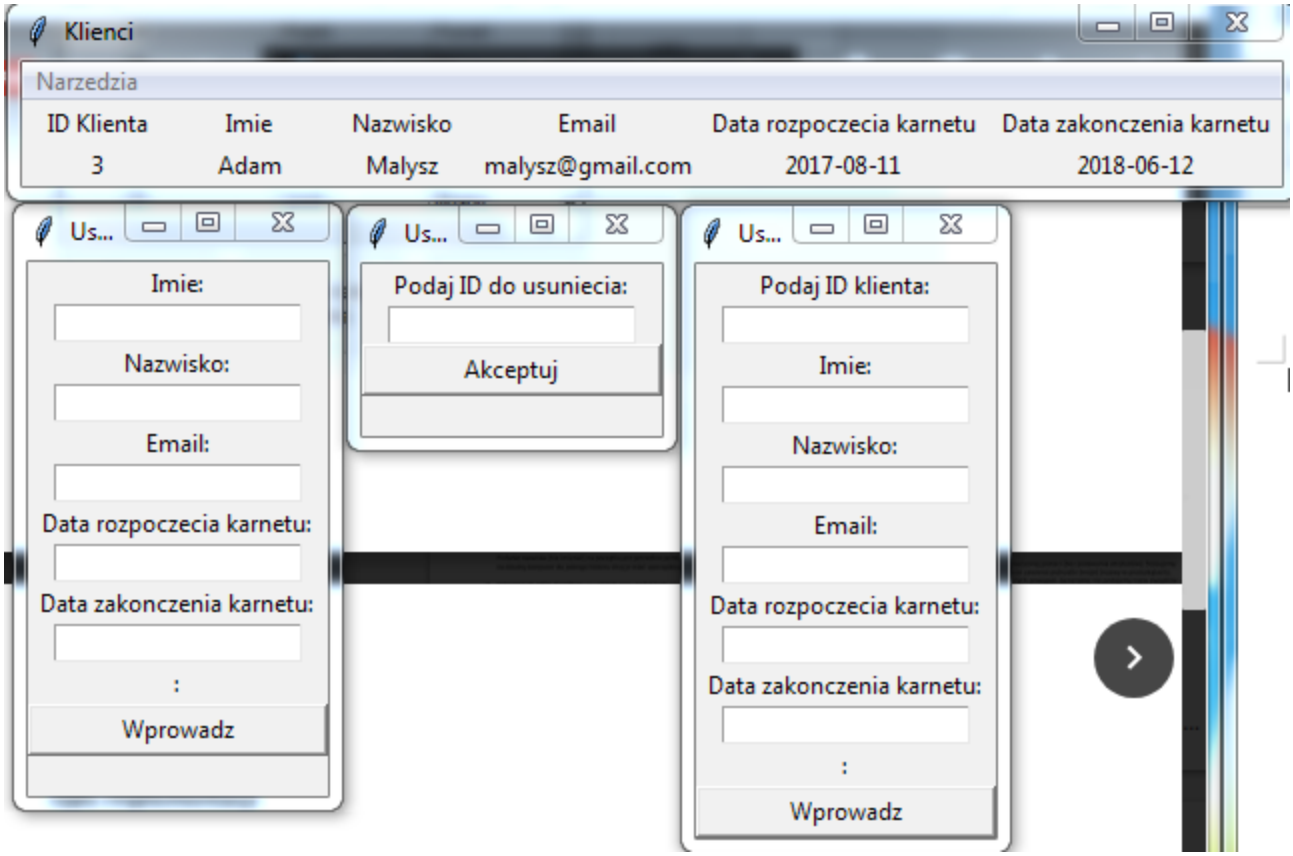
Dominik Krystkowiak

BD2-Projekt Sprawozdanie, Część 2

Temat: System rezerwacji – Fitness Klub

Część 2: Obsługa pojedynczej tabeli i kontrola integralności bazy danych



Opis implementacji:

Wybrano tabelę klienci

1. Operacje wykonywane w kontekście bazy danych:

```
Base = declarative_base()
root=Tk()
try:
    engine = create_engine("mssql+pyodbc://dominik:dominik@DOMINIK-LAPTOP\SQLEXPRESS/System_rezerwacji?driver=SQL+Server+Native+Client+11.0")
    metadata = MetaData(bind=engine)
except:
    tkinter.messagebox.showinfo("Blad", "Nie udalo sie polaczyc")
class Klienci(Base):
    try:
        __table__ = Table('Klienci', metadata, autoload=True)
    except:
        tkinter.messagebox.showinfo("Blad", "Nie udalo sie pobrac tabeli")
```

2. Zawartość tabeli odświeżana jest poleceniem:

```
    def readfromdatabase(self):
        session = create_session(bind=engine)
        testlist = session.query(Klienci).all();
        session.close()
        return testlist
```

3. Operacje na tabeli:

3.1. Dodawanie obiektu:

```
try:
    nowyKlient = Klienci(Imie = self.fnameEntry.get(), Nazwisko = self.lnameEntry.get(),
                         Adres_email = self.emailEntry.get(), Karnet_data_rozpoczecia=self.datestartEntry.get(),
                         Karnet_data_zakonczenia=self.dateendEntry.get())

    session = create_session(bind=engine)
    session.add(nowyKlient)
    session.flush()
    session.close()
    self.root.destroy()
    Odswiez()
except:
    tkinter.messagebox.showinfo("Blad", "Naruszono warunki spojnosci. \nAdres email musi byc w formie aaa@aaa.aaa, "
                                        "data w formacie YYYY.MM.DD, data zakonczenia pozniejsza od daty rozpoczecia")
```

3.2. Modyfikowanie obiektu:

```
try:
    session = create_session(bind=engine)
    q = session.query(Klienci)
    q = q.filter(Klienci.ID_Klienta == self.index_Entry.get())
    record = q.one()
    if len(self.fnameEntry.get())>0:
        record.Imie = self.fnameEntry.get()
    if len(self.lnameEntry.get())>0:
        record.Nazwisko = self.lnameEntry.get()
    if len(self.emailEntry.get())>0:
        record.r = self.emailEntry.get()
    if len(self.datestartEntry.get())>0:
        record.Karnet_data_rozpoczecia = self.datestartEntry.get()
    if len(self.dateendEntry.get())>0:
        record.Karnet_data_zakonczenia = self.dateendEntry.get()
    session.flush()
    session.close()
    self.root.destroy()
    Odswiez()
except:
    tkinter.messagebox.showinfo("Blad", "Naruszono warunki spojnosci lub nie ma takiego klienta. \n"
                                        "Adres email musi byc w formie aaa@aaa.aaa, data w formacie YYYY.MM.DD, data zakonczenia pozniejsza "
                                        "od daty rozpoczecia")
```

3.3. Usunięcie obiektu:

```
self.id = self.index_Entry.get()
session = create_session(bind=engine)
session.query(Klienci).filter(Klienci.ID_Klienta==self.id).delete()
session.close()
self.root.destroy()
Odswiez()
```

4. Obsługa wyjątków:
   Została zaprezentowana w funkcjach powyżej.

   Rodzaje obsługiwanych wyjątków:

   4.1. Naruszony warunek NOT NULL dla pól w tabeli.
   4.2. Naruszony warunek dla adresu email [tekst]@[tekst].[tekst]
   4.3. Naruszony warunek dla daty – data zakończenia musi być późniejsza niż data rozpoczęcia.
   4.4. Klucz główny posiada auto-inkrementację (IDENTITY).
   4.5. Istnieje możliwość dynamicznego zmieniania warunków spójności bazy danych bez ingerowania w kod programu. Zmiany wykonywane są na poziomie bazy danych.

   Definicja tabeli oraz warunków spójności:

```
create table Klienci
(ID_Klienta int PRIMARY KEY NOT NULL Identity(1,1),
    Imie varchar(25) NOT NULL,
    Nazwisko varchar(25) NOT NULL,
    Adres_email varchar(255),
    Karnet_data_rozpoczecia date NOT NULL,
    Karnet_data_zakonczenia date not null)
```

```
alter table Klienci
add constraint Email_check check (Adres_email like '[a-z,0-9,_,-]%@[a-z,0-9,_,-]%.[a-z][a-z]%')

alter table Klienci
add constraint Date_check check (Karnet_data_rozpoczecia < Karnet_data_zakonczenia)
```