

Dokumentation „Vocabulary Pet“

von marionline

Inhalt

Dokumentation „Vocabulary Pet“	1	Die Funktion zum Überprüfen der	
Einleitung.....	2	Antwort.....	9
Projektidee.....	2	Anleitung zur Ausführung.....	12
Definition.....	3	Installieren.....	12
Was sind typische Elemente von		Das Programm.....	12
Vokabeltrainersoftware?.....	3	Getting Started.....	12
Anki und Mnemosyne.....	3	Benutzeranleitung.....	13
Memrise.....	3	Pet's Home.....	14
Was ist ein Tamagochi?.....	4	Status anschauen.....	14
Wie motiviert das Programm zum lernen? ..	4	Das Vokabellernen im Lernmodus.....	15
Hintergrundinformationen.....	4	Die Wortliste.....	16
Was ist Gamification?.....	4	Einkaufen von Gegenständen und an das	
Womit wird das Programm umgesetzt?.....	5	Pet verfüttern.....	17
Begründungen.....	5	Verwendete Tools.....	18
Warum Bottle?.....	5	Tool: Bottle Framework.....	18
Warum sqlite3 und javascript?.....	5	Tool: SQLite Manager Plugin für Firefox	18
Und warum überhaupt so ein Programm? ..	5	Reflexion.....	18
Projektplan.....	5	Wurde das Ziel erreicht?.....	18
Projektaufbau.....	6	Was lief schief?.....	20
Einzelne Elemente des Programms.....	7	Lösungen?.....	20
Element: game.py.....	7	Fazit.....	21
Element: Websitentemplates.....	7	Verwendete Literatur:.....	21
Element: Datenbank.....	8	Gamification:.....	21
Element: Javascript.....	8	SQL:.....	21
Einzelne Funktionen des Programms:.....	9	Javascript:.....	21

Einleitung

Dieses Projekt entstand im Rahmen des Kurses 'Angewandte Informationswissenschaft' des Moduls I4. Der Kurs fand in den Sommersemesterferien 2016. Da es keine feste Themenvorgabe gab, konnten wir uns selbst ein Projekt aussuchen, um es mit Mitteln unserer Wahl umzusetzen.

Projektidee

„Ein Vokabellernprogramm, das besonders gut zum L[ernen] motivieren soll. Dazu wird ein Vokabeltrainer mit einem virtuellen Haustier kombiniert.“ - *Quelle: Projektplan.md*

Definition

Was sind typische Elemente von Vokabeltrainersoftware?

Mir bekannte Beispiele für Vokabeltrainersoftware sind unter anderem:

- Anki (<http://ankisrs.net/>)
- Memrise (<https://www.memrise.com/>)
- Mnemosyne (<http://mnemosyne-proj.org/>)

Anki und Mnemosyne

Anki und Mnemosyne sind OpenSource-Programme, sie laufen offline. Anki kann mit Plugins erweitert werden. Mnemosyne ebenfalls. Beide können über das Internet synchronisiert werden, um die Vokabeln auf dem gleichen Stand zu halten, wenn man von verschiedenen Geräten aus lernt. Die beiden Programme funktionieren nach dem Karteikartenkasten-Prinzip. Eine Frage wird angezeigt, wenn der Nutzer klickt erscheint die Antwort. Dann bewertet der Nutzer auf einer Skala wie schwer oder leicht ihm die Lösung fiel. Die Karte erhält dem entsprechend ein neues "Fälligkeitsdatum", an dem sie dem Nutzer erneut angezeigt werden wird.

Memrise

Es handelt sich hierbei um ein kommerzielles Produkt, allerdings kann man es kostenlos nutzen. Memrise benötigt eine Internetverbindung und läuft im Browser. Für einige Smartphonebetriebssysteme gibt es kostenpflichtige Apps. Den Content, den man auf der Memrisewebsite erstellt, kann die Firma weiter nutzen. So füllen die Nutzer als Prosumer die Vokabeldatenbank. Memrise arbeitet mit Bildern, sogenannten Mems die einer Vokabel zugeordnet und bei fehlerhafter Antwort angezeigt werden, um das lernen zu erleichtern. Erstellte Bilder und Vokabeln, werden in Datenbanken gespeichert. Es gibt allgemeine Datenbanken, aus denen Nutzer Vokabeln beziehen können, sowie Datenbanken nur für bestimmte Kurse, bzw. Sprach-Kombinationen. Gewinn macht Memrise dadurch, dass die Nutzer gegen monatliche Gebühr ihre Accounts upgraden können, wodurch ihnen mehr Features zur Verfügung stehen.

Von den drei Beispielen nutzt Memrise die meisten Gamificationelemente: Lernende erhalten Punkte für das erfolgreiche Beantworten der Vokabelabfragen. Die zu lernenden Vokabeln symbolisieren Blumen die gepflanzt werden sollen, durch Lernen gießt man sozusagen die Vokabel-

Blumen. Auch werden Level benutzt: Mit bestimmter Punktzahl steigen Nutzer auf und erhalten einen höheren Rang, dies zeigt sich am Logo.

Das Programm ist gut gemacht, aber, nach einiger Zeit wird es langweilig und eintönig. Der Abstand gerade bei hohen Leveln ist sehr groß und die Punkte zu sammeln ist an sich sind ziemlich nutzlos.

Was ist ein Tamagochi?

Tamagochi ist ein Spielzeug für Kinder, das ein Monster bzw. Haustier simuliert um das der Spieler sich kümmern muss, sonst stirbt es. Je nachdem wie gut man das Tamagochi pflegt entwickelt es sich weiter, das heißt es ändert Form und Aussehen. Zu Zeiten des PokemonGo-Hypes, einem Spiel wo Spieler Monster fangen und großziehen um sie gegeneinander Kämpfen zu lassen, spricht wenig gegen das Spielelement Monsteraufzucht.

Wie motiviert das Programm zum lernen?

Der Spieler erhält für das richtige Beantworten von Vokabelfragen Punkte.

Diese Punkte werden für die Pflege eines Monsters, das im folgenden als Pet bezeichnet wird, verbraucht. Das Monster, welches ebenfalls benannt werden kann, wird nach einiger Zeit hungrig.

Um das Monster zu Füttern kann der Player seine erspielten Punkte gegen Items, wie z.B. Beeren oder Schokolade eintauschen mit denen die Kreatur gefüttert werden kann.

Ursprünglich war geplant, dass das Pet wenn es nicht gefüttert wird, letztendlich verhungert.

Das Grundprinzip bleibt jedoch unverändert: Das Monster dient als "Punkte-Drain", so dass der Spieler ständig neue Punkte 'erlernen', bzw. erspielen muss. Es wird ein Anreiz zum lernen gegeben.

(Quelle: Projektplan.md)

Hintergrundinformationen

Was ist Gamification?

Gabe Zichermann und Christopher Cunningham (2011) definieren den Begriff Gamification wie folgt:

The process of game-thinking and game mechanisms to engage users and solve problems.'

Gamification kann nach Aussage der Autoren eine Änderung des Verhaltens der Nutzer hervorrufen, wenn diese eine Erfahrung als Spiel betrachten und durch Belohnung geködert werden. (Cunningham und Zichermann 2011, S. xiv)

Womit wird das Programm umgesetzt?

Bottle, sqlite3, html und ein wenig javascript und css.

Begründungen

Warum Bottle?

Bottle scheint mir einfacher zu benutzen als Flask oder ähnliche Frameworks. Dafür kann es auf der andere Seite weniger. Ich wollte das Programm im Webbrowser bauen, weil ich a) am liebsten eine Version für mehrer Benutzer erstellt hätte -und da bietet sich etwas mit Server an - und b) weil ich etwas anderes als Tk ausprobieren wollte. Außerdem dachte ich mit Html lässt sich eine Website einfach gestalten, einfacher als wenn man mit z.B. Pygame versucht Schaltflächen und Knöpfe zu erstellen.

Einen lokalen Server zu nehmen, ist – wie in der ersten Vorstellung des Projekts geschrieben - ein Art Kompromiss zwischen Online und Offline. Ich wollte etwas über den Umgang mit Frameworks lernen. Die Vorteile sind, dass ich mir da das Programm im Singleplayermodus läuft weniger Gedanken über schummelnde Spieler, Netzwerkprogrammierung und Deployment machen muss. (Wer schummeln will schreibt sich jetzt einfach in der Datenbank 1000 Punkte gut.) Andererseits fällt somit die Interaktivität als Gamificationelement weg, und somit eine handvoll interessante Spielelemente wie Bestenlisten, Item- oder Vokabelaustausch.

Warum sqlite3 und javascript?

Die Verwendung eine Datenbank war ungeplant und ergab sich daraus, das Bottle keine Zustände speichert und ich kein csv parsen wollte. Javascript ist auch eine Notlösung, um per Intervall beim Server Daten abzufragen und eine Aktualisierung der Daten zu veranlassen.

Und warum überhaupt so ein Programm?

Meine Motivation so ein Programm zu schreiben ist, ich hätte gerne ein solches Programm zum Vokabellernen gehabt.

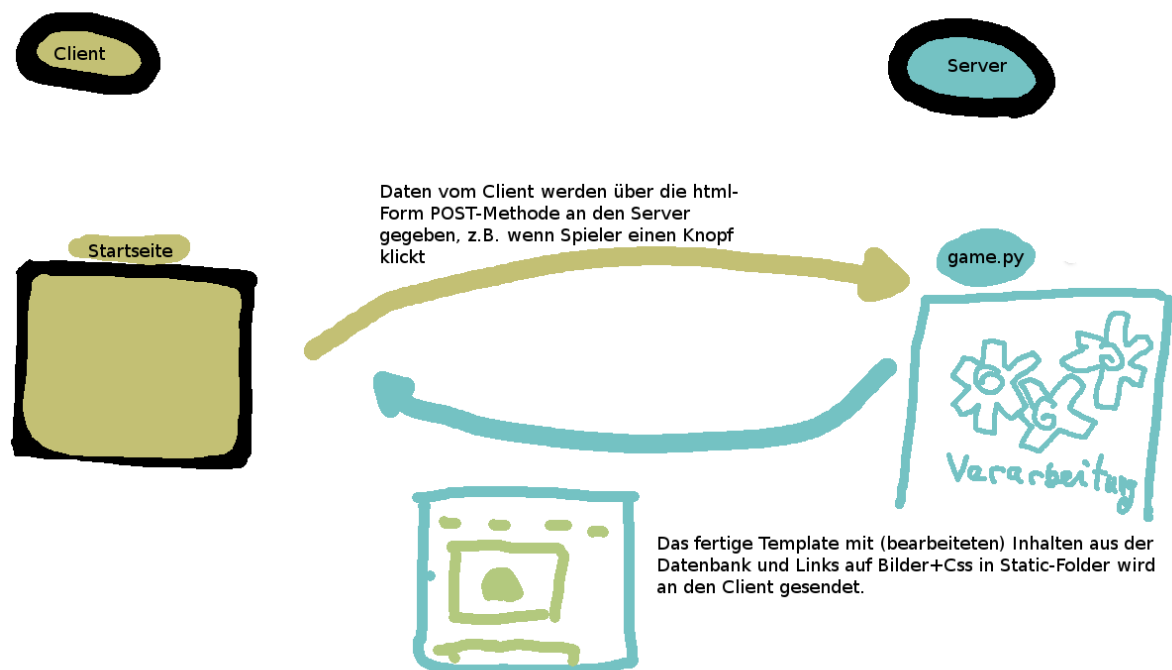
Projektplan

Der Projektplan kann nachträglich als zu ambitioniert angesehen werden. In der Umsetzungsphase der Projekts habe ich bewusst auf die meisten Features verzichtet und mich nur auf die wesentlichsten Funktionen konzentriert, die das Programm benötigt um nutzbar zu sein.

(Für weitere Informationen, z.b. wie der Aufbau geplant war siehe Projektplan.)

Projektaufbau

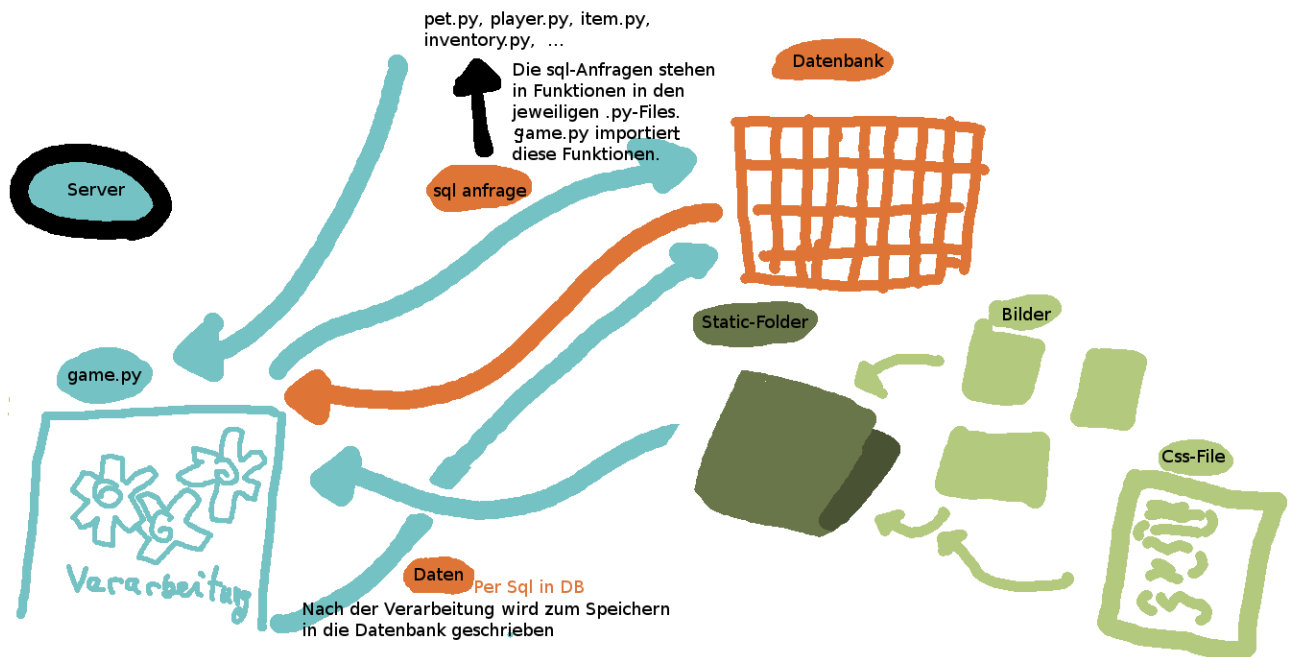
Der Programmaufbau ergab sich aus den Vorgaben die das Bottleframework macht. Über den Aufruf einzelner Urls im Webbrowser lassen sich Funktionen starten, die dem Client Information senden können, zum Beispiel html zum Aufbau einer Website aufbaut oder aber auch Daten im Json-Format.



Drawing 1: Client and Server Kommunikation

Bottle arbeitet nicht statisch und bei jedem Seitenaufruf wird das Programm erneut ausgeführt. Daraus ergibt es zwangsläufig, dass alle Daten die nach dem erneuten Aufruf noch vorhanden sein sollen gespeichert und beim Neuladen ins Programm gelesen werden müssen. Die Reihenfolge ist:

Anfrage vom Client erhalten → Daten auslesen, → verarbeiten, → speichern, → Template mit Daten füllen und als Antwort an Client senden.



Einzelne Elemente des Programms

Das Programm besteht im Groben aus den Pythondateien die die Funktionalitäten des Bottleservers steuern, Templatedateien (.tpl), die html und selten javascript enthalten zum Aufbau der Websites, sowie statische Dateien (im Ordner namens static) wie unter anderem eine Cssdatei und Bilder. Die Dateien im Ordner 'static' werden vom Server auf den Websites dargestellt. Der Server legt Daten in einer Sqllitedatenbank 'game.db' ab und liest sie auch wieder aus, bei jedem Neuaufbau der Seite.

Element: game.py

Game.py nutzt die Funktionen des Bottleframework um Htmlseiten zu erzeugen, die im Webbrowser angezeigt werden.

Element: Websitentemplates

Die Templatefiles .tpl sind Schablonen aus denen Webseiten generiert werden. Sie werden vom Programm mit Inhalt gefüllt. Dazu werden Parameter in game.py weitergegeben. Die Templateengine von Bottles unterstützt Parameter sowie if-conditions oder for-Schleifen.

Element: Datenbank

Die Datenbank hat verschiedene Tabellen um Daten zu verwalten. Ich verwende eine sqlite-Datenbank. Die Daten in den Tabellen verweisen wenn nötig auf Daten in anderen Tabellen, zum Beispiel im Fall des Inventars.

Element: Javascript

Das Programm nutzt zwei Javascriptfunktionen: EineInterval-Funktion und eine jquery-Funktion zum Laden der vom Server gesendeten json-Datei. Json steht für 'Javascript object notation' und die Darstellung von Json entspricht der Darstellung eines Objekts in Javascript. Beide Funktionen sind in einander verschachtelt, so dass in einem vorgegebenen Intervall (1000 milisekunden) ein Teil der Seite nachgeladen wird.

Code (ohne Kommentare):

```
var myVar = setInterval(myTimer, 1000);

function myTimer() {

    $.getJSON('http://localhost:8080/update', function(data){

        // data.msg = data['msg'] in python

        document.getElementById('body').src= data.body;

        document.getElementById('face').src= data.face;

        document.getElementById('deco').src= data.deco;

        if (data.speechbubble != "") {

            document.getElementById('speechbubble').src= data.speechbubble;

        };

    });

}
```

Die myTimer Funktion ruft sich jede Sekunde erneut auf. \$.getJSON('http://localhost:8080/update', function(data) ruft die Url 'localhost:8080/update' auf, diese triggert beim Server die Funktion zum Update der Petdaten. Außerdem wird eine json-Datei an die anfragende Website zurück gesendet. Die json-Daten werden in die Function function als

Parameter gegeben. Sie beinhalten Pfadangaben der drei Bildebenen (Body, Face, Deco) aus denen das Bild des Pets besteht geladen. Auch kann wenn vorhanden, also nicht als leerer String angegeben eine Sprechblase angezeigt werden.

Einzelne Funktionen des Programms:

Im Programm haben wir zwei Arten von Funktionen: In game.py stehen die Funktionen zum Websiteaufbau und in den anderen Pythondateien die Funktionen zum Lesen und Schreiben in die Datenbank. Exemplarisch stelle ich hier im Detail nur die Funktion zum Überprüfen der Antwort, als wichtigste Funktion des Programms, vor.

Die Funktion zum Überprüfen der Antwort

Die Funktion **learn_check_answer()** wird ebenfalls durch Aufruf der Url 'server:port/learn' gestartet, der Zusatz 'method=POST' sorgt dafür, dass die nur geschieht wenn Daten per Post an den Server übertragen werden.

```
@route('/learn', method='POST')
```

```
def learn_check_answer():
```

Als erstes werden die Daten über den Spieler aus der Datenbank gelesen.

```
player_data_list = read_everything_from_player_table()
```

```
player_id_num, player_name, score, id_of_last_question_asked, combo = player_data_list[0]
```

Die Antwort die in der HTML-Formular eingegeben wurde wird in der Variable answer gespeichert.

```
answer = request.forms.get('answer')
```

Daten über das abgefragte Wort werden aus der Datenbank gelesen.

```
voc_list = read_value_from_table_voc('*', 'id', id_of_last_question_asked)
```

```
voc_id_num, question, expected_answer, times_answered_correctly, question_language,  
answer_language = voc_list[0]
```

Die Antwort, die per POST an den Server gesendet wurde muss angepasst werden um Probleme wegen der Zeichenkodierung zu vermeiden, da Umlaute nicht richtig angezeigt werden.

```
answer = answer.replace("Ã", "ä").replace("Ã¶", "ö").replace("Ã¼", "ü")
```

Die Variablen msg und points werden erstellt. Msg dient dazu Mitteilungen an den Spieler auf der Website anzuzeigen. Points ist der Punktestand den der Spieler durch eine richtige Antwort hinzugewinnen kann.

```
msg="
```

```
points = 0
```

Die Daten des Pet werden ausgelesen.

```
pet_data = read_everything_from_tabel_pet()
```

```
pet_id_num, pet_name, pet_life_points, pet_current_life_points, pet_current_state, pet_body,  
pet_face, pet_deco = pet_data[0]
```

Die Variable combo hätte sequenz-bonus heißen müssen, sie misst ob der Spieler einen Punktebonus erhält. Dieser Punktebonus dient als zusätzliche Lernmotivation, wenn der Spieler ohne Fehler eine bestimmten Anzahl von Vokabeln beantwortet, erhält er extra Punkte (siehe weiter unten).

```
if combo == None:
```

```
    combo = 0
```

```
if times_answered_corectly == None:
```

```
    times_answered_corectly = 0
```

Hier werden die Antwort des Spielers und die richtige Antwort verglichen. Bei einer richtigen Antwort wird der Gesichtsausdruck des Vokabelpets angepasst, damit es so aussieht als ob es sich freut. Bei falscher Beantwortung wird die Mimik auf neutral oder unglücklich gesetzt.

```
if answer.lower() == expected_answer.lower():
```

```
    times_answered_corectly += 1
```

```
    happy_faces = ['happy01.png', 'happy02.png']
```

```
    face = random.choice(happy_faces)
```

Ab hier wird in verschiedenen Abstufung geschaut wie viele Bonuspunkte der Spieler erhält. Macimal 100 richtige Antworten in einer Reihe werden belohnt. 100 richtige Antworten in einer Reihe halte ich für eine extrem hohe Zahl, die unwahrscheinlich zu erreichen ist, aber gleichzeitig dient dies als weitere Motivation.

Wenn der Spieler einen Bonus erhält, gibt es auch gleichzeitig ein Lob.

```
if combo > 100:
```

```
    combo = 0
```

```
    points = 200
```

```
    combo += 1
```

```
    msg = ["Unbelievable!", "You answered correctly 100 times in a row.", "That's astonishing!  
You reward are 200 points."]
```

Die einzelnen Teile der Funktion sind sehr ähnlich, daher habe ich den Rest ausgelassen.

Wenn es keinen Bonus gibt, wird stattdessen der Score angezeigt.

```

else:

    points += 1

    combo += 1

    score += points

    msg=['Your score is: {}'.format(score)]

    score += points

```

Bei falscher Beantwortung erhält der Spieler keine Extrapunkte und die Combovariable, die richtige Beantwortung hintereinander misst wird zurück auf 0 gesetzt.

Außerdem erhält der Spieler einen Hinweis war er falsch gemacht hat. Die Frage, die Zielsprache, die richtige Lösung und die falsche Antwort des Spielers werden angezeigt. Das Pet macht ein trauriges Gesicht.

```

else:

    combo = 0

    points = 0

    msg=['The question was: {}'.format(question), 'The target language was
    {}'.format(answer_language), 'The correct answer is: {}'.format(expected_answer), 'You said:
    {}'.format(answer)]

    unhappy_faces=['angry01.png', 'angry02.png', 'outch01.png', 'outch02.png', 'neutral01.png',
    'neutral02.png']

    face = random.choice(unhappy_faces)

```

Die geänderten Werte Punktstand des Spielers und Combo, und wie oft die Vokabel richtig beantwortet wurde werden in der Datenbank gespeichert.

```

alter_tabel_player_where('score', score, 'id', player_id_num)

alter_tabel_player_where('combo', combo, 'id', player_id_num)

alter_tabel_voc_where('times_answered_correctly', times_answered_corectly, 'id', voc_id_num)

```

Nach dem Speichern werden die Vokabeln ausgelesen und eine neue Frage wird ausgewählt, (für Details siehe oben).

```

data = read_everything_from_tabel_voc()

sorted_data = sorted(data, key=itemgetter(3))

my_dict={}

for element in sorted_data:

    if element[3] in my_dict.keys():

```

```

        my_dict[element[3]].append(element)
    else:
        my_dict[element[3]] = []
        my_dict[element[3]].append(element)
    keys = sorted(list(my_dict.keys()))
    output= random.choice(my_dict[keys[0]])
    alter_tabel_player('id_of_last_question_asked', output[0])

```

Das Template wird gerendert und im Webbrowser angezeigt, mit neuer Frage, Nachricht an den Spieler und passender Mine des Pets.

```

    return template('learn', msg=msg, question=output, menu=NAV, footer=FOOTER, face=face,
body=pet_body, deco=pet_deco)

```

Anleitung zur Ausführung

Installieren

Aus Zeitgründen und auch weil ich noch nie ein 'sich-selbst-installierndes' Programm geschrieben habe und mir die Fehleranfälligkeit zu groß (oder anders gesagt die verbleibende Zeit um Fehler auszubauen zu gering erschien), gibt es nur eine Installationsanleitung. Getestet wurde die Installation auf Linux Mint 18. Das Programm läuft auch unter Xubuntu und Kanotix.

Das Programm

Getting Started

Folgende Einstellungen sind in der Datei **hidden_server_data.py** vorzunehmen:

Normalerweise ist der Server 'localhost'. Port kann zum Beispiel 8080 sein.

Achtung: Wichtig ist es den Pfad zu static Dateien anzupassen, wie im Readme.md beschrieben wurde.

Den Bottle-Server startet man in der Terminal/Shell im Verzeichnis des Programms per:

```
python3 game.py
```

Das Programm läuft im Webbrowser unter dem in der **hidden_server_data.py** angegebenen Host und Port.

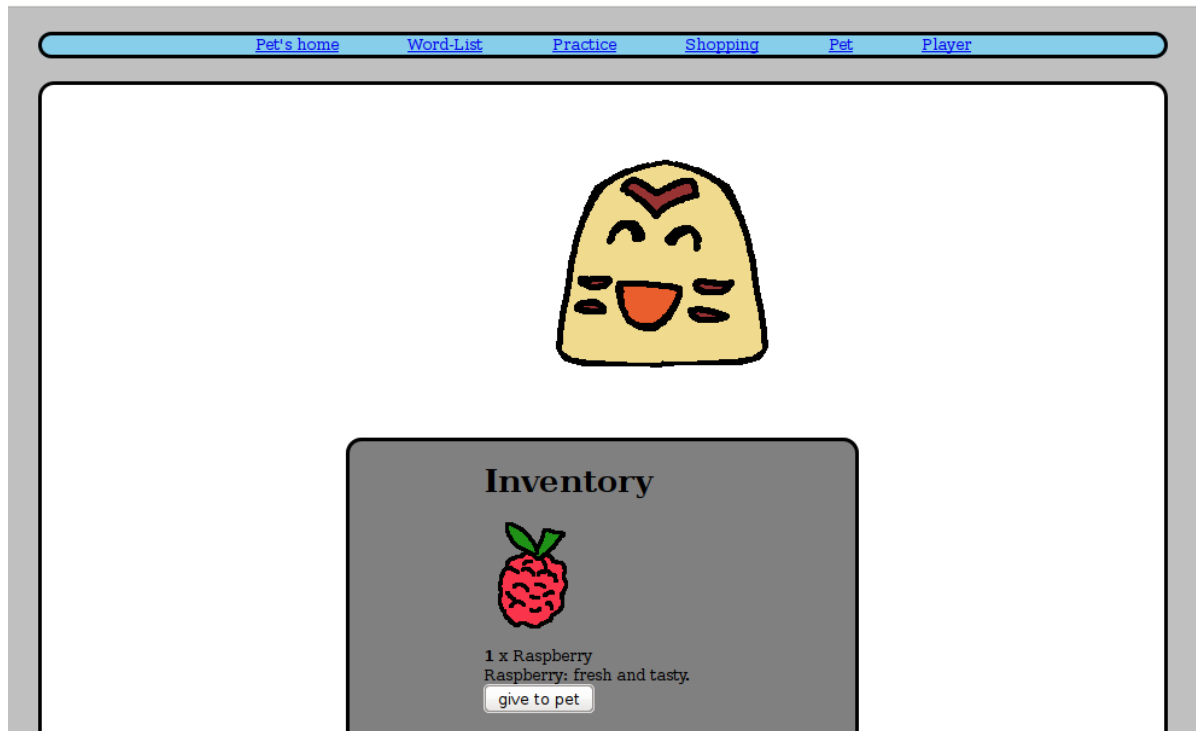
Achtung: Falls du einen Scriptblocker benutzt muss man ihn zur Benutzung ausschalten, denn das Programm benötigt Javascript und jquery.

Ich verweise an dieser Stelle auf die ausführlichere readme-Datei.

Benutzeranleitung

Der folgende Teil der Dokumentation richtet sich an Benutzer und erklärt anschaulich die Funktionsweise des Programms. Die folgenden Abschnitte adressieren den Leser als 'Du', da das Programm sich hauptsächlich an Schüler und andere Lernende richtet, und ich diesen Teil später als Handbuch verwenden werde.

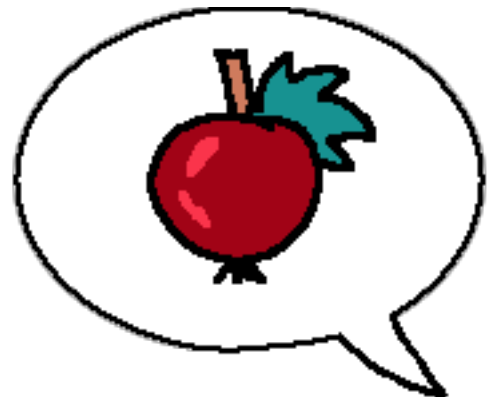
Pet's Home



Drawing 2: Pet's home

Pet's home zeigt das Pet und seine Stimmung an. Außerdem siehst du hier das Inventar. Während du Pet's Home anschaust ändert sich langsam die Stimmung des Haustiers.

Wenn das Pet hungrig ist erscheint eine Sprech- oder Gedankenblase in der eine Beere abgebildet ist.

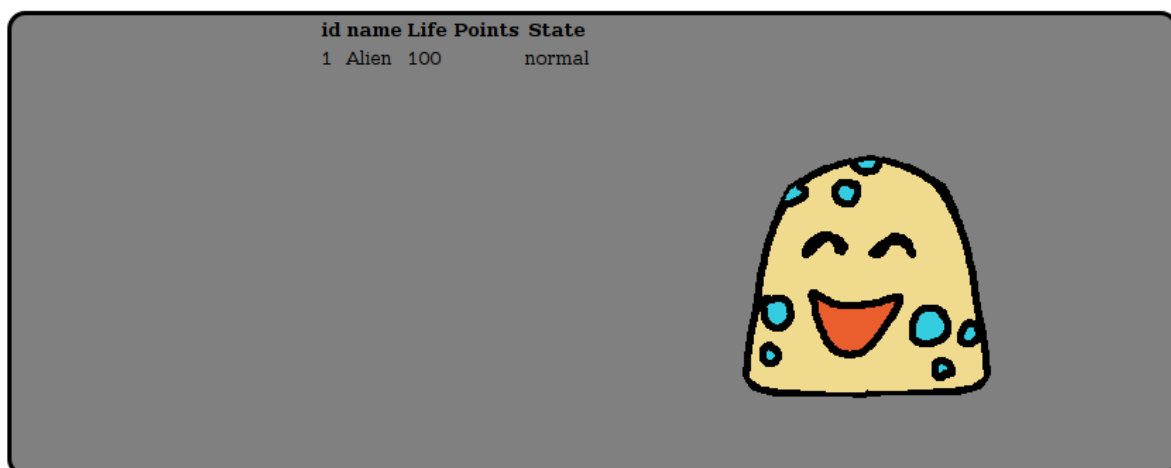


Status anschauen

Wenn du dir - bei nur zwei verschiedenen Stati - mal

unsicher bist, welche Laune dein Vokabel Pet gerade plagt, kannst du unter dem Link **Pet** nachschauen.

Beere



Drawing 4: Pets Statusinformation

Deinen eigene Status kannst du unter **Player** nachschauen.

id	name	credits	answered right in sequence
1	marionline	590	8

Drawing 5: Playerbildschirm

Das Vokabellernen im Lernmodus

Der Vokabellernenbildschirm findest du unter **Practice**.

Der wichtigste Fenster ist in der Bildschirmmitte. Hier steht die Frage hervorgehoben in blauer Schrift. Außerdem gibt es Informationen zur Zielsprache. Hier gibst du die Antwort ein und bestätigst mit einem Klick auf 'send'.

Wenn du richtig geantwortet hast, wird du etwas sehen wie in Abbildung Drawing E. Im oberen Bildschirmbereich, der für Informationen gedacht ist, wird der neue Score eingeblendet.

Für viele richtige Antworten in einer Reihe erhältst Bonus-Punkte z.B. nach 10 richtigen Antworten in einer Reihe. Nachrichten über Bonuspunkte werden ebenfalls im oberen Bildschirmbereich eingeblendet.

Your score is: 591

Question: **Ich bin ...**

Question Language: German

Please translate into English (and include all given punctuation marks).



Drawing 6: User hat richtig geantwortet

Wenn du nicht richtig geantwortet hast, wird dir stattdessen die richtige Lösung angezeigt und deine Antwort zum Vergleich. Dein Punktestand wird sich bei einer falschen Antwort nicht ändern und wird nicht angezeigt.

The question was: Ich habe Hunger.

The target language was English.


The correct answer is: I am hungry.

You said: I am hungri

Question: Wo ist ...?

Question Language: German

Please translate into English (and include all given punctuation marks).



Drawing 7: Lernbildschirm nach falscher Antworteingabe

Die Wortliste

id	question	answer	question language	target language	score	
3	<input type="text" value="Guten Abend"/>	<input type="text" value="good evening"/>	<input type="text" value="German"/>	<input type="text" value="English"/>	19	<input type="button" value="submit changes in row"/>
4	<input type="text" value="Wo ist ...?"/>	<input type="text" value="Where is ...?"/>	<input type="text" value="German"/>	<input type="text" value="English"/>	19	<input type="button" value="submit changes in row"/>
5	<input type="text" value="Ich bin ..."/>	<input type="text" value="I am ..."/>	<input type="text" value="German"/>	<input type="text" value="English"/>	19	<input type="button" value="submit changes in row"/>
6	<input type="text" value="Ich habe Hunger."/>	<input type="text" value="I am hungry."/>	<input type="text" value="German"/>	<input type="text" value="English"/>	19	<input type="button" value="submit changes in row"/>
7	<input type="text" value="Help!"/>	<input type="text" value="Hilfe!"/>	<input type="text" value="English"/>	<input type="text" value="German"/>	19	<input type="button" value="submit changes in row"/>
8	<input type="text" value="Bye!"/>	<input type="text" value="Dui dui!"/>	<input type="text" value="English"/>	<input type="text" value="Dutch"/>	18	<input type="button" value="submit changes in row"/>

Drawing 8:






Wenn du deine Vokabeln überarbeiten möchtest nutze die Wordliste (**Wortlist**). Sie zeigt dir all

deine Vokabeln an. Hier wird die ID der Frage angezeigt. Du kannst Frage, Antwort, die Sprache der Frage und die Sprache der Antwort nachsehen und ändern. Außerdem siehst du wie oft du die Vokabel schon richtig beantwortet hast.

Achtung: Vergiss nicht nach dem du Änderungen vorgenommen hast, auf den Knopf 'Submit changes in row' in der jeweiligen Zeile zu klicken.

Einkaufen von Gegenständen und an das Pet verfüttern

Das Pet wird nach einiger Zeit hungrig und verlangt nach Futter. Unter dem Menüpunkt **Shopping** kannst du Punkte (Credits), die du im Lernmodus (**Practice**) erspielt hast, gegen Futter tauschen. Es wird eine Liste mit Items angezeigt. Oberhalb der Liste wird angezeigt wie viele Credits du besitzt.

You have 17 credits.				
name	price	description	image	buy?
Berry	5	This red berry makes good food for pets.		<input type="button" value="buy"/>
Raspberry	10	Raspberry: fresh and tasty.		<input type="button" value="buy"/>
Sandwich	15	A sandwich made from bread, cheese and salad. Yum!		<input type="button" value="buy"/>
Melon	25	A tasty piece of melon.		<input type="button" value="buy"/>
Chocolate	50	A delicious chocolate bar.		<input type="button" value="buy"/>

Drawing 9: Futter für das Pet

Tipp: Je teurer der Gegenstand um so besser hilft er gegen Hunger.

Verwendete Tools

Tool: Bottle Framework

Bottle (Link: <http://bottlepy.org/docs/dev/index.html>) ist ein Mikroframework mit dem man Webapplikationen in Python erstellen kann. Leider ist es nicht statisch. Bei diesem Vokabellernspiel müssen u.a. Daten über Lernfortschritt und Status des Pets gespeichert werden. Daher musste eine Lösung gefunden werden damit der Server beim Neuladen der Seite alle Vokabeln und Punktstände „vergisst“.

Tool: SQLite Manager Plugin für Firefox

Das SQLite Plugin für Firefox (Link: <https://addons.mozilla.org/de/firefox/addon/sqlite-manager/>) erwies sich als sehr hilfreich und nützlich. Es ist einfach zu bedienen, läuft über den Firefoxwebbrowser -und damit auf PC und Laptop- und es macht den Umgang mit Datenbanken sehr viel anschaulicher. Es eignet sich sehr gut zum testen, weil man ohne weiteres Werte in den Tabellen ändern kann.

Reflexion

Wurde das Ziel erreicht?

Im Projektplan steht:

Ziel

** Mein Ziel: ein schönes kleines Projekt zum laufen bekommen :)*

** Die Punkte der Must-Have-Liste umsetzen*

Must-Have:

** Gui - Anzeige des Programms über das der Nutzer mit dem Spiel interagiert*

** Monster*

** Spieler*

** Lernmodus (Multiple Choice)*

** Status: Hunger - Monster wird mit der Zeit [~~Hunger~~-hungrig]*

- * *Items: Futter*
- * *Export (json)*
- * *Vokabelliste schon vorgeben mit 10 Vokabeln*
- * *Progressbar*
- * *Dokumentation*
- * *Animationen*

Quelle: Projektplan

Das Ziel zu erreichen ist mir nur teilweise gelungen. Die Must-Have-Liste war zu ambitioniert. Andererseits habe ich mich im Rahmen der Projekts in SQL eingearbeitet. Etwas Neues zu Lernen war unter anderem auch Motivation für mich am Projektseminar teilzunehmen, insofern bin ich sehr zufrieden.

Aus Zeitgründen musste ich jedoch viele Features weglassen, die ich gerne eingebaut hätte. Vor allem ein Mehrspielermodus ist nicht umsetzbar gewesen. Ein Mehrspielermodus hätte jedoch einige interessante Gamificationelemente z.B. Bestenlisten oder Itemaustausch ermöglicht (Gabe Zichermann & Christopher Cunningham, 2011, S.24-25).

Badges (Gabe Zichermann & Christopher Cunningham, 2011, S.55-59), also Auszeichnungen, die ich als zusätzliche Motivation für den Spieler eingeplant hatte, konnte ich ebenfalls nicht einbauen. Es fehlte mir neben Zeit auch ein gutes Konzept dafür.

Ich habe ebenfalls auf mehrere Stati/Launen für das Pet verzichtet. Es ist geplant weitere Stati und Items einzubauen.

Letztendlich habe ich aus Zeitgründen sogar auf Must-Have-Features verzichtet, um eine abgabefertige Version des Programms für die Erstellung der Dokumentation zu haben.

Nicht umgesetzt wurden:

- Multiple Choice Lernmodus
- Export der Vokabeln als json
- Progressbar, der während des Lernens angezeigt wird
- Animationen (zum Beispiel, wenn das Pet etwas isst)
- Das Pet verhungert

Was lief schief?

Das Speichern der Daten, also die Persistenz war ein großes Problem, wesentlich größer als erwartet. Das lies sich mit einer SQLite-Datenbank lösen, aber das Programm läuft, aber es ist recht langsam. Schwierig war auch die Verknüpfung von Items und Moods (Launen) des Pets. Das Konzept hat mir mein Vater erklärt.

Ein Problem für das ich keine gute Lösung gefunden habe ist, das Bottle nicht mit Unicode zu arbeiten scheint. Wenn im Lernmodus Vokabeln mit Umlaut abgefragt wurden, wurden diese in der game.py falsch dekodiert und die richtige Nutzereingabe somit als falsch bewertet. Ich habe für Ö,Ä und Ü die replace- Funktion von Python benutzt. Sprachen mit einem anderen Schriftsystem wie etwa Chinesisch oder Arabisch lassen sich so leider nicht mit dem Vokabel Pet lernen. Französisch muss man die nicht-ASCII Zeichen und deren Codierung von Hand ins Programm schreiben.

Ein weiteres Problem ist, dass Webbrowser sich Nutzereingaben merken können und somit der Spieler Vokabeln nur einmal richtig eintippen muss, und ansonsten die Eingabe aus dem Dropdownmenu wählen kann.

Lösungen?

Anstatt es mit .csv und parsing zu versuchen habe ich die Gelegenheit genutzt und mich ein wenig in SQL eingearbeitet. Dabei habe ich feststellen müssen das SQL mehr kann als nur ein paar Tabellen mit Daten füllen und wieder auslesen, wie zuerst angenommen. Da aber das Ziel die Fertigstellung des Programms war, habe ich die weiteren Möglichkeiten von SQL ignoriert.

Ebenfalls ungeplant war, ohne Klassen zu arbeiten gearbeitet. Da bei jeden Seitenaufruf das Bottle-Program erneut ausgeführt wird, erschien es überflüssig mit Klassen zu arbeiten, die stets erneut initialisiert werden. Die Daten in der Datenbank könnte man vielleicht als eine Art „Ersatzklasse“ betrachten. Es handelt sich dabei zumindest um jene Werte, die beim initialisieren als Parameter an die eigentliche Klassen im Programm gegeben würden.

Für verschiedene Probleme, wie unter anderem das Sortieren von Tupeln nach beliebigem Element, oder Javascript- und SQL-Syntaxfehler erwies sich <http://stackoverflow.com/> als nützlich.

Für Konzepte zum Thema Datenbankaufbau war diese Seite als sehr, sehr hilfreich:

<http://en.tekstenuitleg.net/articles/software/database-design-tutorial/intro.html>

Problemlösungen zu Javascript, Css und Html habe ich auch auf <http://www.w3schools.com/> gefunden.

Fazit

Wie bereits am Anfang der Reflexion geschrieben, ich bin zufrieden. Der Kurs hat mir die Möglichkeit gegeben das Vokabellernspiel umzusetzen. Außerdem konnte ich den Umgang mit git und Bottle üben und etwas über SQL und Datenbanken lernen.

Was die zu ambitionierte Must-Have-Feature-Liste betrifft, vielleicht gelingt es mir die fehlenden Features später einzubauen.

Verwendete Literatur:

Gamification:

Gabe Zichermann & Christopher Cunningham, Gamification by Design- Implementing Game Mechanics in Web and Mobile Apps, 2011, O'Reilly Beijing, Cambridge, Farnham, Köln, Sebastopol, Tokyo

SQL:

Bartosch O. & Throll M. Einstieg in SQL – verstehen einsetzen nachschlagen, Galileo Computing, 2011, Bonn

Javascript:

Morgen Nick, Javascript kinderleicht!- Einfach Programmieren lernen mit der Sprache des Web, 2015, dpunkt.verlag, Heidelberg