, um ein finales Dokument zu erzeugen. Durch Hinzufügen von einem CSS Datei ist es möglich das Layout und Design jederzeit anzupassen.

## 5.1 Kommandozeilen Modul

Das Kommandozeilen-Modul ist gleichzeitig unser Hauptmodul, mit welchem der Anwender über die Kommandozeile kommuniziert. Dieses Modul importiert argprse das Unterbefehle Optionen und Argumente behandelt. Die Main-Methode von Reportrick übergibt ein Objekt mit allen Attributen an die parsecli-Methode. Hier wird überprüft, ob das Objekt eine Unterbefehl enthält oder nicht. Abhängig davon verläuft der weitere Programmablauf unterschiedlich. Dies ist in den Abschnitten 5.1.1 Kommandozeilen Interaktion mit Menü, ohne Unterbefehle und 5.1.2 Kommandozeilen Interaktion ohne Menü mit Unterbefehl festgehalten. Um die Fehlersuche zu vereinfachen und den Ablauf des Programms nachzuvollziehen, wird das logging Modul eingesetzt. Das Modul unterstützt mehrere Ausgaben des Loggings wie beispielsweise Debug, Info, Warning, Error und Critical. Dadurch können bestimmte Ausgaben durch den Anwender ein- oder ausgeblendet werden. Für die Fehlersuche ist Debug sinnvoll, während für den normalen Ablauf Warning oder Error sinnvoll wären.

### 5.1.1 Kommandozeilen Interaktion mit Menü ohne Unterbefehle

Falls kein gültiger Wert übergeben worden ist, dann wird auf der Konsole ein Menü ausgegeben. Der Nutzer wird begrüßt und das aktuelle Datum ausgeben. Durch das Nutzten der datetime und date-Module können wir sehr einfach das Datum und die aktuelle Kalenderwoche ausgeben und verwenden. Der Anwender wird aufgefordert eine Eingabe in Form einer Zahl zu machen, um auf Funktionen wie in der Entwurfsphase 4.1.1 Interaktion ohne Kommandozeilenargumente auszuführen.

Ein Auszug mit der Logik zeigt, wie der Anwender auf die einzelnen Unterfunktionen zugreift:

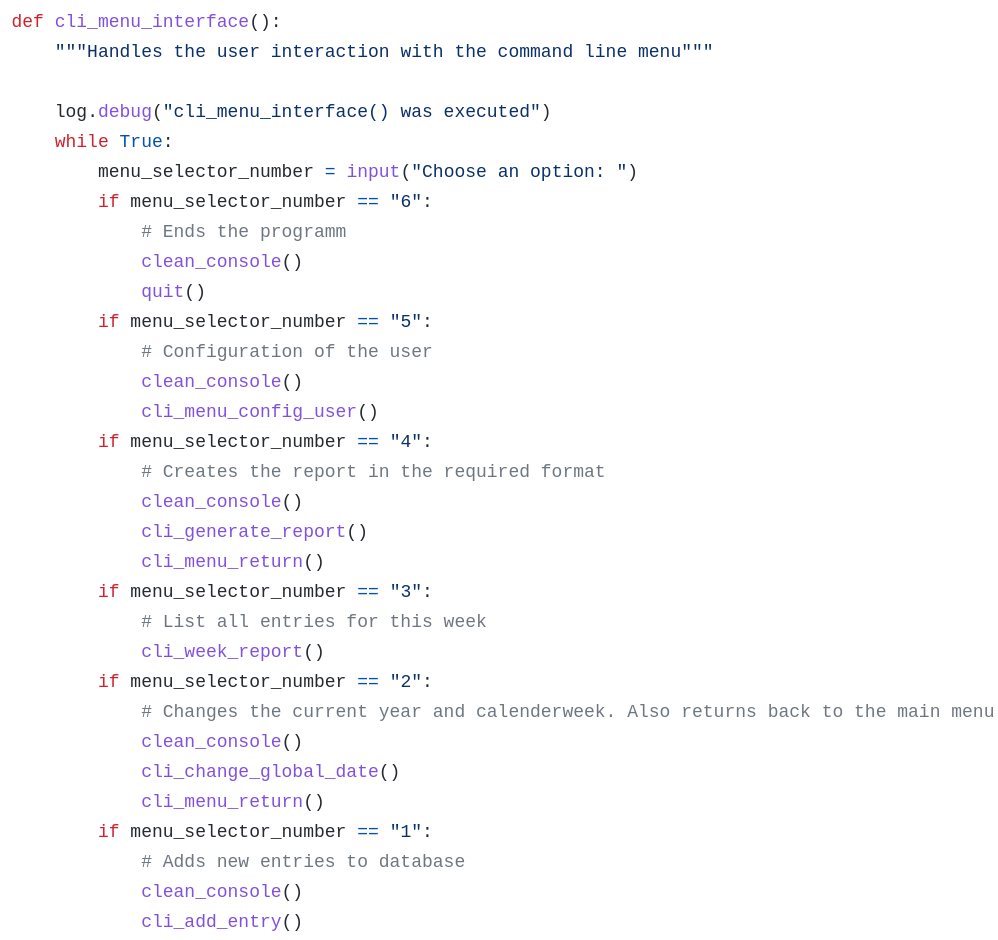


Abbildung 4 Implementierung des Menü Interfaces

### 5.1.2 Kommandozeilen-Interaktion ohne Menü, mit Unterbefehl

Tools teilen ihre Funktionalität auf unterschiedliche Unterbefehle auf. Im Falle von Reportrick gibt es Befehle für das Anlegen, Anzeigen und Ausgeben. Für diese Funktionalität werden sog. "Subparser" verwendet. Subparser analysieren die Kommandozeile nach den Befehlen, für die diese "zuständig" sind. Die Implementierung einer „help“ Funktion, zeigt alle möglichen Befehle und Unterbefehle, mit Hinweisen wie diese zu nutzten sind. Dabei greifen wir auf das argparse Modul zurück. Die Implementierung findet in der parsecli() Methode statt. Befehle werden in einem ähnlichen Muster implementiert. Hier ein Codeauszug für den Befehl „ADD“ der einen Eintrag in der passenden Kategorie hinzufügt.

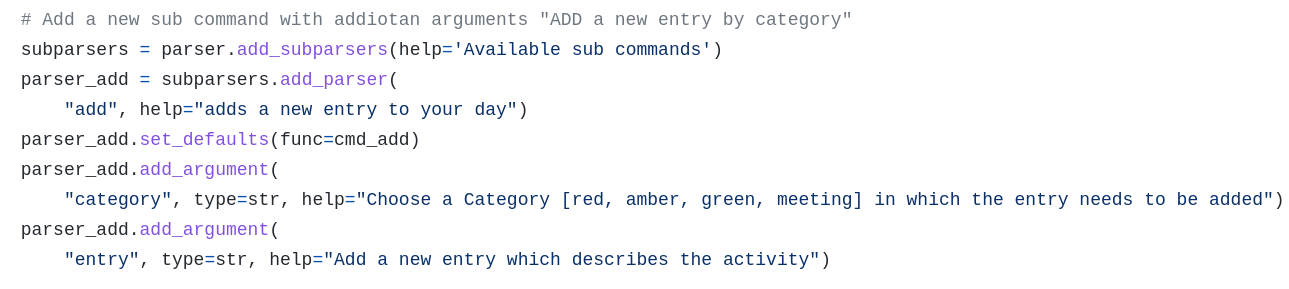


Abbildung 5 Implementierung eines Unterbefehls mit zwei Argumenten

Beispielaufruf: reportrick add red Code Review

Dieser Befehl würde für die aktuelle Kalenderwoche/Jahr den Eintrag „Code Review“ in der Kategorie „red“ hinzufügen. Wie in 4.1.2 Interaktion mit Kommandozeilenargumenten beschrieben, ist diese Funktionalität essenziell für den Anwender.

## 5.2 Datenbank Modul

Die Nutzung der Datenbank wird in einer Klasse “Database“ umhüllt. Bei dem ersten Start von Reportrick wird eine Instanz der Klasse Datenbank erzeugt und der Konstruktor erstellt mit dem Aufruf einer privaten Methode eine leere Datenbank. SQLite-Befehle werden als Zeichenketten in Variablen abgespeichert.   
Siehe Abbildung 6 Implementierung der privaten Methode für die Tabellenerstellung

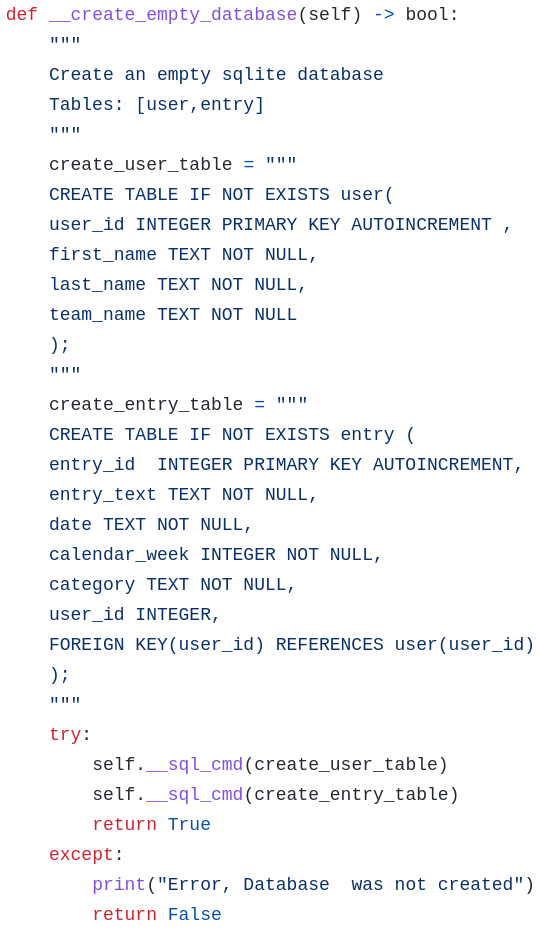


Abbildung 6 Implementierung der privaten Methode für die Tabellenerstellung

## 5.3 Berichts-Modul

Das Berichts-Modul ist explizit für die Erstellung der Berichte verantwortlich. Das Hauptmodul ruft Methoden aus dem Berichts-Modul auf und übergibt ihnen alle Einträge aus der Datenbank in Form einer Python Liste(=Array).

Das Jinja2 Modul nutzt ein Dictionary(=“Schlüssel-Werte“-Dateistruktur), um Werte an bestimmten Stellen in die Struktur der Vorlage einzubinden.  
Beispiel für ein Dictionary:



Abbildung 7 Implementierung eines Dictionary für das Berichts-Modul

Das Modul beinhaltet für jedes Format eine eigene Methode für die Generierung. Dabei bezieht sich das Modul immer auf eine Vorlage, die in einem „template“ Ordner innerhalb des Projektes abgespeichert worden ist.

## 5.4 Vorlage für das Bericht-Modul

Um Berichte zu erstellen wird eine HTML Vorlage für das Berichts-Modul verwendet. Eine HTML-Struktur mit Platzhaltern wird angelegt. Beim Ausführen greift das Jinja2-Modul auf die HTML-Vorlage zu und befüllt diese mit Werten aus der Datenbank. Auszug aus der jinja2 HTML-Vorlage für die dynamische Befüllung aller Einträge in der Kategorie grün.



Abbildung 8 HTML Vorlage in Jinja2 mit Platzhaltern

# 6. Projektkontrolle

## 6.1 Zeitlicher Soll-Ist-Vergleich

Im Soll-Ist-Vergleich wird meine Zeitplanung mit der tatsächlichen

Zeit verglichen. In der nachfolgenden Grafik ist ein Soll-Ist-Vergleich grafisch dargestellt:

Abbildung 9 Soll-IST-Vergleich

Wie es sich aus der Grafik entnehmen lässt, habe ich in der Entwurfsphase 1,5 Stunden länger gebraucht als es in der Planung angesetzt war. Das liegt daran, dass ich den Entwurf für den Bericht mehrfach entwerfen musste, da nach einer Rücksprache mit dem Auftraggeber einige Verbesserungen vorgeschlagen worden sind. Dafür konnte ich in der Durchführung durch ein detailliertes Konzept der Umsetzung 1,5 Stunden wieder einsparen. Dadurch gibt es Abweichungen von der ursprünglichen Zeitplanung, welche aber die gesamte vorgegebene Projektzeit von 70 Stunden nicht beeinflussen.

Die detaillierte Ansicht ist unter Table 8 Tatsächlicher Zeitaufwand zu finden.

## 6.2 Ziel-Vergleich

Am Ende der Umsetzung sollen die Anforderungen mit dem Ergebnis verglichen werden. Als Hilfsmittel dazu dient das Pflichtenheft aus dem Abschnitt 4.6 Pflichtenheft. Hierbei werden alle festgelegten Anforderungen durchlaufen und mit dem tatsächlichen Ergebnis verglichen.

In der nachfolgenden Tabelle ist die Funktionalität zusammen mit dem Ergebnis aufgelistet:

Tabelle 5 Ziel-Vergleich

|  |  |
| --- | --- |
| **Funktionalität** | **Ergebnis** |
| Hinzufügen und Entfernen von Einträgen | V |
| Zugriff auf alte Berichte | V |
| Anzeige aller Einträge für den aktuellen Bericht | V |
| Erstellung von Berichten in HTML, PDF oder Text | V |

## 6.3 Kostenvergleich

In diesem Abschnitt werden die geplanten Kosten den tatsächlichen Kosten gegenübergestellt. Der Kostenvergleich lässt sich aus der folgenden Tabelle entnehmen:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vorgang | Mitarbeiter | Stundensatz | Zeit | Kosten |
| Entwicklungskosten | 1x Auszubildender | 9.34 € | 70 h | 653.80 € |
| Code Review | 1x Mitarbeiter | 30.00 € | 2 h | 60.00 € |
| **Gesamtkosten** | | | | 713.80 € |

Tabelle 6 Kostenvergleich

Trotz kleiner Abweichungen in einzelnen Projektphasen wurden die geplanten Kosten eingehalten.

## 6.4 Testdurchführung

In der folgenden Phase wird für korrekte Funktionalität und Qualität des Quellcodes gesorgt. Dafür wird die Testdurchführung in zwei Schritte aufgeteilt, welche unten aufgeführt sind.

### 6.4.1 Code Review als White-Box-Test

Der White-Box-Test beinhaltete das Testen des Quellcodes. Dazu wurde zunächst ein Code Review von einem erfahrenen Entwickler durchgeführt, um die Zuverlässigkeit, Wartbarkeit und Erweiterbarkeit des Codes sicherzustellen. Das Ergebnis der Code Review ergab, dass die Namen nicht sprechend genug sind. Außerdem wurden Hinweise mitgegeben, dass ungenutzte Python Bibliotheken entfernt werden können. Des Weiteren wurden versucht, die Anzahl der Abhängigkeiten zu reduzieren. Hierbei musste der Code nochmal umgeschrieben werden. Die Namen wurden sprechender beschrieben und unnötige Module wurden entfernt.

Ein wiederholter Code Review ergab keinen weiteren Anpassungsbedarf.

### 6.4.2 Systemtest als Black-Box-Test

Die Anpassung wird anhand der Spezifikation getestet. Hier wird geprüft, ob die Anforderungen erfüllt werden. Dabei werden die in der Planungsphase festgelegten Testszenarien durchlaufen. Die Testergebnisse werden zusätzlich dokumentiert.

# 7. Projektabschluss

## 7.1 Übergabe an Kunden/Team

Nach einer kurzen Einführung wurde das neue Tool von Mitarbeitern getestet. Nachdem keine Fehler gefunden worden sind, wurde das Projekt firmenintern von den Mitarbeitern in ihre tägliche Arbeitsroutine aufgenommen.

## 7.2 Fazit

Mein Fazit habe ich in drei wichtigste Punkte aufgeteilt:

### 7.2.1 Projekt umgesetzt

Durch dieses Projekt wurde die Problemstellung gelöst und somit ist jetzt der wöchentliche Bericht der Mitarbeiter einheitlich und vollständig.

### 7.2.2 Zufriedener Auftraggeber

Der Auftraggeber ist sehr zufrieden, da sowohl alle Anforderungen als auch das Projektziel erreicht wurde.

### 7.2.3 Gelerntes Wissen

Durch das Projekt konnte ich wertvolle Erfahrungen im Bereich Projektmanagement gewinnen. Bei der Analyse und Konzeption hatte ich einen tieferen Einblick in verschiedene Teams innerhalb der Firma. Zusätzlich konnte ich bei der Umsetzung mein fachliches Wissen als Programmierer erweitern.

## 7.3 Ausblick

Durch die Implementierung dieses Projektes entsteht ein solider Grundbaustein, der schon bereit zur Nutzung ist. Die gewählte Architektur ermöglicht Erweiterungen nach Bedarf. Wie in der Projekt Abgrenzung unter 2.5 Projektabgrenzung ist die automatisierte Weiterleitung des Berichtes durch E-Mail- möglich. Weitere möglichen Features wie Importieren eigener CSS Dateien für die Umgestaltung von Farben und Logos sind möglich. Eine weitere Möglichkeit dieses Projekt zu erweitern ist die Implementierung ein Betriebssystem-unabhängigen grafischen Benutzerinterfaces mithilfe von der Godot Spiele Engine. Somit kann eine Interaktion ohne Kommandozeile auf üblichen Betriebssystem wie Windows, MacOS, Linux und Android genutzt werden.

8. Glossar

A

argprse

Modul in der Standartbibleothek von Python. Hilft beim Parsen von Kommandozeilen um Optionen und Argumente zu trennen **9**

B

Black-Box-Test

Methode des Softwaretests. Hierbei werden Tests anhand der Spezifikation/Anforderung entwickelt. Dies bedeutet, dass Tests ohne Kenntnisse über die innere Funktionsweise/Implementierung des zu testenden Systems entwickelt werden. **9**, **14**

C

Code Review

Systematische Untersuchung von Quellcode mit dem Ziel, Fehler und Mängel im Code zu finden, um die Softwarequalität zu verbessern **4**, **6**, **11**, **14**

CSS

Cascading Style Sheets ist eine Sprache um das Design von HTML-Dokumenten zu bestimmen. **4**, **5**, **9**, **20**, **21**

D

DAPS

openSUSE DocBook Authoring and Publishing Suite ist eine Software die Dokumentation in verschiedenen Formaten erstellen kann. **3**

date

Teil des Python datetime Moduls 10

datetime

Das Python datetime Modul gibt die Möglichkeit, mit dem Datum zu rechnen, dieses als String auszugeben oder einen String in ein Datumsobjekt zu konvertieren **10**

Debug

Als Debuggen bezeichnet man in der Informatik den Vorgang, in einem Computerprogramm Fehler oder unerwartetes Verhalten zu diagnostizieren und zu beheben **9**, **10**

Draw.io

Werkzeug für Diagramme, Organigramme und Mindmaps und viel mehr. 5

E

ER Diagramm

Entity-Relationship-Modell zur Darstellung von Dingen, Gegenständen, Objekten und der Beziehungen/Zusammenhänge zwischen diesen **8**

G

Git

Freie Software zur verteilten Versionsverwaltung von Dateien. **5**

GitHub

Netzbasierter Dienst zur Versionsverwaltung für Software-Entwicklungsprojekte **3**, **4**, **5**

H

HTML

Hypertext Markup Language ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. **3**, **4**, **5**, **8**, **9**, **12**, **13**, **20**, **21**

J

Jinja2

Jinja ist eine Web-Template-Engine für die Programmiersprache Python **5**, **9**, **11**

L

Linux

Freie, unixähnliche Mehrbenutzer-Betriebssysteme, die auf dem Linux-Kernel und wesentlich auf GNU-Software basieren **3**, **15**

logging

Modul der Standardbibliothek von Python. Dient zum automatischen Protokollieren von Statusinformationen **9**

M

Mockup

Vorführmodell **5**, **7**, **18**, **19**

O

*Open Source*

Computersoftware, die unter einer freien Lizenz veröffentlicht wird, bei der der Urheberrechtsinhaber den Benutzern das Recht einräumt, die Software und ihren Quellcode an jedermann und für jeden Zweck zu verwenden, zu studieren, zu ändern und zu verteilen **3**

openSUSE

Eine Linux-Distribution mit Fokus auf ein stabiles und benutzerfreundliches Betriebssystem **3**, **4**

openSUSE Tumbleweed

Tumbleweed ist eine Version der openSUSE. Dank laufender Aktualisierung hat man Zugriff auf die aktuelle Software. **5**

P

PDF

Portable Document Format ist ein plattformunabhängiges Dateiformat **3**, **4**, **8**, **9**, **13**

Python

Python ist eine universelle, üblicherweise interpretierte, höhere Programmiersprache **5**, **9**, **11**, **14**

R

RPM

Package Manager ist ein freies Paketverwaltungs-System für Linux, ursprünglich entwickelt von dem Unternehmen Red Hat **4**

S

SQLite

Programmbibliothek, die ein relationales Datenbanksystem enthält. SQLite wird in Mobiltelefonen, in Browsern, Skype und vielen anderen Anwendungen eingesetzt. **5**, **9**, **11**

V

Visual Studio Code

Kostenloser Quelltext-Editor von Microsoft **5**

W

White-Box-Test

Methode des Software-Tests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems entwickelt werden. **14**

# Anhang A: Mockup Erstes Berichts Layout

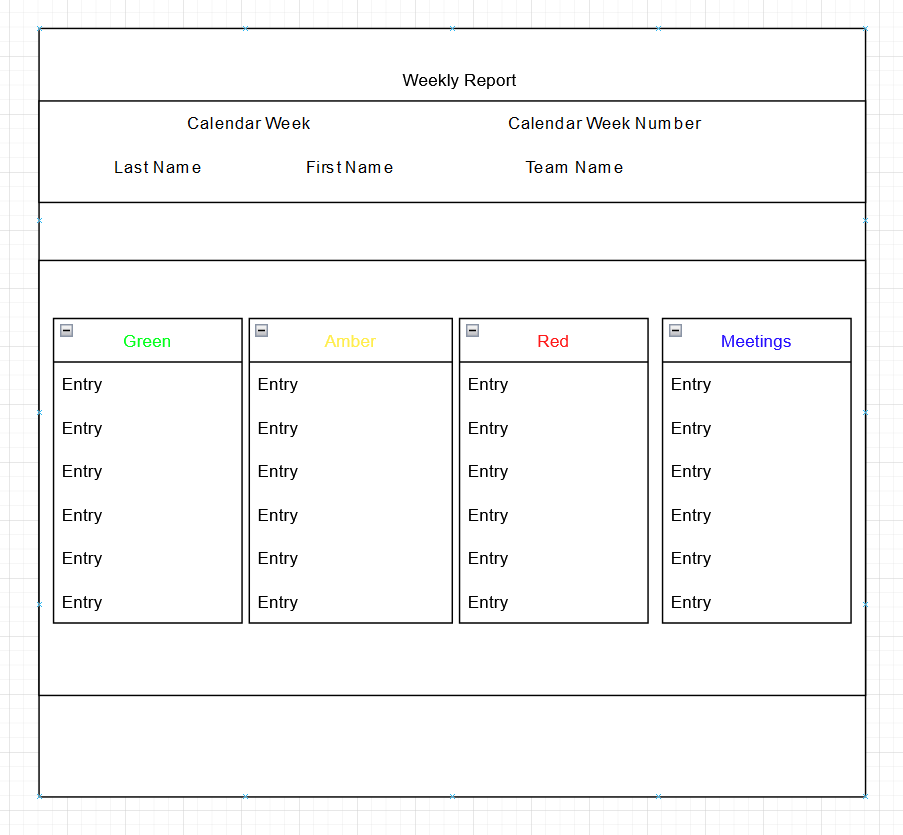


Abbildung 10 Vorläufiges Berichtsmodell

# Anhang B: Mockup Endgültiges Berichts Layout

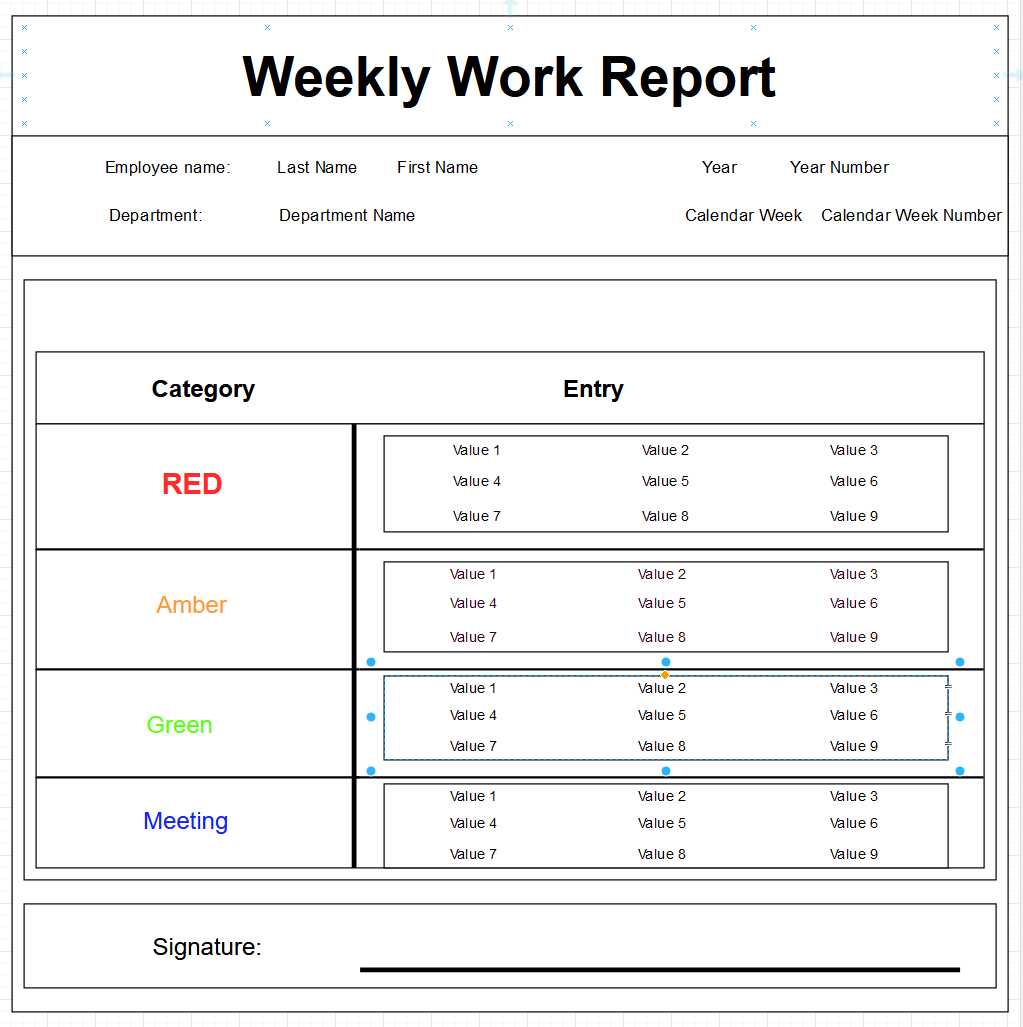


Abbildung 11 Endgültiges Berichtsmodell

# Anhang C: Detaillierte Zeitplanung

|  |  |  |
| --- | --- | --- |
| Projektphase | | Zeit in Stunden |
| 1 | Erstellung der Ist-Analyse | 2 |
| 2 | Erstellung des Soll-Konzepts | 2 |
| 3 | Zeit und Aufwandsplanung | 4 |
| 4 | Entwurf der Kommunikation zwischen Benutzer und Tool | 5 |
| 5 | Entwurf der Datenbank | 1 |
| 6 | Design des Berichtes | 2 |
| 7 | Aufteilung der Logik in mehrere Module | 1 |
| 8 | Planung der Testszenarien | 1 |
| 9 | Pflichtenheft | 2 |
| 10 | Erstellung der Datenbanklasse | 5 |
| 11 | Erstellung des Kommandozeilenprogramms | 20 |
| 12 | Entwicklung der Kommandozeilenstruktur | 5 |
| 13 | Entwicklung der Datenbankklasse | 5 |
| 14 | Entwicklung der HTML Struktur | 3 |
| 15 | Entwicklung der CSS Struktur | 3 |
| 16 | Implementierung des Jinja Scripts | 3.5 |
| 17 | Zeitlicher Soll-Ist-Vergleich | 1.5 |
| 18 | Zielerreichungsvergleich | 1 |
| 19 | Kostenvergleich | 1.5 |
| 20 | Testdurchführung | 5 |
| 21 | Übergabe an Team | 1 |
| 22 | Erstellung der Projektdokumentation | 10 |
|  |  |  |
|  |  |  |

Tabelle 3 Detaillierte Zeitplanung

# Anhang D: Datenbank Modell

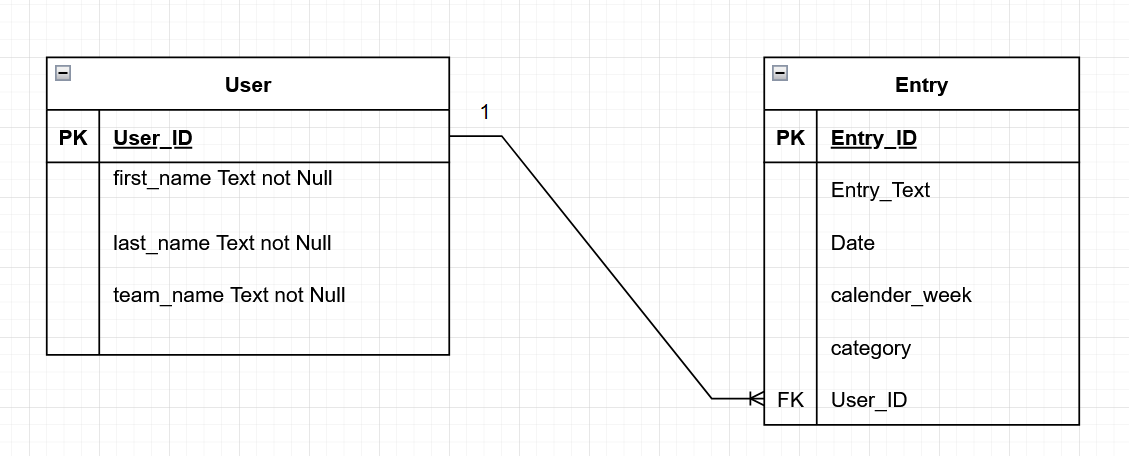


Abbildung 12 Datenbank Modell

# Anhang E: Klassendiagramm

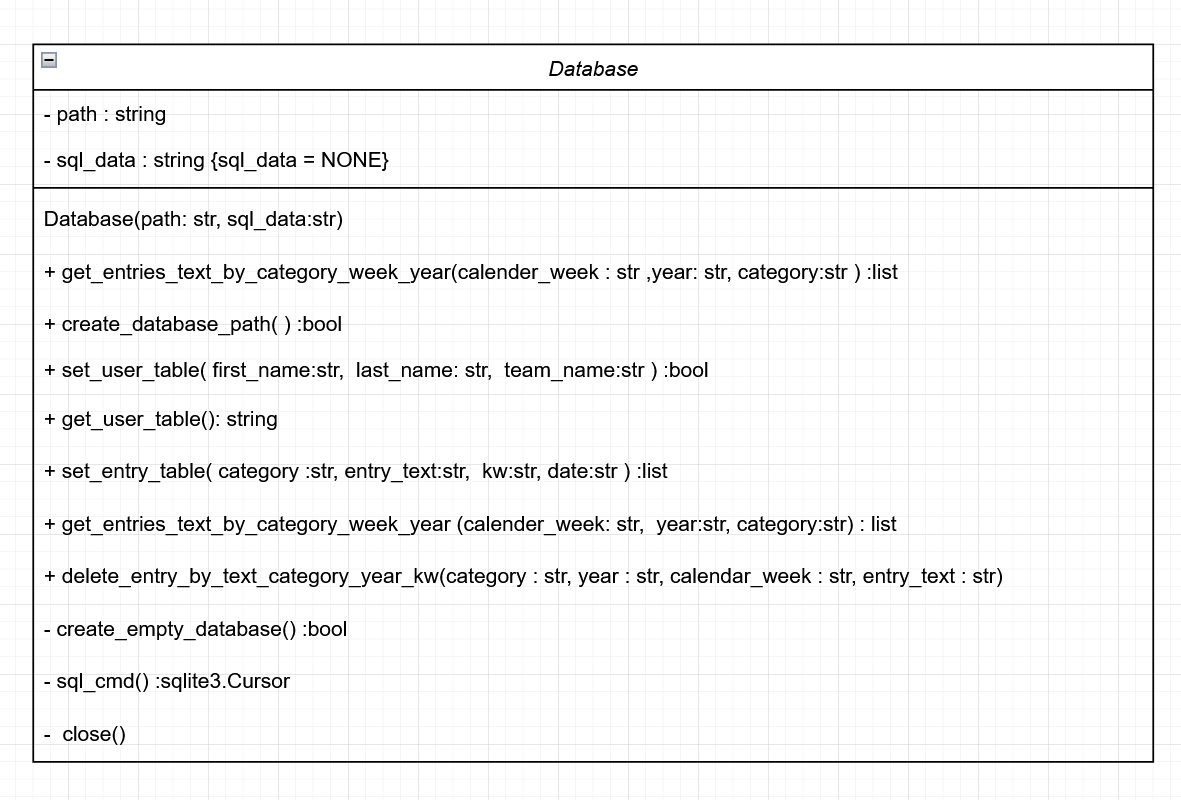


Abbildung 13 Database Klassendiagramm

# Anhang F: Zusätzliche Bearbeitungszeit

Detaillierter Zeitvergleich ist im Anhang zu finden

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | | Zeit in Stunden | Abweichung |
|  | Erstellung der Ist-Analyse | 2 | 0 |
|  | Erstellung des Soll-Konzepts | 2 | +1 |
|  | Zeit- und Aufwandsplanung | 1 | +2 |
|  | Planung der Testszenarien | 1 | -2 |
|  | Struktur und Ablauf des Kommandozeilentools | 4 | 0 |
|  | Entwurf des Layouts für die HTML Vorlage | 2 | 0 |
|  | Design der Datenbank | 2 | 0 |
|  | Pflichtenheft | 2 | 0 |
|  | Entwicklung der Kommandozeilenstruktur | 5 | 0 |
|  | Entwicklung der Datenbankklasse | 5 | 0 |
|  | Entwicklung der HTML Struktur | 3 |  |
|  | Entwicklung der CSS Struktur | 3 | 0 |
|  | Implementierung des Jinja Scripts | 3.5 | 0 |
|  | Zeitlicher Soll-Ist-Vergleich | 1.5 | 0 |
|  | Zielerreichungsvergleich | 1 | 0 |
|  | Kostenvergleich | 1.5 | 0 |
|  | Testdurchführung | 5 | 0 |
|  | Übergabe an Team | 1 |  |
|  | Erstellung der Projektdokumentation | 10 |  |
|  | Gesamt | 70 | 0 |
|  |  |  |  |

Tabelle 7 Tatsächlicher Zeitaufwand