Components:

1. DB Struct
2. Car Processor
3. Rent Processor

Responsibilities:

DB Struct:

1. Responsible to initialize DB information about cars
2. Responsible to provide correct data upon request

Car Processor:

1. Responsible to store/retrieve information about Cars
2. Responsible to provide correct data upon request
3. Responsible for communication with DB table cars

Rent Processor:

1. Responsible to store/retrieve information about Rents
2. Responsible to provide correct data upon request
3. Responsible for communication with DB table rents

API's:

1. **/api/cars?**
    a. **GET –** without URL values will return all cars
    b. **POST** – will add a new car to DB. Car object need to be provided in body with structure:

```
{
    "carCompanyName": "MyCar",
    "doors": 5,
    "bigLuggage": 1,
    "smallLuggage": 4,
    "adultPlaces": 7,
    "airConditioner": false,
    "minimumAge": 61,
    "price": 5,
    "carGroup": 4,
    "availableLocations":[
      "New York"
    ],
    "description": "Best choice for big family"
}
```

    c. **GET –** with URL values:
        i. Possible URL values are:
            1. location=* Will search for cars that are available in such citys. For example: location=Tel Aviv,Jerusalem , location=Tel Aviv

2. age=* Will search for cars that are available for such an age interval. For example: age=30-45 , age=30 (mean 30+)
3. car=2 Will search for cars in such car group ID. For example: car=2 , car=1
4. fromDate=*&toDate=* Will search for cars that are available in such a time interval. Both values must be provided. For example: fromDate=2022-01-14T15:13:30Z&toDate=2022-01-15T15:13:30Z , fromDate=2022-01-16T15:13:30Z&toDate=2022-01-18T15:13:30Z

2. **/api/cars/{CARID}** – CARID is number
   a. **GET –** will provide info about car related to such ID
   b. **PUT** – will update info about car related to such ID. Need to provide body with structure of

```
{
    "carCompanyName": "MyCar",
    "doors": 5,
    "bigLuggage": 1,
    "smallLuggage": 4,
    "adultPlaces": 7,
    "airConditioner": false,
    "minimumAge": 61,
    "price": 5,
    "carGroup": 4,
    "availableLocations":[
      "New York"
    ],
    "description": "Best choice for big family"
}
```

   c. **DELETE –** will remove car info related to such ID
3. **/api/rents**
   a. **GET** – will provided all registered rents
   b. **POST** – will add new rent info to DB. Location, age group, car group and dates will be checked against related DB info. Rent info object need to be provided in body with structure:

```
{
    "carID": 1,
    "fromDate": "2022-01-15T15:13:30Z",
    "toDate": "2022-01-15T15:15:30Z",
    "location": "Rishon LeZiyyon",
    "availableExtras": [
      "Free day"
    ],
    "discounts": [
```

```
        "5%"
      ],
      "ageGroup":"50",
      "carGroup":1
}
```

4. **/api/rents/{RENTID} – RENTID is number**
    a. **GET –** will provide info about rent related to such ID
    b. **DELETE –** will remove rent info related to such ID

Additional requests are located in requests.http file.
Backend can be started with go run cmd/main.go within the project folder. SQLite is used in project, in case of errors check dependencies
Tests are located in folders:

1. internal/server – server_test.go. Preferable to run whole test file, because rest is raised before first test, and closed in last test
2. internal/server/cars – cars_test.go