FLY SCAN: progetto APL 2023/24 studenti **Antonio Inveninato – Angelo Cocuzza**

ARCHITETTURA DELL'APPLICAZIONE

Il seguente progetto vuole realizzare una piattaforma dedicata al tracciamento dei prezzi dei voli, fornendo agli utenti la possibilità di monitorare e ricevere notifiche quando i prezzi dei voli per le loro destinazioni preferite hanno un ribasso.
Il sistema è progettato come un Client-Server che comunicano tra loro mediante REST API, con C# viene implementato lato client e lato server abbiamo utilizzato Go e Python.

Python

Auth

Questo modulo implementa funzionalità di autenticazione e gestione utenti.

Le varie funzioni di questo modulo sono:

1. is_valid_password(password):
    - Verifica se una password è valida (lunghezza minima di 8 caratteri).
2. logout(token):
    - Gestisce il logout dell'utente revocando il token fornito.
    - Inserisce il token nella tabella token_revoked del database.
3. login(username, password):
    - Gestisce il processo di login.
    - Verifica le credenziali dell'utente nel database.
    - Se le credenziali sono valide, crea e restituisce un nuovo token JWT.
4. register(username, password):
    - Gestisce la registrazione di un nuovo utente.
    - Verifica che la password sia valida e che l'username non sia già in uso.
    - Cripta la password usando bcrypt prima di salvarla nel database.
    - Crea e restituisce un nuovo token JWT per l'utente registrato.
5. createToken(username):
    - Crea un token JWT per l'utente specificato.

- Il token include il nome utente e un tempo di scadenza (2 ore dal momento della creazione).

Inoltre, il modulo configura la connessione al database utilizzando SQLAlchemy e definisce due tabelle:

- users: per memorizzare le informazioni degli utenti (id, username, password criptata).
- token_revoked: per tenere traccia dei token revocati durante il logout.

Graphs

Questo modulo si occupa di generare un grafico basato sui dati delle ricerche di viaggio memorizzati nel database.

Il modulo si articola in:

1. Configurazione del Database:
    - Utilizza SQLAlchemy per connettersi a un database MySQL.
    - Le credenziali del database sono recuperate dalle variabili d'ambiente per sicurezza.
2. Definizione della Tabella:
    - Viene definita una tabella chiamata 'research' con tre colonne:
        - 'id': chiave primaria
        - 'city_from': città di partenza
        - 'city_to': città di destinazione
3. Funzione graph(): Questa è la funzione principale del modulo.
    - Si connette al database e recupera tutti i dati dalla tabella 'research'.
    - Crea una lista di tuple (città_partenza, città_arrivo) per ogni ricerca.
    - Utilizza Counter per contare le occorrenze di ogni tragitto unico.
    - Utilizza matplotlib per creare un grafico a barre orizzontali.
    - L'asse y mostra le rotte (es. "Milano -> Roma").
    - L'asse x mostra il numero di ricerche per ogni rotta.
    - Il grafico è intitolato "Mete più Ricercate (Andata e Ritorno)".
    - Il grafico viene salvato in un buffer di memoria (BytesIO) come immagine PNG.
    - Questo permette di utilizzare l'immagine del grafico senza salvarla su disco.
    - La funzione restituisce il buffer contenente l'immagine del grafico.

Notifier

Questo modulo e si connette a un database MySQL e utilizza API di terze parti (KIWI) per cercare offerte di volo in base alle preferenze salvate degli utenti. Inoltre, invia notifiche via email agli utenti quando vengono trovate nuove offerte.

1. Funzione get_iata(city)
    - Questa funzione interagisce con l'API Tequila di Kiwi.com per ottenere il codice IATA della città specificata.
    - Effettua una richiesta all'API e, se la risposta è positiva, estrae il codice IATA dalla risposta JSON. Questo codice IATA è essenziale per le ricerche di voli.
2. Funzione get_flights()
    - Questa funzione effettua una richiesta all'API di Kiwi.com per ottenere i voli disponibili tra due città specificate (identificate dai codici IATA) nelle date fornite.
    - I parametri della richiesta includono dettagli come date di partenza e ritorno, prezzo massimo, e il tipo di veicolo (ad es. aereo).
    - La funzione ritorna i dati dei voli disponibili sotto forma di JSON.
3. Funzione send_notify()
    - Questa funzione è il cuore del modulo. Si connette al database e recupera tutte le tratte preferite salvate dagli utenti.
    - Per ogni tratta, chiama get_iata() per ottenere i codici IATA delle città di partenza e arrivo, quindi chiama get_flights() per trovare i voli disponibili in quelle date.
    - Se vengono trovate nuove offerte, i dati del volo vengono passati alla funzione check_flights().
4. Funzione calc_date(d)
    - Questa funzione aggiunge due giorni alla data specificata per calcolare una nuova data di ritorno.
    - La nuova data viene ritornata in formato stringa.
5. Funzione formatta_date(d)
    - Questa funzione converte una data dal formato ISO (utilizzato nelle API) a un formato più leggibile.
6. Funzione check_flights()
    - Questa funzione verifica se il volo trovato dall'API è un'offerta migliore rispetto a quella salvata nel database.
    - Se il nuovo prezzo è inferiore a quello salvato, viene costruito un messaggio email con i dettagli del volo e viene chiamata la funzione send_notification_email() per inviarlo all'utente.
7. Funzione send_notification_email()
    - Questa funzione invia un'email di notifica all'utente usando un server SMTP.

- L'email contiene i dettagli del volo e viene inviata se c'è una nuova offerta disponibile.

App

App.py è il cuore del server python, viene utilizzato il framework Flask di Python. Flask è progettato per essere semplice e flessibile, permettendo agli sviluppatori di creare rapidamente applicazioni web o API.

App offre diverse funzionalità legate alla gestione di utenti (registrazione, login, logout), alla generazione di grafici e all'invio periodico di notifiche via e-mail.

1. Funzione send_email()
    - Questa funzione gira in un ciclo infinito, chiamando periodicamente (ogni 600 secondi, cioè 10 minuti) la funzione send_notify() del modulo notifier.
    - Questa funzione viene eseguita in un thread separato per non bloccare l'esecuzione principale dell'app Flask.
2. /graph (metodo POST):
    - Questa rotta chiama la funzione graph() dal modulo graphs, che genera un grafico (probabilmente in base ai dati forniti) e lo restituisce come un'immagine PNG.
    - La risposta viene inviata con il tipo MIME image/png.
3. /login (metodo POST):
    - Questa rotta gestisce il login degli utenti. Riceve una richiesta JSON con username e password, quindi chiama la funzione login() del modulo auth per autenticare l'utente.
    - La risposta è il risultato della funzione auth.login(), che probabilmente restituisce un token di autenticazione o un messaggio di errore.
4. /logout (metodo GET):
    - Questa rotta gestisce il logout degli utenti. Il token di autenticazione è passato nell'intestazione Authorization.
    - La rotta verifica che il token sia presente e nel formato corretto (Bearer Token). Se tutto è corretto, chiama auth.logout() per effettuare il logout dell'utente.
5. /register (metodo POST):
    - Questa rotta gestisce la registrazione degli utenti. Riceve una richiesta JSON con username e password, quindi chiama la funzione register() del modulo auth.
    - La risposta è il risultato della funzione auth.register(), che probabilmente restituisce un messaggio di successo o errore.
6. Threading
    - Viene creato un thread separato per eseguire la funzione send_email() in background. Questo thread è impostato come "daemon", il che significa che terminerà automaticamente quando l'applicazione principale si ferma.
7. Avvio del server

- Infine, l'applicazione Flask viene avviata con app.run(), che permette di far girare il server web incorporato di Flask, rendendo l'app disponibile per ricevere richieste HTTP.

Go

Database Spedizioni

Il package search gestisce la ricerca di voli e l'analisi delle ricerche più popolari. Questo modulo utilizza un'API esterna per ottenere i codici IATA delle città e per effettuare ricerche di voli tra due destinazioni. Inoltre, memorizza le ricerche effettuate in un database e consente di visualizzare le ricerche più popolari.

1. Importazioni
    - database: Modulo interno per la gestione del database.
    - encoding/json, fmt, log, net/http, net/url, os, sort, strings: Librerie standard di Go utilizzate per varie operazioni, come gestione delle richieste HTTP, manipolazione di stringhe, gestione delle variabili d'ambiente e operazioni di logging.
    - github.com/joho/godotenv: Una libreria per caricare le variabili d'ambiente da un file .env.
2. Funzione Init()
    - Carica le variabili d'ambiente dal file .env usando la libreria godotenv.
    - Recupera la chiave API da queste variabili e la assegna alla variabile globale apiKey.
3. Struttura SearchRequest
    - Definisce i campi JSON che vengono utilizzati per le richieste di ricerca di voli. Include informazioni come la città di partenza (CityFrom), la città di destinazione (CityTo), le date di viaggio (DataFrom, DataTo, ReturnFrom, ReturnTo), e i limiti di prezzo (PriceMin, PriceMax).
4. Funzione SearchHandler()
    - Riceve un'istanza di SearchRequest come input.
    - Inizializza l'ambiente chiamando Init().
    - Converte le città di partenza e destinazione in codici IATA utilizzando l'API esterna attraverso la funzione getIata().
    - Esegue una ricerca di voli utilizzando l'API esterna attraverso la funzione getFlights().
    - Registra le ricerche nel database.
    - Restituisce i dati dei voli trovati o un errore se qualcosa va storto.
5. Funzione getIata()
    - Effettua una richiesta HTTP GET all'API esterna per ottenere il codice IATA di una città.
    - Elabora la risposta JSON per estrarre il codice IATA e lo restituisce.
6. Funzione getFlights()

- Effettua una ricerca di voli utilizzando l'API esterna.
- Accetta come parametri i codici IATA delle città di partenza e destinazione, le date di viaggio, e i limiti di prezzo.
- Elabora la risposta JSON e restituisce i risultati della ricerca di voli.

7. Funzione ShowPopular()
   - Recupera dal database le ricerche di voli effettuate in precedenza.
   - Conta quante volte ciascuna combinazione di città è stata ricercata.
   - Restituisce un elenco delle ricerche più popolari, ordinato per frequenza e limitato alle prime quattro.
8. Funzione parseKey()
   - Utilizzata internamente da ShowPopular() per suddividere una stringa che rappresenta una coppia di città (city_from|city_to) in due città distinte.
9. Interazione con il Database
   - Il modulo interagisce con un database per salvare e recuperare le ricerche di voli. Utilizza query SQL per creare tabelle, inserire dati e recuperare risultati.
   - Le ricerche sono memorizzate in una tabella chiamata research, che conserva le città di partenza e di destinazione.
10. Utilizzo dell'API Esterna
    - L'API utilizzata per ottenere i codici IATA e cercare voli è Tequila API di Kiwi.com. Questa API richiede un'API key che viene caricata dalle variabili d'ambiente.

Check

Check è un package che fornisce funzionalità per gestire e verificare i token JWT. Il modulo include funzionalità per controllare se un token è valido, se è stato revocato, e se è scaduto.

1. Struttura Token
   - Definisce la struttura del payload di un token JWT.
   - Contiene:
     - Username: Nome utente associato al token.
     - ExpirationTime: Timestamp Unix che indica quando il token scade.
     - StandardClaims: Claim standard definiti dalla libreria JWT, che includono informazioni come l'emittente, il soggetto, ecc.
2. Funzione Init()
   - Carica le variabili d'ambiente dal file .env usando la libreria godotenv.
   - Inizializza la variabile SECRET_KEY con il valore della chiave segreta specificata nel file .env.
3. Funzione CheckToken()
   - Questa funzione verifica se un token JWT è valido, non revocato, e non scaduto.
   - Passaggi principali:
     - Inizializzazione: Carica la chiave segreta usando Init() e apre una connessione al database.

        - Controllo Token Revocati:
        - Controlla se il token è presente nella tabella token_revoked del database.
        - Se il token è presente, significa che è stato revocato e la funzione restituisce un errore.
        - Parsing e Verifica del Token:
        - Il token viene parsato usando la libreria jwt-go.
        - Viene verificato se il metodo di firma è valido e se il token è stato firmato correttamente con la chiave segreta.
        - Verifica Scadenza:
        - Se il token è valido, la funzione controlla se è scaduto confrontando il timestamp attuale con il campo ExpirationTime del token.
        - Se il token è scaduto, viene restituito un errore.
        - Restituzione dell'Username:
        - Se tutte le verifiche passano, la funzione restituisce l'username associato al token.
4. Interazione con il Database
    - La funzione CheckToken() interagisce con il database per verificare se un token è stato revocato.
    - Viene utilizzata una tabella token_revoked per memorizzare i token che sono stati revocati.

Favourites

Il package favourites gestisce le operazioni relative ai preferiti degli utenti, come l'aggiunta, la selezione e la rimozione di viaggi preferiti in un'applicazione. Questo package interagisce con un database per memorizzare e recuperare i dati relativi ai viaggi che gli utenti hanno contrassegnato come preferiti.

1. Importazioni
    - database: Un package interno per gestire la connessione al database.
    - _ "github.com/go-sql-driver/mysql": Driver MySQL per la connessione al database. La notazione _ indica che il package viene importato per il suo effetto collaterale di registrare il driver MySQL con il pacchetto database/sql.
    - log: Pacchetto per la gestione dei log.
2. Struttura FavouritesRequest
    - Definisce la struttura per rappresentare una richiesta di gestione dei preferiti.
    - Contiene:
        - CityFrom: La città di partenza.
        - CityTo: La città di destinazione.
        - DataFrom: La data di partenza.
        - ReturnData: La data di ritorno.
        - Price: Il prezzo del viaggio.
        - User: L'utente associato al preferito.
3. Funzione FavouritesHandler()
    - Gestisce l'aggiunta di un viaggio alla lista dei preferiti di un utente.
    - Passaggi principali:
        - Inizializzazione della Connessione al Database: Connette al database usando la funzione RunDB del package database.
        - Creazione della Tabella favourites: Se non esiste già, viene creata una tabella favourites nel database per memorizzare i preferiti.

        - Verifica Esistenza Preferito: Controlla se un viaggio specifico è già presente tra i preferiti dell'utente. Se esiste, restituisce un messaggio indicante che l'elemento è già nei preferiti.
        - Aggiunta ai Preferiti: Se il viaggio non è già presente, lo inserisce nella tabella favourites.
        - Chiusura Connessione al Database: Chiude la connessione al database.
4. Funzione FavouritesSelect()
    - Recupera la lista dei viaggi preferiti di un utente.
    - Passaggi principali:
        - Inizializzazione della Connessione al Database: Connette al database usando la funzione RunDB.
        - Esecuzione Query: Esegue una query per selezionare tutti i viaggi preferiti di un determinato utente.
        - Popolamento della Lista dei Preferiti: Scorre i risultati della query e li aggiunge a una lista di strutture FavouritesRequest.
        - Chiusura Connessione al Database: Chiude la connessione al database.
        - Restituzione dei Preferiti: Restituisce la lista dei viaggi preferiti dell'utente.
5. Funzione FavouritesDelete()
    - Gestisce la rimozione di un viaggio dalla lista dei preferiti di un utente.
    - Passaggi principali:
        - Inizializzazione della Connessione al Database: Connette al database usando la funzione RunDB.
        - Esecuzione Query: Esegue una query per eliminare un determinato viaggio dalla tabella favourites.
        - Chiusura Connessione al Database: Chiude la connessione al database.
        - Restituzione Messaggio di Conferma: Restituisce un messaggio che conferma l'eliminazione del preferito.

DB

Il package database è progettato per gestire la connessione a un database MySQL.

1. 2. Funzione Init()
    - Descrizione: Questa funzione è responsabile del caricamento delle variabili d'ambiente dal file .env, che contiene le configurazioni necessarie per connettersi al database.
    - Passaggi principali:
        - Caricamento delle Variabili d'Ambiente: Usa il package godotenv per caricare le variabili d'ambiente definite nel file .env.
        - Assegnazione delle Variabili Globali: Le variabili globali DB_HOST, DB_NAME, DB_PASSWORD, DB_USER vengono assegnate con i valori caricati dalle variabili d'ambiente.
        - Gestione degli Errori: Se c'è un errore durante il caricamento delle variabili d'ambiente, il programma termina e restituisce un messaggio di errore.
2. Funzione RunDB()
    - Descrizione: Questa funzione è responsabile di stabilire una connessione al database MySQL utilizzando i dettagli di configurazione caricati dalla funzione Init().
    - Passaggi principali:

- Inizializzazione delle Variabili di Connessione: Chiama Init() per assicurarsi che le variabili globali siano correttamente caricate.
- Creazione della Stringa di Connessione: Usa fmt.Sprintf per costruire la stringa di connessione al database MySQL, che include l'utente, la password, l'host e il nome del database.
- Apertura della Connessione: Usa sql.Open per aprire una connessione al database con la stringa di connessione creata.
- Verifica della Connessione: Esegue un Ping() sul database per verificare che la connessione sia stata stabilita correttamente.
- Gestione degli Errori: Se c'è un errore durante l'apertura della connessione o durante il ping al database, il programma termina e restituisce un messaggio di errore.
- Ritorno del Puntatore al Database: Se la connessione è stabilita con successo, restituisce un puntatore al database (*sql.DB) e l'eventuale errore (che in caso di successo sarà nil).

Main

Il package main permette di creare un server utilizzando il framework Echo. Questo è in grado di gestire grandi quantità di richieste con un overhead minimo, rendendolo ideale per lo sviluppo di API RESTful e servizi web.

Questo server espone diverse API REST per gestire operazioni legate a preferiti, ricerca di voli e token di autenticazione.

1. Importazioni
    - "APL_go/check": Importa il package che gestisce la verifica dei token JWT.
    - "APL_go/favourites": Importa il package che gestisce le operazioni sui preferiti.
    - "APL_go/search": Importa il package che gestisce la ricerca di voli e l'elenco dei voli popolari.
    - "encoding/json": Fornisce funzionalità per codificare e decodificare i dati in formato JSON.
    - "fmt": Utilizzato per la formattazione delle stringhe.
    - "github.com/labstack/echo/v4": Importa il framework Echo per la gestione delle API web in Go.
    - "net/http": Fornisce costanti e funzionalità per la gestione delle richieste HTTP.
2. Funzione main()
    - Descrizione: Questa funzione avvia un server HTTP che ascolta su una specifica porta e gestisce varie rotte per operazioni specifiche. È il punto d'ingresso dell'applicazione.
    - /favourites (POST):
        - Aggiunge un elemento ai preferiti di un utente.
        - Richiede l'autenticazione tramite un token JWT, che viene verificato tramite il package check.
        - Se il token è valido, l'elemento viene aggiunto ai preferiti tramite il package favourites.
    - /selectfav (POST):
        - Restituisce l'elenco dei preferiti di un utente.
        - Come per la rotta precedente, richiede un token JWT valido per l'autenticazione.
    - /deletefav (POST):

        - Elimina un elemento dai preferiti di un utente.
        - Richiede un token JWT valido per verificare l'identità dell'utente.
    - /search (POST):
        - Gestisce la ricerca di voli.
        - Richiede un token JWT valido.
        - Dopo la verifica del token, accetta una richiesta di ricerca voli (contenente parametri come città di partenza, destinazione, date, ecc.) e restituisce i risultati tramite il package search.
    - /show_pop (POST):
    - Restituisce l'elenco dei voli più popolari.
    - Non richiede autenticazione tramite token.
3. Verifica e Manipolazione del Token
    - Per ogni rotta che richiede autenticazione, viene estratto il token JWT dall'header della richiesta.
    - Il token viene verificato usando il package check per assicurarsi che sia valido e non sia stato revocato.
    - Se il token non è valido o manca, viene restituito un errore HTTP 401 (Unauthorized).
4. Avvio del Server
    - e.Start(":8080"): Avvia il server HTTP su localhost alla porta 8080.

C#

Form 1

In questo form viene data la possibilità all'utente di passare al form di Login o di Registrazione.

Form Login

In questo form viene data la possibilità all'utente di effettuare il login.

Form Registrazione

In questo form viene data la possibilità all'utente di effettuare la registrazione dell'account.

Form Home

Schermata di home della piattaforma, in cui viene data la possibilità all'utente di inserire i dati relativi al volo che vuole ricercare come:

- Intervallo date di partenza
- Intervallo date di ritorno
- Città di partenza
- Città di destinazione
- Prezzo min
- Prezzo max

Inoltre viene mostrato un grafico sulle varie rotte, e una lista contenente le 4 rotte più cercate con la possibilità di premere il pulsante per aggiungere le città nelle condizioni di ricerca.

Form Search

Qui viene creato un form che permette di visualizzare i risultati della ricerca. Inoltre è presente un pulsante che permette di inserire un volo tra i preferiti

Form Favourites

Questo form mostra i voli preferiti dell'utente con la possibilità di rimuoverli

Form Logout

Questo form consente all'utente di eseguire il logout