

## **EPR 08 Team Virtuelles-Vereinskassen-System**

In unserem Programm geht es um ein Verwaltungssystem für Vereine, was mit Thinkter (Bibliothek für GUI) erstellt wurde. Durch diese Bibliothek können Administratoren neue Vereine erstellen und Dateien speichern, wo alle Vereine aufgelistet sind. Wie man bereits am Anfang des Programmodes erkennt, werden zunächst weitere Bibliotheken erstellt. Messagebox zeigt Nachrichten, wie Erfolg und Misserfolg an. Die Bibliothek „font“ kann die Schriftart ändern. Und „csv“ nutzt man, um Daten wie die Clubdaten in der Datei zu speichern.

Die erste Klasse ist **Administration** verwaltet zum einen das Hauptfenster und sorgt dafür, dass ein Admin die Möglichkeit hat verschiedene Funktionen zu nutzen. Dabei speichert **self.admin** wer der Admin ist. **Self.account** speichert eine Liste, wo die Vereine aufgelistet sind und **self.main\_window** erstellt ein Hauptfenster dieses Programms. Mit der Funktion **admin\_option(self)** wird ein Fenster mit dem Namen „Administrator“ geöffnet. Danach erscheint ein Text, mit der Frage, was der Admin tun möchte. Darauffolgend gibt es dann zwei Buttons. Für „Create Club“ kann man einen neuen Verein erstellen und durch „See current status club in CSV file“ werden gespeicherte Vereinsdaten angezeigt. Danach ermöglicht die Funktion **create\_club(self)** einen neuen Verein zu erstellen. Hierfür wird ein neues Fenster geöffnet, mit dem Titel “Create Club”. In einem Eingabefeld kann man den Namen des Vereins eingeben und der Button „create“ speichert den Namen ab. Zunächst folgt die Funktion **submit()**, welche den neuen Verein speichert. Hier wird der Name, der eingegeben wird, überprüft. Das bedeutet, wenn da kein Name steht oder der Name bereits existiert erscheint eine Fehlermeldung. Aber wenn gerade ein neuer Name eingetragen wurde, dann wird der Club gespeichert. Die Funktion **save\_csv(self,club\_name)** speichert dann die ganzen Vereine in einer Datei namens “club.csv”. Erwähnenswert hierbei ist, dass eine Warnung ausgegeben wird, wenn ein Fehler auftritt.

Durch die Funktion **back\_to\_main(self)**, wird man in das Hauptmenü zurückgebracht. Diese Funktion hat die Aufgabe das Fenster zu schließen, um ein neues Hauptmenü zu öffnen. Nun kommt eine weitere Klasse „Cashier“ ins Spiel. Diese Klasse speichert wichtige Informationen über den Vereinsnamen und den Kassierer und verwaltet die Ein- und Auszahlungen. Denn durch die Methode “transaction\_window(self, trans)” kann beispielsweise ein Kassierer eine neue Transaktion eingeben. Dabei kann ein Kassierer zwischen Ein- und Auszahlung entscheiden. So kann man hier den Betrag und die Beschreibung eingeben. Sobald die Daten stimmen, werden die Transaktionen in einer Datei gespeichert. Aber wenn was nicht korrekt ist, dann erscheint eine Fehlermeldung. Eine weitere Methode “show\_transaction\_history(self, parent)” ist dafür da, um die gespeicherten Transaktionen anzuzeigen. Dafür wird eine Datei mit

Transaktionen geöffnet, wo diese Liste mit den gespeicherten Einträgen gelesen werden und als Liste nochmal angezeigt werden. Durch eine weitere Klasse **FinanceOfficer** kann man die Transaktionshistorie vom Verein finden. Darauf folgen wichtige Funktionen wie **show\_transaction\_history**, welche die vorherigen Transaktionen aus der CSV-Datei einliest und schließlich anzeigt. Danach schreibt man wieder die Funktion **back\_to\_main()**, damit man als Nutzer zurück in das Hauptmenü kommt.

Die Klasse Apps erstellt durch die erste Methode **\_init\_(self)** das Hauptfenster, wobei die Größe auf 800x400 Pixel gesetzt wird. Der Titel „Vereinskassen System“ erscheint.

Die Funktion **show\_main\_menu(self)** zeigt das Hauptmenü und eine Begrüßung an. Dabei hast du die Möglichkeit auszuwählen, ob du als Kassierer, Finanzoffizier oder Administrator arbeiten möchtest. Hierfür gibt es Buttons für alle drei Tätigkeiten. Bei **entry(self, role)** wird ein neues Fenster geöffnet, wo man sein Passwort eingeben kann. Wichtig ist hier noch, dass **tk.Toplevel()** ein neues Fenster erstellt.

Deswegen gib es auch die **login(self, role)** Funktion, die das Passwort überprüft. Und wenn das Passwort richtig ist, dann öffnet sich ein anderes Fenster für z.B. den Kassierer. Ist das Passwort falsch, dann erscheint durch das Element **messagebox.showerror** eine Fehlermeldung. Außerdem organisiert **pack()** die Positionierung der Buttons. Letztendlich kann durch **if \_\_name\_\_ == “\_\_main\_\_”** das Programm richtig gestartet werden.