Programmierung eines Vocitrainers

Untertitel

Inhaltsverzeichnis

Einleitung

Man kann dem Umstand nicht aus dem Weg gehen, Vokabeln lernen gehört zum Alltag eines jeden Schülers. Niemand macht es gerne, aber man muss durch. Da stellt sich die Frage, wie man mit möglichst wenig Zeitaufwand seinen Wortschatz effizient und nachhaltig erweitern kann.

Früher haben wir vermutlich alle noch mit analogen Karten aus Papier und einem Karteikasten gelernt und dann Quizlet entdeckt. Quizlet macht vieles leichter. Der Vorteil dieser Webseite ist es, dass man sich bereits von anderen Benutzern erstellte Lernsets von Wörtern herunterladen kann und dabei fast jedes Lehrmittel auf der Seite abgedeckt ist. Auch sonst muss nur eine Person das Set erstellen und kann es danach mit allen anderen teilen.

Quizlet war übersichtlich, hatte alle nötigen Funktionen ohne Schnick-Schnack, es war alles was man brauchte. Leider liegt hier die Betonung auf «war». Wie es so häufig ist, sobald ein Produkt beliebt ist, fängt das dahinterstehende Unternehmen an, möglichst viel Geld herauszuholen. Funktion um Funktion ist hinter die Paywall verschwunden. Aber vor allem wurde Quizlet unübersichtlich und voller Ablenkungen. Es gibt unnötige Lernspiele, nervige Werbung und KI-Assistenzen, welche vom eigentlichen Kern ablenken: Möglichst zeitsparend Vokabeln lernen.

Das hat mich bereits vor 2 Jahren aufgeregt. Ich habe damals gerade gelernt mit Office-Makros, das sind kleine Programme in einem Dokument, zu programmieren. Deshalb habe ich direkt in der Excel-Datei, in der wir die Englisch-Wörter bekamen, begonnen ein Vokabeltrainer zu programmieren. Das Projekt wuchs und der «Vocitrainer» unterstützt inzwischen fünf Sprachen, bietet drei Lernmodi an und hilft mir mit der Aussprache des Wortes direkt im Programm.

Allerdings hat dieser Vokabeltrainer aus verschiedenen Gründen, wie einer mangelhaften Wartung von Seite Microsofts, keine grosse Zukunft. Ich habe immer wieder mit dem Gedanken gespielt, den «Vocitrainer» in der inzwischen erlernten Programmiersprache 'Python' von Grund auf neu zu schreiben. Als in der sechsten Klasse begonnen habe, mir Gedanken über meine Maturaarbeit zu machen, kam schnell die Idee auf, diese Gedanken in die Tat umzusetzen.

Frage: Vergangenheit oder Gegenwart benutzen

Anforderungen und Zielgruppe

Da meine Ressourcen für das Projekt begrenzt sind, ist es wichtig, klare Anforderungen zu setzen und andere «unnötige» Funktionen wegzulassen.

Mir ist es wichtig, das es mit dem neuen Vokabeltrainer möglich ist, ganz unkompliziert selbst erstellte Lernsets mit anderen zu teilen, indem man sie auf einen öffentlichen 'Marketplace' hochlädt. Jeder andere Benutzer kann diesen durchsuchen und sich Lernsets davon herunterladen.

Da aber nicht von Anfang an viele Lernsets bereits zu Verfügung stehen werden, ist es mir wichtig, dass man ganz einfach die Lernsets, welche man zum Beispiel in Form von CSV-Dateien schon hat, importieren kann. Am Besten gibt es ein Tool, dass Lernsets direkt von Quizlet herunterlädt. (Nicht sicher, ob das noch programmiert wird)

Die wichtigste Eigenschaft eines Vokabeltrainers ist es, dass man damit effizient lernen kann. Gerade weil man in der Praxis nicht regelmässig lernt sondern erst die paar Tage vor der Prüfung, ist es wichtig, dass der Algorithmus ein schnelles Lernen fördert. Dabei ist es wichtig,

Erkenntnisse aus der Lernpsychologie und der Funktionsweise des Gehirns direkt in den Abfragealgorithmus einzubauen.

Meine Erfahrung ist es, dass Webapplikationen zur Ablenkung verführen, da sie im Browser offen sind und damit nur ein bis zwei Klicks von den neusten Nachrichten und kleinen Onlinespielen entfernt sind. Da Ablenkung für das Gehirn den Lerneffekt massiv reduzieren, gilt es das Potenzial dafür möglichst zu minimieren. Deshalb und weil ich bereits Python kann, habe ich mich entschieden, dass Programm standalone und damitoffline nutzbar zu programmieren.

Die Zielgruppe sind schlussendlich Benutzer, die sich nicht mit den üblichen Lösungen zufrieden geben, Ablenkung minimieren und Effizienz maximieren wollen: Der typische computeraffine «Poweruser».

Aufbau

Warum Python?

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Client-Server-Model

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Client

Server

Datenbank

Relationale Datenbanken

Tabelle Ordner

Tabelle Vociset

Tabelle Karte

Tabelle Benutzer

Qt-Framework

Das Qt-Framework (Qt ausgesprochen wie das Englische ‘cute’) ist wohl die wichtigste Komponente meiner Arbeit. Deshalb möchte ich ihm ein eigenes Kapitel widmen.

Was ist ein Framework

Den meisten technisch versierten Leser wird ‘Framework’ ein Begriff sein. Es geht darum, dass man nicht alles von Grund auf programmieren kann. Es ist unsinnig, wenn jeder, der auch nur das kleinste Programm mit einem Fenster schreiben möchte, sich zuerst in tausenden von Code

Zeilen darum kümmern muss, wie man dem Betriebssystem sagt, wie das Fenster aussehen soll und wie ein Button funktioniert. Mit einem Framework habe ich den Vorteil, dass ich dem vorherigen Beispiel folgend nur noch zu definieren bräuchte, wie der Inhalt des Fensters lautet. Natürlich kann je nach Framework auch noch genauer beschreiben, wie bspw. das Icon des Fensters und vieles mehr aussehen soll.

Im unterschied zu einer Bibliothek, die eine ähnliche Funktion hat, bestimmt ein Framework auch die Struktur des Programmes und setzt eine definierte Anwendung voraus.

Funktion

Qt ist ein Framework für grafische Benutzeroberflächen auch GUI (Graphical User Interface) genannt. Das Framework ist sehr verbreitet, da es einen grossen Funktionsumfang hat, Open Source ist (jeder kann es verwenden) und in vielen Sprachen verwendbar ist (siehe Absatz PyQt). Qt unter anderem vom legendären VLC Media Player verwendet wie auch von: Adobe Photoshop Elements, Google Earth, VirtualBox, Spotify (Linux Version) oder dem Simulationsprogramm der ESA.[1]

Alles vom Vocitrainer, das man sieht, ist mit Qt umgesetzt.

PyQt

Qt ist, wie viele Bibliotheken für Python in C++ da diese Programmiersprache einfach viel schneller und «genauer» ist und sich deshalb besser dafür eignet. Das heisst aber auch, dass ich Qt in Python nicht einfach so verwenden kann. Es braucht ein Glied zwischen Python und dem in C++ programmierten Qt, das die beiden verbindet. Das nennt man einen Wrapper. Für Qt gibt es 2 Python-Wrapper. PyQt und PySide. Es gibt keine grossen Unterschiede; ich habe mich für PyQt entschieden. PyQt implementiert über 1000 Klassen von Qt.[2]

Beispiel eines Miniprogrammes

Ab hier wird es ein bisschen technischer. Das ist ein Beispiel eines minimalistischen PyQt-Fensters.

```python
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel

# Erstellen einer Qt-Anwendung [1]
app = QApplication(sys.argv)

# Erstellen eines Hauptfensters [2]
window = QWidget()
window.setWindowTitle('Ein PyQt-Fenster')
window.setGeometry(100, 100, 300, 100)  # Position und Größe des Fensters

# Erstellen eines Labels mit Text "Hallo Welt" [3]
label = QLabel('Hallo Welt', window)

# Position des Labels im Fenster festlegen
label.move(100, 40)

# Fenster anzeigen
window.show()

# Anwendung ausführen [4]
sys.exit(app.exec_())
```

Zuerst erstellen wir eine Qt-Anwendung [1]. Dann wird das eigentliche Fenster initialisiert [2]. Dabei kann jedes Widget ein Fenster sein. Es gibt aber auch extra Klassen dafür wie QDialog oder QMainWindow. Nun fügen wir dem Fenster ein einfaches Label hinzu [3]. Als letztes Starten wir das Programm. Ab dann läuft die Event Loop bis das Programm geschlossen wird.

Signale und Slots

Eine wichtige Frage von Gui-Anwendungen ist immer, wie werden Benutzereingaben verarbeitet? Qt hat hier sein eigenes Konzept namens Signals & Slots.

Elemente können in Qt bei einer Aktion ein Signal senden. Zum Beispiel ein Button der geklickt wird. Diese Signale können mit einem Slot verbunden werden. Dieser Slot ist meistens eine selbst geschriebene Funktion, in welcher man dann die Eingabe verarbeitet.

In unserem Beispiel von oben könnten wir folgendes ergänzen.

Wir erstellen eine Funktion, die das Fenster schliesst [1] und verbinden diese als Slot mit dem 'clicked'-Signal des Buttons den wir erstellt haben.

Sobald man nun auf diesen neuen Button klickt, wird das Fenster geschlossen.

```
# Slot für das Schliessen Signal des Buttons [1]
1 usage
def fenster_schliessen():
    window.close()


# Einen Schliessen Button einfügen
cmd_schliessen = QPushButton("Schliessen", window)
cmd_schliessen.move(100, 40)
cmd_schliessen.clicked.connect(fenster_schliessen)  # [2]
```

Wer das Beispiel das wir bis jetzt gemacht haben weiter zu einem grösseren Projekt denkt, merkt, dass das schnell unübersichtlich werden kann. Deshalb brauchen wir zwei Hilfsmittel.

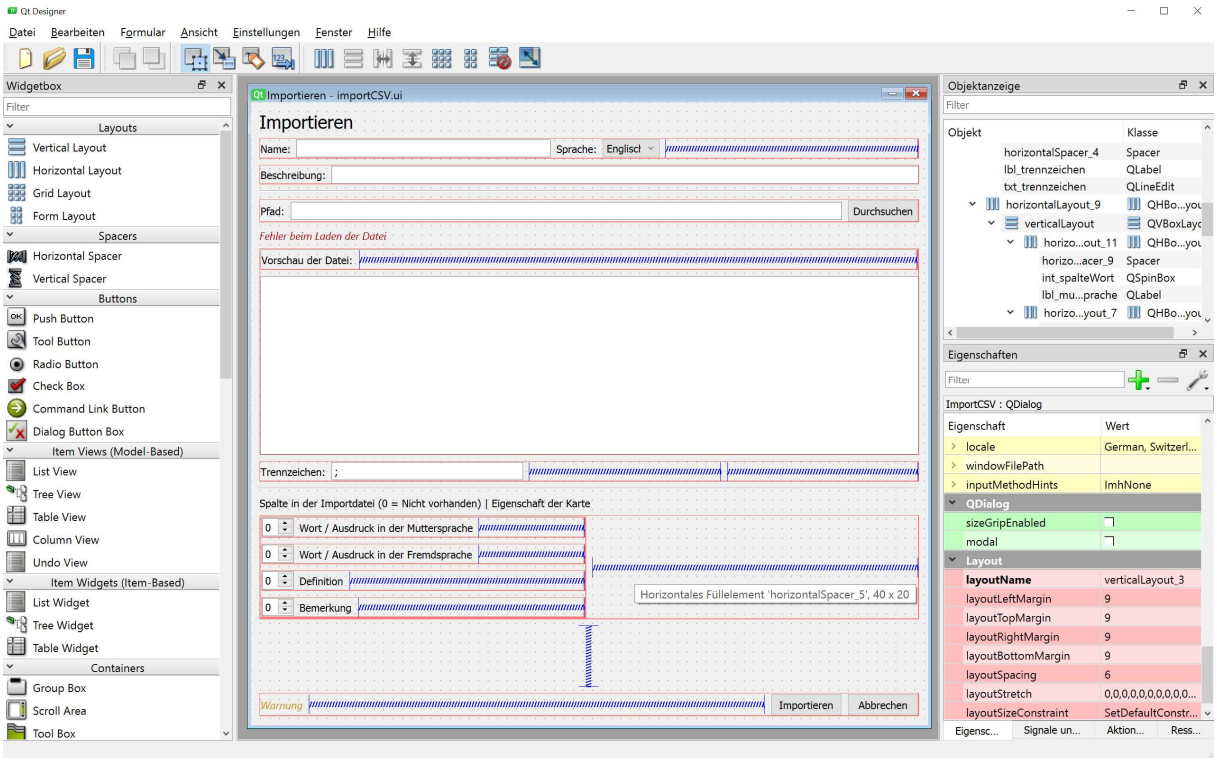
Objekt Orientierte Programmierung

Objekt Orientierte Programmierung kurz OOP wurde mit dem Aufkommen von Grafischen Benutzeroberflächen erfunden. Bei den Unmengen an Objekten, Variabeln und Funktionen braucht es irgend eine Struktur, welche das ganze organisiert. OOP heisst kurz gesagt, dass es Objekte, nämlich Klassen gibt, denen man Funktionen und Variabeln zuordnen kann, welche man dann Methoden und Attribute nennt. So hat beispielsweise jedes QLabel die Methode QLabel.setText(). Auch Fenster lassen sich am besten in selbst geschriebenen Klassen zusammenfassen.

QtDesigner

Das zweite Hilfsmittel erleichtert das Leben sehr. Es geht um das Problem, dass es irgendwann ziemlich schwierig wird, ein Fenster zu gestalten und alle Widgets am richtigen Ort zu platzieren, weil man immer ausprobieren muss, wo sie stehen, bis sie am richtigen Ort sind. Umso komplizierter wird es, wenn dann noch responsive Layouts dazu kommen.

Die Lösung ist der mit Qt mitgelieferte QtDesigner. Er ermöglicht es, dass Fenster per Drag and Drop zusammen zu bauen.



Hier ein Bild des Fensters zum Importieren aus CSV-Dateien. Der wichtigste Bereich ist in der Mitte, das Fenster. Links sind die verfügbaren Widgets und rechts unten kann man die Eigenschaften des angewählten Widgets bearbeiten.

QtLinguist

Hier möglicherweise noch ein Absatz zur Übersetzung von Gui-Programmen

Bibliotheken und Tools

Pythons Standartbibliothek

socket

pickle

ssl

sqlite

csv

typing

PyCharm

Git

ChatGPT

Die Psychologie des Lernens

Wiederholung macht den Meiser

Die Zahl 7

Pausen

Dokumentation

[1] Wikipedia über Qt: https://de.wikipedia.org/wiki/Qt_(Bibliothek)#Verwendungsbeispiele
[2] PyQt5 Dokumentation: https://www.riverbankcomputing.com/static/Docs/PyQt5/introduction.html