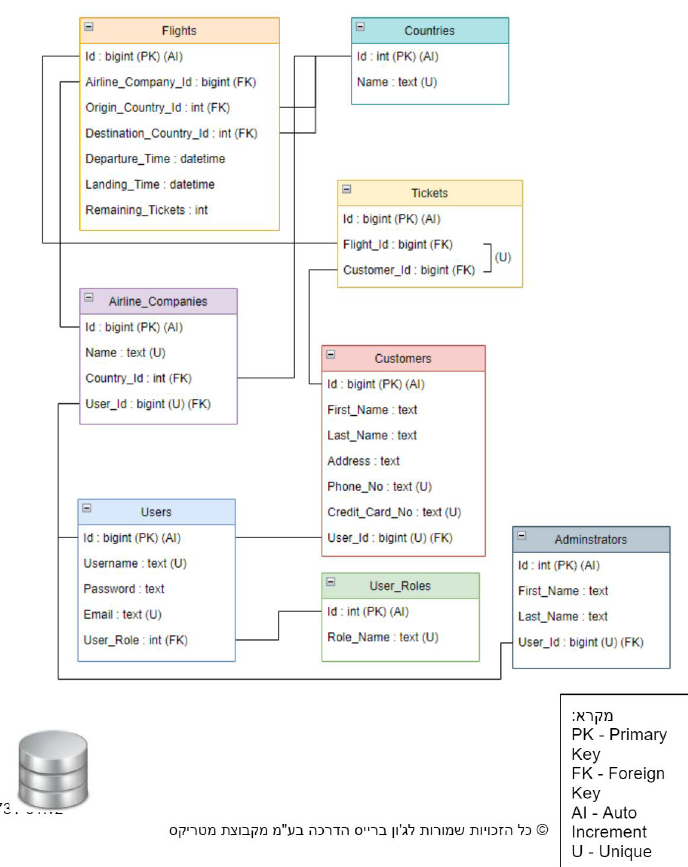
**Flights Project Structure**

Users:

* Administrator
* Airline
* Customer
* Anonymous (Unlogged)

Database Structure:

DB stored procedures:

* get\_airline\_by\_username(\_username text)
* get\_customer\_by\_username(\_username text)
* get\_user\_by\_username(\_username tex)
* get\_flights\_by\_parameters(\_origin\_country\_id int, \_destination\_country\_id int,\_date date)
* get\_flights\_by\_airline\_ID(\_airline\_id bigint)
* get\_arrival\_flights\_12hours(\_country\_id int)
* get\_departure\_flights\_12hours(\_country\_id int)
* get\_tickets\_by\_customer(\_customer\_id int)

Repository:

* .config- Connection String
* requirements.txt- שמות הספריות שהשתמשתם בהם
* Logger
  + Logger class-singleton. Activation of every function
* Flight
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
    - getFlightsByOriginCountryId(country\_id)
    - getFlightsByDestinationCountryId(country\_id)
    - getFlightsByDepartureDate(date)
    - getFlightsByLandingDate(date)
    - SP: get\_flights\_by\_parameters(\_origin\_country\_id int, \_destination\_country\_id int,\_date date)
    - SP: get\_flights\_by\_airline\_ID(\_airline\_id bigint)
    - SP: get\_arrival\_flights\_12hours(\_country\_id int)
    - SP: get\_departure\_flights\_12hours(\_country\_id int)
* AirlineCompany
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
    - getAirlinesByCountry(country\_id)
    - SP: get\_airline\_by\_username(\_username text)
* Customer
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
    - SP: get\_customer\_by\_username(\_username text**)**
* Administrator
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
* User
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
    - SP: get\_user\_by\_username(\_username tex)
* Country
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
* Ticket
  + class(dbmodel)
    - Get by id, Get All, Add, Update, Add All, Remove
    - getFlightsByCustomer(customer)
    - SP: get\_tickets\_by\_customer(\_customer\_id int)
* FacadeBase
  + FacadeBase (parent class)
    - get\_all\_flights()
    - get\_flight\_by\_id(id)
    - get\_flights\_by\_parameters(origin\_country\_id, destination\_country\_id, date)
    - get\_all\_airlines()
    - get\_airline\_by\_id(id)
    - get\_airline\_by\_parameters( … )
    - get\_all\_countries()
    - get\_country\_by\_id(id)
    - create\_new\_user ( user ) - for internal usage
* AnonymousFacade
* AnonymousFacade class(inherits façade base)
  + - login ( username, password )

After a successful login the function will return the ***correct facade*** [customer, airline or admin] and a ***login-token*** member inside (@getter)

Upon failure will return None or raise an error (i.e. WrongPasswordError)

* + - add\_customer ( … )
* CustomerFacade
* CustomerFacade class(inherits façade base)
  + update\_customer (customer)
  + add\_ticket (ticket)
  + remove\_ticket (ticket)
  + get\_my\_tickets ()
* AirlineFacade
* AirlineFacade class(inherits façade base)
  + get\_my\_flights ()
  + update\_airline (airline)
  + add\_flight (flight)
  + update\_flight (flight)
  + remove\_flight (flight)
* AdministratorFacade
* AdministratorFacade class(inherits façade base)
  + get\_all\_customers()
  + add\_airline (...)
  + add\_customer (...)
  + add\_administrator (...)
  + remove\_airline (airline)
  + remove\_customer (customer)
  + remove\_administrator (administrator)
* Login Token
  + Login token class
    - Parameters: ID, name, role (customer/airline/administrator)
* Testings unit-pytest