Habit Tracker

From app concept to open source program

Further training project of

https://www.iu-akademie.de

Project by

Björn Leue, *25.11.1985
iu akademie e-mail: bjoern.leue@iu-academy.org
personal e-mail: webmaster@wildsite.de

Last editing: 19.12.22

Contentlist

Introduction

Für die Weiterbildung der IU-Academy ("Python and SQL programming") habe ich im dritten Modul den Auftrag erhalten, einen Gewohnheitstracker zu kreieren.
Ich sollte mich dabei dabei orientieren wie ich es Kollegen erklären wollen würde.

Im folgenden finden Sie alle Gedanken wie ich sie mir für Version 0.1 gemacht habe.

Preview

Ich sah mir die Aufgabenstellung an und analysierte die nötigen ToDo's.
Weil ich bei diesem Kurs jedoch ein wenig vor eilte, entschied ich mich eine erweiterte Variante mit minimalen User-interface zu planen.

Aufgaben:
Gewohnheiten sollen mit Aufgabenspezifikation erstellt werden können (mit Zeitraum der Ausführung).
Zusätzliches "erledigt" Feld
Mindestens 2 Zeiträume (Täglich, Wöchentlich)
2 Wochen täglich einstellbar
Wenn eine Gewohnheit nicht eingehalten wird, soll darauf hingewiesen werden
Gewohnheiten sollen augewertet werden können.

- Längste Serie
- Aktuelle tägliche gewohnheiten
- Letzten Monat am schwierigsten

Akzeptanzkriterien:

- Python 3.7
- Installations und ausführungsanweisung (ReadMe.md, docstrings,..)
- Mindestens eine Objektorientierte Klasse
- Mindestens 2 Perioden (Wöchentlich, Monatlich)
- 5 vordefinierte Gewohnheiten mit transparenter usabillity
- Speicherung des start/stop der tätigkeit (datetime)
- 4 Wochen speicherung der Daten, später für *"Test Fixture"* verwendet
- Oft eingegebene Daten sollen zwischengespeichert werden (sqlite, cookies, json....)
- Analysemodul zur darstellung von daten.. *in der console?*
- API zum erstellen, löschen und anzeigen... nun doch vielleicht mit anzeige..
- pytest oder unittest können/sollten benutzt werden
- Gewohnheiten sollen als erledigt markierbar sein

App concept

Für die Umsetzung dachte ich mir, den Vorgaben entsprechend, ein Python Backend.
Dieses kann über über annahme von Parametern in Key=Value format angesteuert werden.

Wie man die Funktionen aufruft:

Windowstaste, „cmd“ eintippen, Enter drücken und folgenden Text eingeben:

zum Beispiel ist „PfadZurDatei“: C:\xampp\htdocs\HabitTracker\Python

cd PfadZurDatei
python dateiname.py action=TestEverything automatic_tests=True show_actions=True
python dateiname.py action=SignupUser user_name=YOURNAME user_password=YOURPASS
python dateiname.py action=AddHabit name=Do³some³sport!!! description='' timespan=dayly
python dateiname.py action=ShowAll

Dafür programmiere ich eine main function, die gegebene Parameter aufnimmt und in eine settings.json einfügt.

Von dieser settings.json werden alle gesetzten parameter für die weitere Verarbeituung genommen.

Im nächsten Schritt wird durch den „action“ Parameter die durchzuführende Funktion gewählt, ausgeführt und wenn benötigt Daten wieder gegeben (Zeilenweise oder per json).

Diese werden bei Benutzung der Webvariante und auswertbaren Daten über das HTML frontend visuell ausgewertet.

Aufbau

HabitTracker.py	Main program, sets Attributes to json and start an action
actions.py	collection of functions for running the program
analyse.py	collection of functions for analysing the data
sqlite.py	collection of functions for database connection
test_project.py	class with collection of functions for testing the whole project

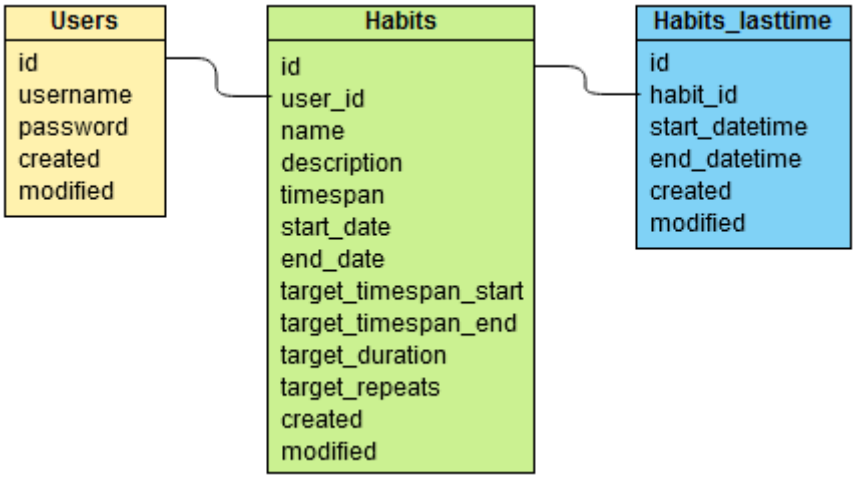
Speicherung der Daten

Für die Speicherung der Daten benutze ich SQLite, weil bei lokaler Benutzung auch eine lokale Datenbank recht logisch erscheint.

Eine zusätzliche implementation von Mysql wäre jedoch möglich.

Den Grundsätzlichen Aufbau plane ich im ersten Schritt nicht sehr Komplex.
Für die Funktionalität sollte ein relationaler Aufbau wie im folgenden reichen:

Ein User hat mehrere Gewohnheiten.
Eine Gewohnheit hat mehrere Aktionszeiträume.



--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---

Für weiteren Ausbau wäre eine Spielcharakterliche Erweiterung möglich.
Dies erhöht den Point to return auf Apps wie diese normalerweise erheblich.

Ein User hat mehrere Gewohnheiten, bei Einhaltung gibt es Punkte.
Ab einer gewissen Punkteanzahl in den Bereichen dayly, weekly, .. gibt es Pokale zum sammeln.

users → user_gamepoints → gamepoints

DBMS-UML

Users

id	ID eines users (autoincrement)
name	Benutzername des benutzers
password	Passwort des benutzers
created	Datetime an dem der Benutzer erstellt wurde
modified	Timestamp der Veränderung

Habits

id	ID der Gewohnheit
user_id	ID des Users der die Gewohnheit erstellt hat
name	Name der Gewohnheit
description	Beschreibung oder Subtext der Gewohnheit
timespan	Zeitspanne der Gewohnheit (daily, weekly, monthly, yearly)
target_datetime_start	Geplante Uhrzeit zum start
target_datetime_end	Geplante maximale Uhrzeit zum beenden
target_duration	Geplante Dauer der Gewohnheit
target_repeats	Geplante Wiederholungen der Gewohnheit
done	Boolscher Wert zur darstellung ob Habit erledigt ist
created	Datetime an dem die Gewohnheit erstellt wurde
modified	Timestamp der Veränderung

Habits_lasttime

id	ID der gemachten Aktivität
habit_id	ID der zugehörigen Gewohnheit
start_datetime	Reelle Uhrzeit des starts
end_datetime	Reellle Uhrzeit der beendigung
created	Datetime wann der Eintrag erstellt wurde
modified	Timestamp der Veränderung

Mögliche Spielcharakterliche Erweiterungen (**ab Version 2**)

User_gamepoints — Tabelle zum speichern von Punkten die man bekam

id
user_id
gamepoints_id
created

Gamepoints — Tabelle mit Punkten als Belohnung für eingehaltene Serien

id
name
description points

Winnings — Tabelle mit Gewinnen bei genug Punkten

id
points
image_url
description

HTML user-interface

Die Variablennamen müssen für das Python Script alle korrekt eingegeben werden.
Weil ich dies ein wenig unkonfortabel finde, werde ich ein Webfrontend erstellen, bei dem die Daten in inputfelder eingetragen werden können.

Über Javascript werden die Felder aufgenommen und an ein PHP Script gesendet, dass das Python Backendscript mit Parametern aufruft.

Von der HTML Benutzeroberfläche aus werden (für usabillity der übertragung zur Kommandozeile) die komma's durch --komma-- ersetzt und leerzeichen mit dem mathematischen kubik (³) Zeichen. Diese Veränderung des gegebenen Textes wird im Python Backend wieder zurück gesetzt.

Html / Javascript ↔ PHP Script ↔ Python Programm ↔ DBMS (MariaDB or SQLite3)

HTML Seiten

index.html	Eingangspunkt, Weiterleitung zu „login.html“
login.html	Login / Signup Formular, Datenauswertung durch Python
start.html	HTML Grundstruktur für Templates, geladen durch Auswahl der Funktion
Templates	HTML snippets, mit benötigten Eingabefeldern zum ansteuern der Python Funktionen