StarWarsApp Technical Documentation

- **Project structure**

StarWarsApp(Root)

```
|
|_____DataBase
|
|
|_____Docs
|
|
|_____lib
      |
      |_____dbFactory
      |
      |_____jsonController
```

- **Components**

- Inside the **StarWarsApp folder** you will find the Main.go and StarWarsApp.exe files.

StarWarsApp.exe: This is the compiled application file that contains the binary. If you move it be sure you move the DataBase folder and store them in the same directory.

Main.go: The main file of the application. Prepares the database to be created, check if the *character* table exists, if it doesn't calls the **dbFactory.CreateDatabase** function, otherwise checks the character name passed by argument and shows its information. After that,it runs the menu that consults the database to show the information needed, deppending on the selected option.

- Inside the **DataBase folder**: before running the application for the first time, thi folder will be empty (except fo an empty dummy file that allows git to upload "empty" folders) Once you run the application a **StarWarsDB.db** file will be created. It constains all the database info.

- Inside the **Docs folder** you can find all the documentation related to the aplication.

- Inside the **lib folder** you can find the custom packages: **dbFactory** and **jsonController**.

dbFactory.go: Inside the dbFactory folder. This package is responsible for the creation of the different tables in the database. When all of them are created, calls the function **JsonController.GetJson.**

JsonController.go: Inside the jsonController folder. This package is responsible for retrieving the Jsons information. It has all the structs where the information gets stored. First of all, it calls the function **GetJson** which calls the function **jsonIntoDB** with the initial url given and the struct that will contain all characters (*CharacterAPIResponse*), as

parameter and returns the same struct with the information needed. This function consults the url by *get* and retrieves the json information into the struct passed by parameter. In case this struct is of type *CharacterAPIResponse* iterates over the number of characters within the struct inserting the information into the database tables. For the character information fileds as: homeworld, species, films, starships or vehicles it calls the **jsonIntoDB** function passing the url and the struct of the specific information field so it returns the desired struct each time, filling the proper information within the character. After all the characters of the initial url have been introduced into the database, the function **jsonIntoDB** is called once again but this time passing the url of the next page of characters. This process iterates until there is no next page of characters.

- **Requirements**

This application runs on Windows operating system.

The whole application has been developed in Go version go1.13.7 windows/amd64

The database is sqlite3