AI-BUDDY
GRUPPE 3 – FLUGBUCHUNG FLASK APP

TECHNISCHE DOKUMENTATION DER APP - erstellt von ChatGPT 3.5

Erörterung zur Korrektheit – siehe AI-Buddy Essay

Studienfach Datenbankentwicklung
Maral Chavoshi, Matthias Piniel, Maximilian Reutterer, Lukas Sattler, Jakob Weinzierl

ChatGPT 3.5

Mein Beitrag zur Arbeit bestand darin, die vorhandenen Codefragmente zu analysieren, zu kommentieren und zu dokumentieren. Dazu gehörte die Erstellung von Dokumentationen für die Python-Dateien app.py und db_access.py, sowie die Beschreibung der Funktionsweise der Website und der Interaktionen zwischen Frontend und Backend. Meine Kommentare und Erläuterungen sollten dazu beitragen, dass andere Entwickler den Code leichter verstehen und warten können.

Die Vorteile der Verwendung von AI-basierten Assistenten wie mir für solche Aufgaben sind:

- Effizienz:
- Konsistenz
- Skalierbarkeit

Die Nachteile und Limitierungen könnten sein:

- Verständnisgrenzen:
- Abhängigkeit von der Qualität des Codes:
- Fehlende menschliche Kreativität:

Insgesamt trage ich dazu bei, die Entwicklungseffizienz zu steigern, indem ich Routineaufgaben wie die Code-Dokumentation automatisiere, aber es ist wichtig zu beachten, dass menschliche Überprüfung und Ergänzung immer noch notwendig sind, um die Qualität sicherzustellen.

Allgemein:

- Was macht die Website?

  Die Website ist eine Buchungsplattform für Flugreisen, die es Benutzern ermöglicht, Flüge zu suchen, zu buchen und zu verwalten. Benutzer können sich anmelden, Flüge suchen, Flugdetails anzeigen, Flüge buchen, Buchungshistorien anzeigen und vieles mehr. Ebenso gibt es für Mitarbeiter eigene Funktionen in einem separaten Bereich.

- Wie funktioniert Flask?

  Flask ist ein Web-Framework für Python, das die Entwicklung von Webanwendungen erleichtert. Es verwendet den WSGI-Toolkit und Jinja2 für Templates, um dynamische

Webseiten zu generieren. Flask ermöglicht es, verschiedene Routen für verschiedene URLs zu definieren und Funktionen zuzuweisen, die auf Anfragen reagieren. Diese Funktionen verarbeiten die Anfragen, führen Aktionen durch und generieren die entsprechenden Antworten.

- Beispiele für POST und GET Requests:

  GET Request: Wenn ein Benutzer die Suchfunktion verwendet, um verfügbare Flüge abzurufen, sendet das Frontend einen GET-Request an die Flask-Route '/flights_search'. Diese Anfrage enthält die Parameter 'from' und 'to', die die Abflug- und Zielorte des Benutzers angeben. Flask verarbeitet diese Anfrage, ruft die entsprechenden Flüge aus der Datenbank ab und sendet die Ergebnisse an das Frontend zurück.

  POST Request: Wenn ein Benutzer einen Flug bucht, sendet das Frontend einen POST-Request an die Flask-Route '/booking/<flightNo>'. Diese Anfrage enthält die Flugnummer des ausgewählten Fluges. Flask verarbeitet die Buchung, fügt den Flug zur Buchungshistorie des Benutzers hinzu und sendet eine Bestätigung an das Frontend zurück.

- Übergabe an das HTML-Frontend:

  Die Daten, die vom Backend generiert werden, werden über Jinja2-Templates an das HTML-Frontend übergeben. Flask verwendet Renderfunktionen, um die HTML-Templates mit dynamischen Daten zu füllen und die fertigen HTML-Seiten an den Browser des Benutzers zurückzugeben. Durch die Verwendung von Templates können Variablen und Daten direkt in HTML eingebettet und angezeigt werden.

Nachfolgend finden Sie eine Dokumentation für die bereitgestellten Python-Klassen und Funktionen:

Registration-Klasse

Die Registration-Klasse ermöglicht es Benutzern, sich zu registrieren.

Methoden:

`register(data, phone)`: Diese Methode ermöglicht die Registrierung eines Benutzers mit den bereitgestellten Daten und Telefonnummer.

Authentification-Klasse

Die Authentification-Klasse bietet Funktionen zur Authentifizierung von Benutzern.

Methoden:

`authentification()`: Diese Methode authentifiziert einen Benutzer anhand seiner E-Mail-Adresse und seines Passworts.

`errormessage()`: Diese Methode gibt eine Fehlermeldung zurück, wenn die Anmeldeinformationen ungültig sind.

USER-Klasse

Die USER-Klasse ermöglicht den Zugriff auf Benutzerinformationen und deren Berechtigungen.

Methoden:

`user(arg, par)`: Diese Methode gibt Informationen über einen Benutzer basierend auf dem angegebenen Argument zurück.

`perm(svn)`: Diese Methode gibt die Berechtigungsebene eines Benutzers basierend auf seiner SVN zurück.

`update_user(svn, data)`: Diese Methode aktualisiert die Benutzerinformationen in der Datenbank.

Transaction-Klasse

Die Transaction-Klasse bietet Funktionen zur Verifizierung von Transaktionen.

Methoden:

`verify(bankaccount, cvv)`: Diese Methode überprüft eine Transaktion anhand der bereitgestellten Bankkontonummer und CVV.

BOOKING-Klasse

Die BOOKING-Klasse ermöglicht die Buchung von Flügen.

Methoden:

`__init__(flights_in_cart, passNo)`: Der Konstruktor initialisiert die Buchung mit den Flügen im Warenkorb und der Passagier-ID.

`__call__()`: Diese Methode führt die Buchungstransaktion durch.

CANCEL_BOOKING-Klasse

Die CANCEL_BOOKING-Klasse ermöglicht das Stornieren von Buchungen.

Methoden:

`__init__(id)`: Der Konstruktor initialisiert die Stornierung mit der Buchungs-ID.

`__call__()`: Diese Methode führt die Stornierung der Buchung durch.

BLACKBOX_CHECKOUT-Klasse

Die BLACKBOX_CHECKOUT-Klasse ermöglicht das Ausleihen und Zurückgeben von Flugschreibern.

Methoden:

`available_blackboxes()`: Diese Methode gibt eine Liste der verfügbaren Flugschreiber zurück.

`blackbox_ids()`: Diese Methode gibt eine Liste der Flugschreiber-IDs zurück.

`checkout(svn, blackbox_id)`: Diese Methode ermöglicht das Ausleihen eines Flugschreibers anhand der Benutzer-SVN und der Flugschreiber-ID.

`user_blackboxes(svn)`: Diese Methode gibt eine Liste der Flugschreiber zurück, die von einem bestimmten Benutzer ausgeliehen wurden.

return_blackbox(id): Diese Methode ermöglicht das Zurückgeben eines Flugschreibers anhand seiner ID.

Ticket-Klasse

Die Ticket-Klasse erstellt ein Boarding-Ticket im PDF-Format.

Methoden:

`content(data)`: Diese Methode erstellt den Inhalt des Boarding-Tickets basierend auf den bereitgestellten Daten.

SS24

Db_acces.py

DB_Access-Klasse

Die DB_Access-Klasse bietet Methoden zum Ausführen von Datenbankabfragen und -operationen.

Methoden:

`executeFetchOne(sql, arg=())`: Diese Methode führt eine SQL-Abfrage aus und gibt das erste Ergebnistupel zurück.

> **`sql`**: Die SQL-Abfrage, die ausgeführt werden soll.
> **`arg`**: Die Argumente für die SQL-Abfrage (Standardwert ist eine leere Tupel).

`executeFetchSingle(sql, arg=())`: Diese Methode führt eine SQL-Abfrage aus und gibt einen einzelnen Wert zurück.

> **`sql`**: Die SQL-Abfrage, die ausgeführt werden soll.
> **`arg`**: Die Argumente für die SQL-Abfrage (Standardwert ist eine leere Tupel).

`executeFetchAll(sql, arg=())`: Diese Methode führt eine SQL-Abfrage aus und gibt alle Ergebnistupel zurück.

> **`sql`**: Die SQL-Abfrage, die ausgeführt werden soll.
> **`arg`**: Die Argumente für die SQL-Abfrage (Standardwert ist eine leere Tupel).

`executeInsert(sql, arg=())`: Diese Methode führt eine SQL-Abfrage zum Einfügen von Daten in die Datenbank aus.

> **`sql`**: Die SQL-Abfrage zum Einfügen von Daten.
> **`arg`**: Die Argumente für die SQL-Abfrage (Standardwert ist eine leere Tupel).

Alle Methoden nutzen eine SQLite-Datenbankverbindung, die auf die Datenbankdatei "instance/db.db" zugreift.

App.py

Flask-App

Die Flask-App stellt eine Webanwendung bereit, die verschiedene Routen für Benutzer und Mitarbeiter bereitstellt, um auf verschiedene Funktionen zuzugreifen.

Hauptfunktionen:

- Benutzeranmeldung und Registrierung: Die App ermöglicht es Benutzern, sich anzumelden und zu registrieren.
- Benutzerprofil: Benutzer können ihr Profil anzeigen und bearbeiten.
- Flugsuche und Buchung: Benutzer können nach verfügbaren Flügen suchen und Buchungen vornehmen.
- Transaktionen: Benutzer können Flugbuchungen durchführen und Transaktionen verwalten.
- Buchungshistorie: Benutzer können ihre vergangenen Flugbuchungen anzeigen.
- Ticketdruck: Benutzer können ihre Tickets herunterladen und drucken.
- Backoffice-Funktionen: Mitarbeiter haben Zugriff auf Backoffice-Funktionen wie die Verwaltung von Flugzeugen, Flugbuchungen und Flugschreibern.

Routen:

`/login`: Route für die Benutzeranmeldung.

`/logout`: Route zum Abmelden eines Benutzers.

`/register`: Route für die Benutzerregistrierung.

`/user`: Route für das Benutzerprofil.

`/`: Startseite der Anwendung.

`/home`: Startseite für angemeldete Benutzer.

`/available_flights`: Route zur Anzeige verfügbarer Flüge.

`/booking/<flightNo>`: Route zur Buchung eines bestimmten Flugs.

`/flights_search`: Route für die Flugsuche.

`/shop`: Route für das Anzeigen des Warenkorbs und den Checkout.

`/transaction`: Route zur Abwicklung von Transaktionen.

`/history`: Route zur Anzeige der Buchungshistorie.

`/cancel_booking/<id>`: Route zum Stornieren einer Buchung.

`/history/<id>`: Route für den Ticketdruck.

`/backoffice`: Route für das Backoffice-Dashboard.

`/backoffice/blackbox/return`: Route für das Zurückgeben von Flugschreibern.

`/backoffice/blackbox/checkout`: Route für das Ausleihen von Flugschreibern.

`/backoffice/flights/`: Route für die Anzeige von Flügen aus Mitarbeiterperspektive.

Die App verwendet Sessions, um die Benutzerauthentifizierung und seitenübergreifende Informationen zu verwalten, und SQLAlchemy für die Datenbankinteraktion.