|  |
| --- |
| Thies Michael, Masseo Winterstetter  5.6.2023 |

https://bbsgut-trier.de/wp-content/themes/BBS.GuT/img/logo1.png

|  |
| --- |
| Geräteverwaltung |
| Inventarisierungssoftware „PyVault“ |
| Abgabedatum: 16.06.2023 |

Inhaltsverzeichnis

[1 Projektbeschreibung 1](#_Toc137124613)

[1.1 Ausgangslage 1](#_Toc137124614)

[1.2 Projektziel 1](#_Toc137124615)

[1.3 Projektabgrenzung 1](#_Toc137124616)

[2 Zeitplan 2](#_Toc137124617)

[3 Aufbau 3](#_Toc137124618)

[3.1 Aufbau der Datenbank 3](#_Toc137124619)

[3.2 Aufbau des Programms 4](#_Toc137124620)

[4 Realisierung des Projekts 6](#_Toc137124621)

[4.1 Installation des Microsoft SQL Datenbank Server 6](#_Toc137124622)

[4.2 Erstellung der Datenbank 6](#_Toc137124623)

[4.3 Programmierung der Ausgabesoftware „PyVault“ 7](#_Toc137124624)

[4.3.1 Auswahl und Installation der Entwicklungsumgebung 7](#_Toc137124625)

[4.3.2 Verbindung der Datenbank und der Ausgabesoftware 7](#_Toc137124626)

[4.3.3 Erstellung des ersten Fensters 8](#_Toc137124627)

[4.3.4 Übergang zur Objektorientierten Programmierung 10](#_Toc137124628)

[4.4 Fazit 11](#_Toc137124629)

[5 Abbildungsverzeichnis 12](#_Toc137124630)

# Projektbeschreibung

## Ausgangslage

Die bisherige Erfassung der IT-Hardware im Unternehmen wird aktuell in verschiedenen Microsoft Excel Tabellen durchgeführt. Dies führt stellenweise zu redundanten Datensätzen und erschwert dadurch die Auswertung. Die Übersicht über den Bestand eines bestimmten Herstellers oder Modells, um beispielsweise bestimmte Geräte zu finden, die das Ende des Lifecycles erreicht haben, ist fehlerhaft. Da dies durch die verschiedenen Speicherorte und Tabellen zu falschen Ergebnissen führen kann und somit eine effiziente Arbeitsweise behindert.

Durch die hohe Zunahme an IT-Hardware, die zu Schulungszwecken genutzt wird, kommt das bisherige Schema basierend auf verschiedenen Tabellen mittlerweile an seine Grenzen.

## Projektziel

Aufgrund der bisherigen Unübersichtlichkeit und der schlechten Skalierbarkeit dieses Systems soll ein zentraler Speicherort der Daten in Form einer Datenbank etabliert werden, die durch eine Client-Software abgerufen werden kann. In dieser soll es auch möglich sein Daten zu filtern und die bisherigen Tabellen via CSV-Import in die Datenbank überführen zu können. Des Weiteren soll die Software mit einer grafischen Benutzeroberfläche bedienbar sein.

## Projektabgrenzung

Aufgrund des Umfangs des Projektes wird die Menge der Datensätze erst einmal auf die Hardware des Schulungsbereiches begrenzt. Weiterhin wird die Benutzeroberfläche der Ausgabemaske erst einmal auf wesentliche Funktionen begrenzt und sehr einfach gehalten werden.

Die Filterfunktion für die Datensätze wird zunächst auf die Optionen „Hersteller“ und „Inventarnummer“ begrenzt mit der Möglichkeit diese später zu erweitern.

# Zeitplan

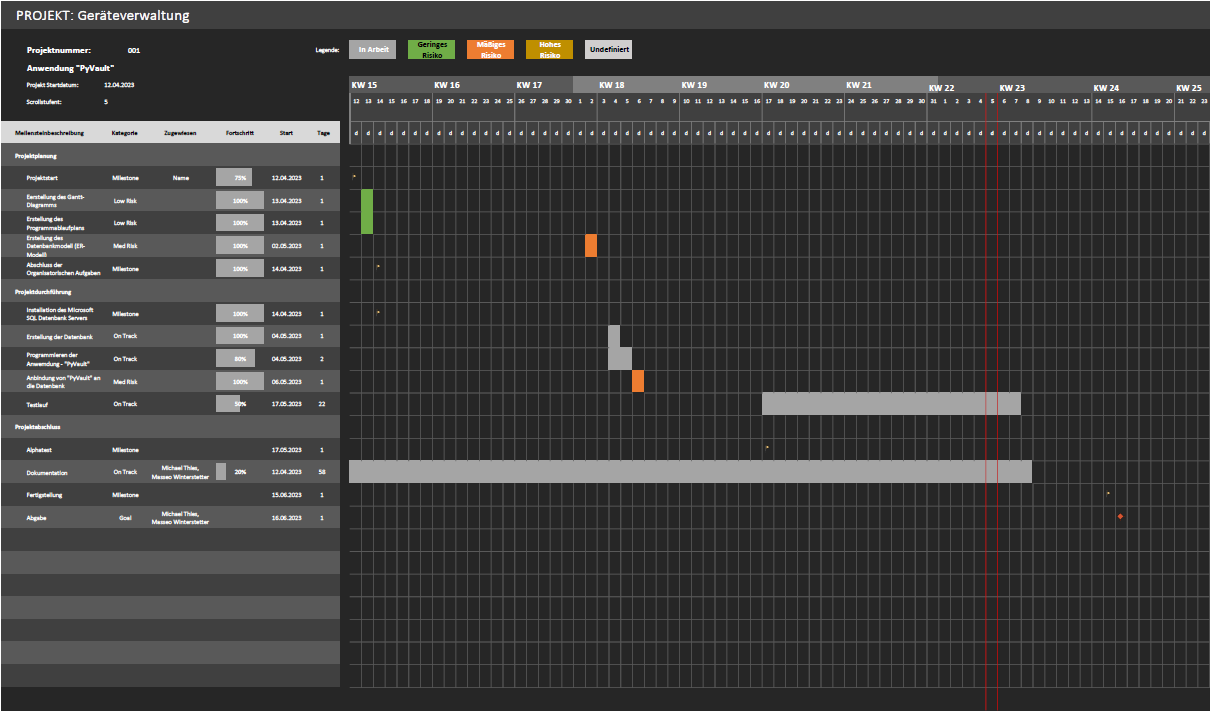


Abbildung Gantt-Diagramm

Bei der Erstellung des Zeitplans haben wir uns, wie in der Abbildung ersichtlich, für das Gantt-Diagramm entschieden, da dieses eine sehr gute Übersicht über den gesamten Projektzeitraum bietet. Des Weiteren bietet diese Art der Darstellung trotzdem alle benötigten Daten zur genauen Auswertung.

Wir haben unser Projekt in die Phasen **Projektplanung**, **Projektdurchführung** und **Projektabschluss** unterteilt.

In der **Projektplanung** wurden alle Organisatorischen Aufgaben untergebracht, die zur Durchführung des Projekts nötig waren. Wie etwa die Zeitplanung und Erstellung eines ER- und eines Relationalen-Datenbankmodells und dem Programmablaufplans (PAP) um erst einmal den Plan festzulegen wie das Programm und die Datenbank aussehen sollen und was benötigt wird.

Während der **Projektdurchführung** ging es dann um die tatsächliche Realisierung des Projekts, wie die Installation des Microsoft SQL Datenbank Servers und die Programmierung der Client-Software mit der grafischen Benutzeroberfläche (GUI) zur einfachen Bedienung des Programms sowie am Ende der Verbindung der Datenbank mit der Client-Software und dem kontinuierlichen Testen der einzelnen Programmabschnitte nach Fertigstellung. Deswegen ist die Testphase auch entsprechend lang in unserer Projektplanung eingeplant. Die fertiggestellten Programmteile werden nach Erstellung und bevor diese zum tatsächlichen Programm zusammengesetzt wurden getestet.

Zum Schluss wird das Projekt mit der Phase **Projektabschluss** beendet. Darin enthalten sind der Alphatest, um einen aktuellen Stand zu erhalten und zu sehen, ob dieser den aktuellen Anforderungen an das Programm entspricht. Unser Projekt endet aus zeittechnischen Gründen mit einem Alpha Test, da die komplette Fertigstellung des Programms samt aller benötigten Funktionen den zeitlichen Rahmen dieses Schulprojekt sprengen würde. Daher haben wir uns entschieden, das Projekt damit zu beenden, da das Projekt erst einmal nur im Unternehmen getestet werden soll und falls dies genutzt würde, wird das Programm dementsprechend angepasst, um es im Produktionsbetrieb nutzen zu können. Des Weiteren umfasst die Phase die Projektdokumentation und die Abgabe des Projekts.

# Aufbau

## Aufbau der Datenbank

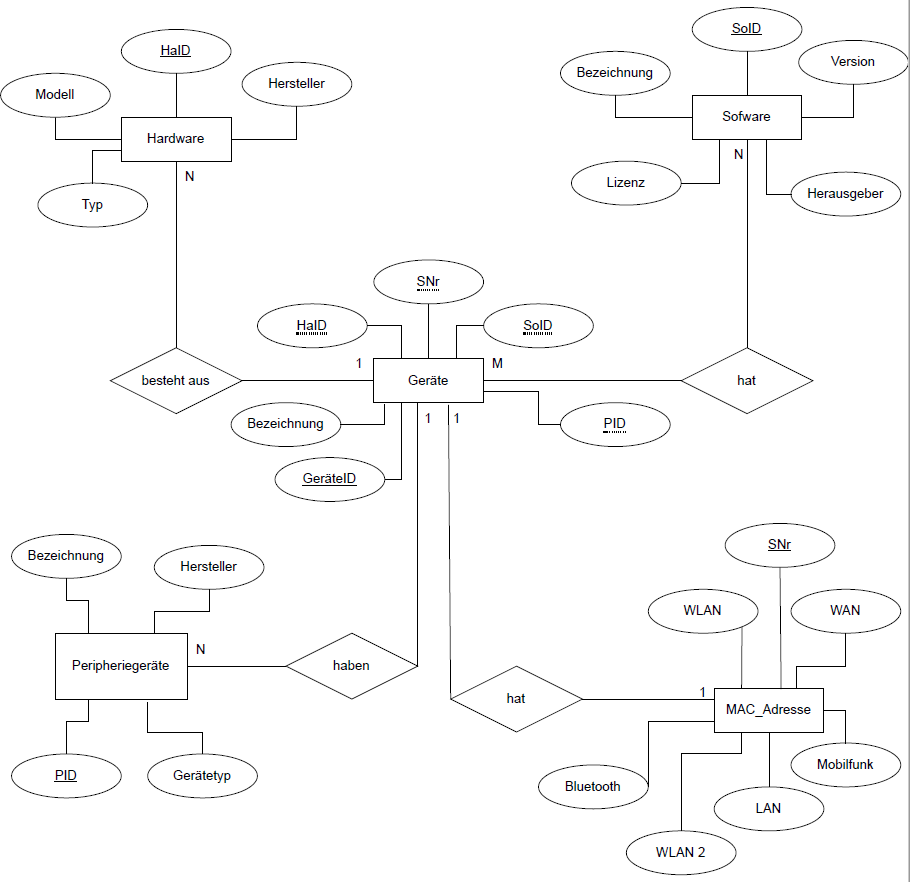


Abbildung ER-Modell

Dies ist unser ER-Modell zur Datenbank, als erstes haben wir uns Gedanken dazu gemacht welche Entitäten möchten wir denn Abbilden bzw. welche Entitäten brauchen wir um ein Gerät in der Datenbank mit allen nötigen Informationen anzeigen zu können für den Nutzten für den die Datenbank gedacht sein soll. Nämlich Geräte anhand einer MAC-Adresse zu identifizieren oder Geräte auf Accesspoints mit MAC-Filtern eintragen zu können zum Beispiel bei einem Raumwechsel im Schulungsbereich und dies Ohne an das Gerät selbst zu müssen. Oder wenn für Schulungen bestimmte Hard- oder Software benötigt wird anhand der Geräte-ID zu sehen welche Komponenten bei den Geräten bereits installiert sind und welche nachgeliefert werden müssen bzw. ob der Rechner oder Laptop überhaupt Hardware seitens dazu geeignet ist für diese Schulung genutzt zu werden oder ob ein neues Gerät dafür gebraucht wird. Daher haben wir uns für die Entitäten *Geräte*, *Peripheriegeräte*, *Hardware*, *Software* und *MAC-Adressen* entschieden.

Danach haben wir die Attribute für die einzelnen Entitäten bestimmt und geschaut das diese am Ende Eindeutig sind. Damit am Ende die Datensätze sich nicht doppeln und somit die Datenkonsisitenz gefährden. Daher ist uns natürlich aufgefallen als wir die Kardinalitäten bestimmt haben, dass bei dem Übergang in ein Relationales Datenbankmodell noch 2 Kreuztabellen benötigt werden.

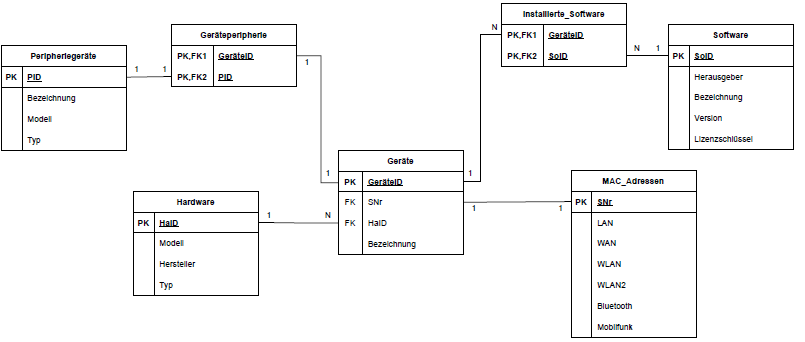


Abbildung Relationales Datenbankmodell

Wie in der Abbildung zu sehen ist wurde das Modell um die Kreuztabellen „*Geräteperipherie*“ und „*installierte\_Software*“ ergänzt. Da in einem relationalen Datenbankmodell keine M:N – Beziehungen vorhanden sein dürfen. Das Zentrum unserer Datenbank bildet die Tabelle Geräte, diese beinhaltet als Fremdschlüssel die Primärschlüssel´s der Tabellen *MAC-Adressen* und *Hardware*. Die Tabellen „*Peripheriegeräte*“ und „*Software*“ werden über die entsprechenden Kreuztabellen mit der Geräte-ID verknüpft. Dadurch kann jedes Peripheriegerät wie zum Beispiel eine Maus oder eine Tastatur einem bestimmten Gerät zugeordnet werden. Genauso kann man somit auch eindeutig sagen welche Software auf dem jeweiligen Gerät installiert ist um diese gegebenenfalls zu aktualisieren oder gegen eine andere zu ersetzen.

## Aufbau des Programms

Bei dem Entwurf des Programms begannen wir erst damit das eine Verbindung zum Datenbankserver und der Datenbank hergestellt werden soll. Dies sollte direkt bei Start des Programms ausgeführt werden. Danach soll um den Zugriff von unbefugten zu verhindern erst einmal eine Maske zur Anmeldung auftauchen. Diese haben wir im Programmablaufplan als Unterprogramm eingefügt.

In diesem sollte nach Eingabe von Benutzernamen und Passwort durch den Anwender diese Daten auf Übereinstimmung überprüft werden. Wenn diese nicht identisch sind werden eine Fehlermeldung und ein Verweis an den Administrator ausgegeben. Sofern die Eingaben mit den hinterlegten Daten übereinstimmen wird der Zugriff gewährt und das Hauptprogramm wird fortgesetzt.

Nach der erfolgreichen Anmeldung des Anwenders soll die GUI des Programms geladen werden damit die weiteren Funktionen vom Nutzer ausgewählt werden können.

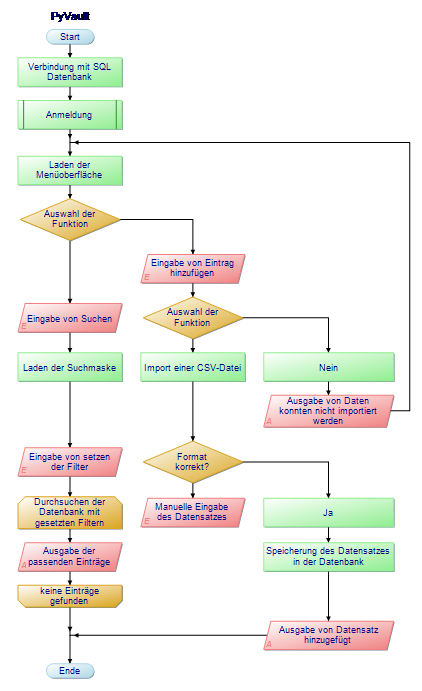


Abbildung Programmablaufplan "PyVault"

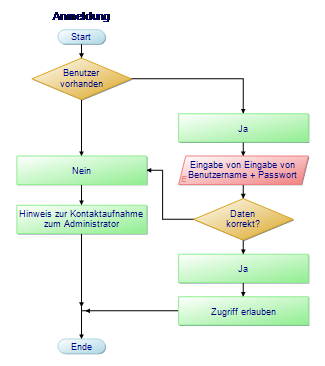


Abbildung Unterprogramm zur Anmeldung der Nutzer

Nach dem laden der GUI werden dem Nutzer die weiteren Funktionen angezeigt diese wurden von uns aufgrund des Umfangs des Projektes wie in der Projektabgrenzung beschrieben beschränkt auf das Suchen bzw. Anzeigen von Datensätzen aus der Datenbank, dem Filtern von Datensätzen und dem manuellen hinzufügen von Datensätzen und durch die Möglichkeit des Imports einer CSV-Datei.

Bei der Suchfunktion wird erstmal eine Maske geladen zum Eingeben der Suchparameter, wenn die Suchparameter eingegeben wurden wird im Programm eine Schleife erzeugt und das Programm durchsucht die Datenbank nach Datensätzen die den gesetzten Parametern entspricht. Sollte dies nicht erfolgreich sein kann der Nutzer neue Parameter setzten und die Suche erneut durchführen lassen bis Datensätze angezeigt werden.

Eine weitere Funktion ist das die Möglichkeit Filter zu setzten nach Auswahl der zur Verfügung stehenden Tabellen kann der Nutzer einen der vordefinierten Filter nutzen um die Anzeige auf bestimmte Werte zu begrenzen.

Die letzte Möglichkeit in dem Programm ist neue Datensätze hinzuzufügen da hat der Anwender die Wahl ob er dies händisch erledigen möchte oder via Import einer CVS-Datei. Nachdem der Anwender die Funktion Eintrag hinzufügen ausgewählt hat kann erwählen ob er die CSV-Datei importieren möchte, sollte das der Fall sein öffnet sich ein Fenster um die Datei auszuwählen und zu bestätigen. Dann wird geprüft ob die Datei das richtige Format (.CSV) entspricht ist das der Fall werden die Daten in die Datenbank übertragen und Die Software gibt eine Meldung aus das die Datei erfolgreich Importiert werden konnte. Ist die Datei nicht im CSV-Format vorhanden hat der Anwender noch die Möglichkeit Manuell über eine Maske Datensätze hinzuzufügen.

# Realisierung des Projekts

Für die Realisierung des Projekts haben wir uns für einen Microsoft SQL Datenbank Server entschieden. Sowie entsprechend der Projektspezifikationen die Software mit der Programmiersprache „Python“ verwirklicht. Da dies eine sehr einsteigerfreundliche Programmiersprache ist deren Syntax gut zu verstehen ist.

## Installation des Microsoft SQL Datenbank Server

Xx Platzhalter xX

## Erstellung der Datenbank

Begonnen haben wir mit dem SQL-Befehl „CREATE DATABASE PyVaultDB“, um die Datenbank auf dem Server zu erstellen. Danach haben wir eine Abfrage erstellt um die Tabellen samt den entsprechenden Primär- und Fremdschlüsseln.

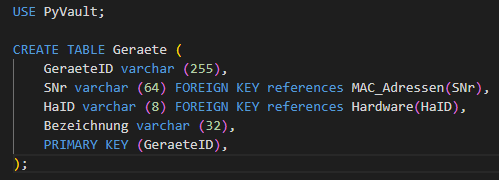


Abbildung Beispiel Erstellung einer Tabelle (Geraete)

Exemplarisch sehen Sie hier den Code für die Tabelle Geräte, da SQL den Buchstaben „ä“ nicht verwenden kann mussten wir die Tabelle „Geraete“ nennen. Der USE-Befehl am Anfang der Abfrage wählt die Datenbank aus in der die Abfrage ausgeführt werden soll. Da im Microsoft SQL Server Management Studio standardmäßig die Master Datenbank ausgewählt ist und dieser Befehl dies direkt ändert und der Befehl in der gewünschten Tabelle ausführt sofern diese natürlich besteht.

In den nächsten Zeilen wird durch CREATE die Tabelle erstellt direkt nach dem Befehl Steht der Name den die Tabelle tragen soll gefolgt von einer geöffneten Klammer.

Den Rest unserer Abfrage können Sie dem Anhang entnehmen dort werden alle Tabellen in einer Abfrage erstellt.

## Programmierung der Ausgabesoftware „PyVault“

Zur Programmierung haben wir uns für Tkinter entschieden, dies ist zwar nur eine recht einfache GUI im Vergleich zu anderen Frameworks wie PyQt, allerdings ist es schon fast das Standard-Framework bei der Erstellung einer GUI. Zudem ist es durch die Masse an Ressourcen seien es Bücher oder Referenzen sowie der Communitys die sich mit diesem Thema beschäftigen. Zudem ist es sehr einfach zu bedienen und somit perfekt für Einsteiger.

### Auswahl und Installation der Entwicklungsumgebung

Xx Platzhalter xX

### Verbindung der Datenbank und der Ausgabesoftware

Wir begannen bei der Programmierung mit der Verbindung zur Datenbank wie im PAP (Programmablaufplan) beschrieben um eine Verbindung mit einem MS SQL Datenbank Server mit Python herstellen zu können müssen wir zuerst noch die Bibliothek „pyodbc“ installieren. Dies haben wir über den pip-Installer getan indem wir in der Commando-Console den Befehl „pip install pyodbc“ eingaben, der Rest geschieht ganz automatisch.

Anschließen müssen wir die Bibliothek in unserer Python Datei einbinden, dies geschieht mit dem import-Befehl in Python.

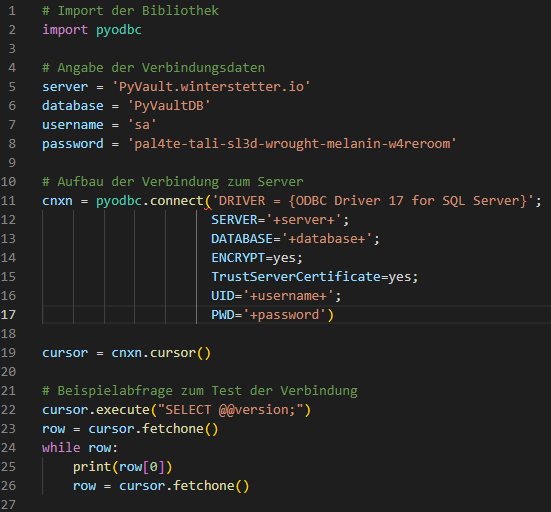


Abbildung Verbindungmit der Datenbank

Dann werden die Zugangsdaten für den Server und die Datenbank in Variablen gespeichert, darauf folgt dann das eigentliche herstellen der Verbindung zum MS SQL Datenbank Server. Als Abschluss folg noch eine Abfrage nach der Version des Datenbankserver um zu kontrollieren ob die Verbindung besteht.

### Erstellung des ersten Fensters

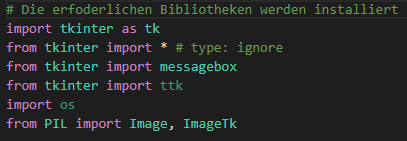


Abbildung Erster Import der nötigen Bibliotheken

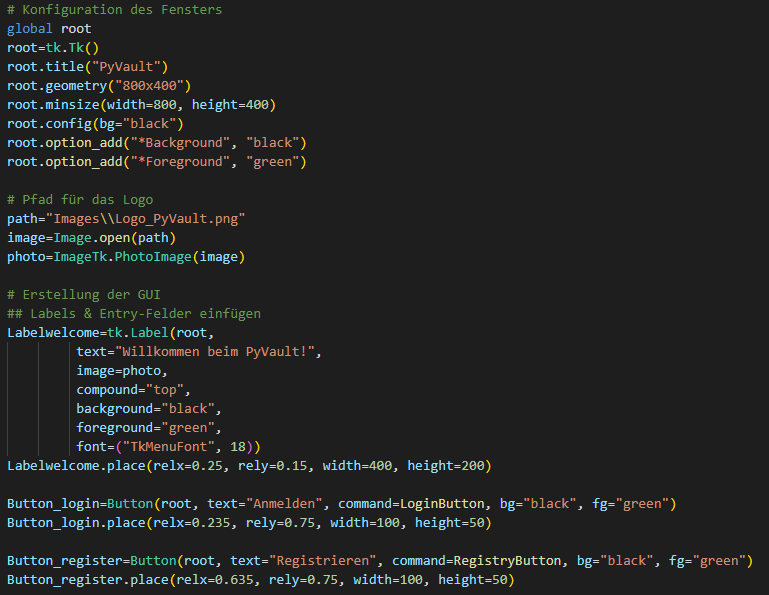


Abbildung Konfiguration der Startseite

Zuerst wurden Wie in der Abb. 8 gezeigt die notwendigen Bibliotheken importiert, der Befehl „# type: ignore“ dient dabei die Fehlermeldung von MS Visual Studio Code zu ignorieren die ausgegeben wird, wenn ganze Bibliotheken statt nur einzelner Elemente importiert werden.

In Abb. 9 wird die Startseite unseres Programms konfiguriert indem wir erstmal mit dem Befehl „root = tk.Tk()“ ein Fenster erstellen und mit den nachfolgenden Befehlen konfigurieren wie z.B. die Größe, den Hintergrund und die Schriftfarbe. Anschließend wird ein Bild eingefügt und in der Variablen „photo“ gespeichert. Das Foto wird dann im Label „LabelWelcome hinzugefügt und durch den Compound-Befehl wird gesagt das das Foto oberhalb des Textes angezeigt werden soll. Am Ende wird das ganze mit dem Place-Manager dem Fenster hinzugefügt und seine exakte Position bestimmt. Nach dem gleichen Vorgehen werden auch die beiden Buttons erstellt allerdings wird hier noch der Befehl „Command“ eingesetzt der auf die Funktion verweist die ausgeführt werden soll wenn der Button gedrückt wird.

Im weiteren Verlauf haben wir dem Fenster noch eine Menüleiste mit einem Dropdown-Menü hinzugefügt wie in Abb. 10 zu sehen ist.

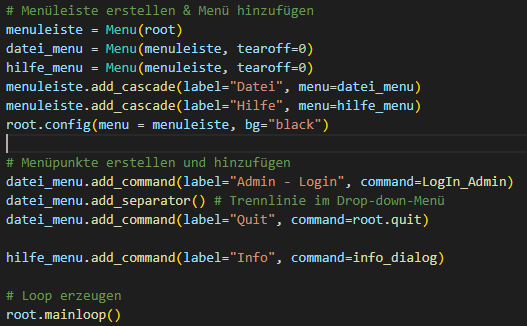


Abbildung Erstellung und Einfügen der Menüleiste

Als erstes wurde die Menüleiste erstellt und dem Fenster root welches unsere Startseite darstellt hinzugefügt. Danach wurden die Beiden Menüpunkte „Datei“ und „Hilfe“ hinzugefügt und die Menüleiste so konfiguriert das sie im Hintergrund schwarz sein soll und die Menüs als Dropdown-Menüs angezeigt werden sollen. Doch leider ist die Funktion den Hintergrund der Menüleiste in Tkinter zu ändern nicht möglich da die Farbe der Menüleiste und des Titels immer vom jeweiligen Betriebssystem des Rechners übernommen wird. Nachdem das Menü erstellt wurde werden diese durch die einzelnen Menüpunkte ergänzt. Durch den Befehl „add command“ werden den einzelnen Menüpunkten genau wie bei den Buttons die Funktionen zugewiesen die ausgeführt werden sollen wenn der Menüpunkt ausgewählt wird. Durch „add.seperator“ wird eine Trennlinie zwischen den Menüpunkten eingefügt.

Am Ende der Abbildung und des Programms wird dem Befehl „root.mainloop“ die Mainloop des Programms gestartet und das Fenster anzeigen zu können.

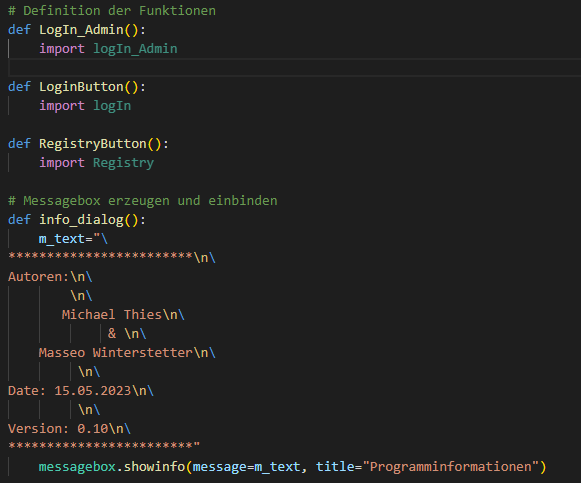


Abbildung Erstellen der Funktionen

Bevor jedoch das Programm mit dem Mainloop-Befehl abgeschlossen wird werden noch die Funktionen definiert. Bei der Funktion „LoginButton“ sieht man nur einen import-Befehl da dort auf eine neue Datei verwiesen wird die geladen wird sobald auf den Button „Login“ geklickt wird da dadurch diese Funktion ausgeführt wird. In der Funktion „info\_dialog“ wird eine Messagebox erzeugt die aufgerufen wird sobald in der Menüleiste auf Info geklickt wird. In der der Text der in der Variablen „m\_text“ gespeichert wird. Der Zeilenumbruch in der Messagebox wird durch \n\ erzeugt. Am Ende der Funktion wird die Messagebox erzeugt mit einem Titel versehen und der Text aus der Variablen übergeben.

### Übergang zur Objektorientierten Programmierung

Nach dem erfolgreichen Erstellen des ersten Fensters wurden die Dateien für die Fenster „Connection.py“, „logIn.py“, „Registry.py“, „Main.py“, „logIn\_Admin.py“ und „Adminbereich.py“ erstellt. Dabei ergaben sich allerdings Probleme. Wenn aus einer Datei auf eine andere im selben Arbeitsverzeichnis zugegriffen werden sollte hat dies nur zeitweise bzw. fehlerhaft funktioniert. Zudem konnte die Startseite nicht geschlossen werden, wenn ein anderes Fenster geöffnet wurde, da dadurch die Mainloop des gesamten Programms beendet wurde und alle Fenster sich schlossen. Dies passiert deswegen, weil die Programme ja in die Schleife der Startseite importiert werden und dadurch darin ausgeführt werden. Wird diese beendet werden auch alle darin befindlichen Unterprogramme beendet.

Die Lösung dieses Problems war die Umwandlung des Programmcodes in die Objektorientierte Programmierung (OOP), dadurch werden die einzelnen Fenster statt in eigenen Dateien in Klassen in der selben Datei gespeichert dadurch ist das Problem mit dem Import behoben. Zudem können auch Fenster geschlossen und andere geöffnet werden ohne die Mainloop zu beenden.



Abbildung Teil 1 Programmcode der Startseite (OOP)



Abbildung Teil 2 Programmcode der Startseite (OOP)

In der OOP wird das was vorher eine eigene Datei war nun in einer Klasse gespeichert durch den Zusatz „self.“ vor einen Befehl wird dieser ein Objekt dieser Klasse z.B. muss vor dem Label „LabelWelcome“ nun der Zusatz „self.“ also lautet dies nun „self.LabelWelcome“ damit dies auch in der GUI später angezeigt werden kann.

## Fazit

# Abbildungsverzeichnis

[Abbildung 1 Gantt-Diagramm 2](#_Toc137124591)

[Abbildung 2 ER-Modell 3](#_Toc137124592)

[Abbildung 3 Relationales Datenbankmodell 4](#_Toc137124593)

[Abbildung 4 Programmablaufplan "PyVault" 5](#_Toc137124594)

[Abbildung 5 Unterprogramm zur Anmeldung der Nutzer 5](#_Toc137124595)

[Abbildung 6 Beispiel Erstellung einer Tabelle (Geraete) 7](#_Toc137124596)

[Abbildung 7 Verbindungmit der Datenbank 8](#_Toc137124597)

[Abbildung 8 Erster Import der nötigen Bibliotheken 8](#_Toc137124598)

[Abbildung 9 Konfiguration der Startseite 9](#_Toc137124599)

[Abbildung 10 Erstellung und Einfügen der Menüleiste 9](#_Toc137124600)

[Abbildung 11 Erstellen der Funktionen 10](#_Toc137124601)

[Abbildung 12 Teil 1 Programmcode der Startseite (OOP) 11](#_Toc137124602)

[Abbildung 13 Teil 2 Programmcode der Startseite (OOP) 11](#_Toc137124603)