Projekt na zajęcia z przedmiotu Bazy Danych

Utworzona aplikacja realizuje popularne „fiszki” czyli ang. „flashcards”, czyli popularną i uznawaną za nadzwyczaj skuteczną metodę zapamiętywania. Aplikacja posiada mechanizm ewidencji i autentykacji użytkowników. Użytkownicy mogą tworzyć zestawy fiszek i nadawać im opisy. Po wgraniu zestawu do bazy każdy użytkownik może go wyszukać podając słowa kluczowe, które są wyszukiwane w opisach. Po wyszukaniu i wybraniu zestawu użytkownik może go przeglądać i oceniać. Może też sprawdzić swoją wiedzę w jednym z kilku rodzajów testów.

Do utworzenia aplikacji użyto języka Python i framework’a Kivy. Dane przechowywane są w bazie danych MongoDB, w chmurze, dostarczanej przez Atlas MongoDB.

Kolekcje

Baza danych zawiera 4 kolekcje:

Cardssets:

Przechowuje dokumenty opisujące zestawy fiszek. Każdy dokument posiada pola:

- _id – identyfikator obiektu JSON
- Description – opis zestawu nadany przez użytkownika przy jego tworzeniu
- Creator – identyfikator twórcy zestawu
- Cards – kolekcja identyfikatorów kart należących do zestawu
- Avg_mark – średnia ocena zestawu

```
db.cardssets.find()
```

```
{
    "_id" : ObjectId("5ecd2dd6b625ad454adaf7e6"),
    "Description" : "mechanika101",
    "Creator" : ObjectId("5ecd2c24b625ad454adaf7dc"),
    "cards" : [
        ObjectId("5ecd2dd6b625ad454adaf7e7"),
        ObjectId("5ecd2dd6b625ad454adaf7e8"),
        ObjectId("5ecd2dd7b625ad454adaf7e9"),
        ObjectId("5ecd2dd8b625ad454adaf7ea"),
        ObjectId("5ecd2ddab625ad454adaf7eb"),
        ObjectId("5ecd2ddbb625ad454adaf7ec"),
        ObjectId("5ecd2ddcb625ad454adaf7ed"),
        ObjectId("5ecd2dddb625ad454adaf7ee")
    ],
    "avg_mark" : 4.0
}
{
    "_id" : ObjectId("5ecd2f131990ac9cbe473cc5"),
    "Description" : "Songs set",
    "Creator" : ObjectId("5ecd2ca4966cc127ad926066"),
    "cards" : [
        ObjectId("5ecd2f141990ac9cbe473cc6"),
        ObjectId("5ecd2f141990ac9cbe473cc7"),
        ObjectId("5ecd2f151990ac9cbe473cc8"),
        ObjectId("5ecd2f151990ac9cbe473cc9"),
        ObjectId("5ecd2f151990ac9cbe473cca"),
        ObjectId("5ecd2f161990ac9cbe473ccb"),
        ObjectId("5ecd2f161990ac9cbe473ccc")
    ],
    "avg_mark" : 4.33
}
```

Flashcards:

Przechowuje poszczególne karty. Każdy dokument posiada pola:

- _id – identyfikator obiektu JSON
- Question – treść na pierwszej stronie fiszki
- Answer – treść na drugiej stronie fiszki
- User – id twórcy fiszki
- Set – id zestawu do którego należy fiszka

```
db.flashcards.find()
```


```
{
    "_id" : ObjectId("5ecd2d4c1990ac9cbe473cc0"),
    "Question" : "1+1",
    "Answer" : "2",
    "User" : ObjectId("5ecd2ca4966cc127ad926066"),
    "Set" : ObjectId("5ecd2d4c1990ac9cbe473cbf")
}
{
    "_id" : ObjectId("5ecd2d4d1990ac9cbe473cc1"),
    "Question" : "5^2",
    "Answer" : "25",
    "User" : ObjectId("5ecd2ca4966cc127ad926066"),
    "Set" : ObjectId("5ecd2d4c1990ac9cbe473cbf")
}
{
    "_id" : ObjectId("5ecd2d4d1990ac9cbe473cc2"),
    "Question" : "10*10",
    "Answer" : "100",
    "User" : ObjectId("5ecd2ca4966cc127ad926066"),
    "Set" : ObjectId("5ecd2d4c1990ac9cbe473cbf")
}
{
    "_id" : ObjectId("5ecd2d4d1990ac9cbe473cc3"),
    "Question" : "80-70",
    "Answer" : "10",
    "User" : ObjectId("5ecd2ca4966cc127ad926066"),
    "Set" : ObjectId("5ecd2d4c1990ac9cbe473cbf")
}
```


Users:

Przechowuje informacje o użytkownikach. Każdy dokument posiada pola:

- _id – identyfikator obiektu JSON
- UserName – nazwa użytkownika
- Password – hasło dostępu użytkownika do systemu
- Email – adres emeil użytkownika
- CardsCreated – ilość utworzonych przez użytkownika kart

```
db.users.find()
{
    "_id" : ObjectId("5ed5fd73f9680b2ba09335ce"),
    "UserName" : "amadeusz",
    "password" : "123456789",
    "email" : "amadeusz1@gmail.com",
    "cardsCreated" : NumberInt(0)
}
{
    "_id" : ObjectId("5edd18a661580542ceef469e"),
    "UserName" : "awd",
    "password" : "123456789",
    "email" : "amadeusz2@gmail.com",
    "cardsCreated" : NumberInt(0)
}
{
    "_id" : ObjectId("5edd190f5115c8d522e7bcec"),
    "UserName" : "Amadeusz",
    "password" : "123456789",
    "email" : "amadeusz3@gmail.com",
    "cardsCreated" : NumberInt(0)
}
```

Ratings:

Przechowuje informacje o ocenach zestawów wystawianych przez użytkowników. Każdy dokument posiada pola:

- _id – identyfikator obiektu JSON
- Creator_ID – id użytkownika wystawiającego ocenę
- Set_ID – id zestawu do którego odnosi się ocena
- Mark – wartość oceny
- Description – komentarz do oceny

```
db.rating.find()
```


```
{
    "_id" : ObjectId("5ede50ddc753c5b6600ca87b"),
    "Creator_ID" : ObjectId("5ed5fd73f9680b2ba09335ce"),
    "Set_ID" : ObjectId("5ecf5d49cf2d8833edd5dc21"),
    "Mark" : "3",
    "Description" : "ok"
}
{
    "_id" : ObjectId("5ede50e9c753c5b6600ca87c"),
    "Creator_ID" : ObjectId("5ed5fd73f9680b2ba09335ce"),
    "Set_ID" : ObjectId("5ed5fda8f9680b2ba09335cf"),
    "Mark" : "1",
    "Description" : "kijowy"
}
{
    "_id" : ObjectId("5edee17dd6af939e0b03bbef"),
    "Creator_ID" : ObjectId("5ecd2ca4966cc127ad926066"),
    "Set_ID" : ObjectId("5ecd2f131990ac9cbe473cc5"),
    "Mark" : "5",
    "Description" : "Nice"
}
{
    "_id" : ObjectId("5edf7a45fa76b43f0a2c70d1"),
    "Creator_ID" : ObjectId("5ecd2c24b625ad454adaf7dc"),
    "Set_ID" : ObjectId("5ecd2f131990ac9cbe473cc5"),
    "Mark" : "3",
    "Description" : "bardzo dobry"
}
```


Funkcje dostępowe:

Funkcje dostępowe do bazy używane przez aplikację zostały zgromadzone w pliku DBConnection.py zawierającym klasę DBConnection której instancja jest używana do odczytu i zapisu informacji z bazy. Zdefiniowano następujące operacje:

Zarejestrowanie użytkownika

Funkcja była wywoływana po wypełnieniu formularza rejestracyjnego. Przyjmuje nazwę użytkownika, jego email i ustawione hasło, sprawdza czy taki użytkownik już istnieje oraz czy długość hasła jest wystarczająca. W przypadku pozytywnej weryfikacji dodaje użytkownika do bazy. W przeciwnym przypadku zwraca wartość informującą o błędzie, która jest obsługiwana w aplikacji.

```
def add_user(self, user_name, user_email, user_password):
    if len(user_password) < 8:
        return -1
    already_exists = self.db["users"].count_documents({"email": user_email})
    if already_exists > 0:
        return -2
    users = self.db["users"]
    users.insert_one({"UserName": user_name, "password": user_password, "email": user_email, "cardsCreated": 0})
    return 1
```

Autentykacja użytkownika

Funkcja jest wywoływana po wypełnieniu formularza logowania. Przyjmuje podany adres email i hasło, wyszukuje w tabeli Users email i porównuje podane hasło do przypisanego do tego rekordu w

bazie. Zwraca wartość informującą o poprawności operacji, która jest interpretowana w kodzie aplikacji który wywołał funkcję.

```
def user_auth(self, email, given_password):
    user = self.get_user(email)
    if user == 0:
        return -1
    result = self.db["users"].count_documents({'email': email, 'password': given_password})
    if result == 0:
        return -2
    return 1
```

Dodanie oceny

Funkcja jest wywoływana gdy użytkownik wprowadzi nową ocenę zestawu. Przyjmuje wartość oceny, ewentualny komentarz oraz identyfikatory użytkownika wystawiającego i zestawu ocenianego. Funkcja sprawdza czy dany użytkownik już ocenił ten zestaw oraz czy podana ocena jest z przewidzianego zakresu 0-5. Jeśli te oczekiwania nie są spełnione zwraca liczbę będącą komunikatem dla nadrzędnej funkcji. W przeciwnym przypadku dodaje do kolekcji rating nowy rekord zawierający tę ocenę. Ponadto wywołuje kolejną funkcję update_average_mark, która oblicza nową średnią ocenę ocenionego właśnie zestawu i zapisuje ją do tego zestawu w kolekcji Sets.

```
def add_rating(self, mark, description, creator_id, set_id):
    rate = self.db["rating"]
    if self.db["rating"].count_documents({"Creator_ID": creator_id, "Set_ID": set_id}) != 0:
        return -1  # already rated
    if not mark.isdigit():
        return -2  # mark isn't an intiger
    if int(mark) < 0 or int(mark) > 5:
        return -2
    rate.insert_one({"Creator_ID": creator_id, "Set_ID": set_id, "Mark": mark, "Description": description})
    self.update_average_mark(set_id)
    return 1
```

```
def update_average_mark(self, set_id):
    ratings = self.db["rating"].find({"Set_ID": set_id})
    marks_sum = 0
    marks_amount = 0
    for rating in ratings:
        marks_sum += int(rating["Mark"])
        marks_amount += 1
    self.db["cardssets"].update_one({"_id": set_id}, {"$set": {"avg_mark": round(marks_sum / marks_amount, 2)}})
```

Pobranie średniej oceny zestawu

Funkcja jest wywoływana gdy przy przeglądaniu zestawu użytkownik wejdzie w zakładkę Ratings. Przyjmuje id zestawu, pobiera i zwraca wartość pola avg_mark w kolekcji Cardssets ustawianego przez funkcję update_average_mark.

```
def get_set_average_mark(self, set_id):
    if (self.db["cardssets"].find_one({"_id": set_id}))["avg_mark"] == 0:
        return -1
    return (self.db["cardssets"].find_one({"_id": set_id}))["avg_mark"]
```

Wgrywanie zestawu

Funkcja jest wywoływana gdy użytkownik zakończy tworzenie zestawu. Funkcja przyjmuje obiekt set stworzony przez aplikację podczas tworzenia zestawu. Wstawia do kolekcji cardssets nowy dokument, a następnie dla każdej fiszki z setu wywołuje funkcję add_flashcard. Funkcja ta sprawdza czy identyczna fiszka nie została już stworzona, jeśli tak to informuje o tym aplikację przez zwracaną wartość. W przeciwnym wypadku dodaje fiszkę do kolekcji flashcards oraz dodaje jej id do kolekcji fiszek w odpowiadającym secie w tabeli cardssets.

```
def upload_set(self, cards_set):
    set_id = self.db["cardssets"].insert_one({"Description": cards_set.description,
                                              "Creator": cards_set.Creator,
                                              }).inserted_id
    for flashcard in cards_set.Flashcards:
        self.add_flashcard(flashcard.Question, flashcard.Answer, flashcard.User, set_id)
```

```
def add_flashcard(self, question, answer, creator_id, set_id):
    flashcards = self.db["flashcards"]
    if self.db["users"].count_documents({"_id": ObjectId(creator_id)}) == 0:
        return -1
    query = {"Question": question, "Answer": answer, "User": creator_id, "Set": set_id}
    if self.db["flashcards"].count_documents(query) != 0:
        return -2
    flash_card_id = flashcards.insert_one(
        {"Question": question, "Answer": answer, "User": creator_id, "Set": set_id}).inserted_id
    self.db["cardssets"].update_one({"_id": ObjectId(set_id)}, {"$addToSet": {"cards": flash_card_id}})
    return 1
```

Wyszukiwanie zestawu

Funkcja przyjmuje słowo klucz wprowadzone przez użytkownika podczas wyszukiwania zestawu. Następnie przeszukuje istniejące zestawy w kolekcji cardssets w poszukiwaniu takiego, którego opis zawiera słowo klucz. Tworzy i zwraca mapę zawierającą utworzone obiekty – zestawy z ich identyfikatorami jako kluczami.

```
def sets_list_for_selection(self, search_word):
    result_sets = {}
    for cards_set in self.db["cardssets"].find({"Description": {"$exists": "true"}}):
        if search_word in cards_set["Description"]:
            cur_set = classes.Set(cards_set["Creator"], cards_set["Description"], cards_set["_id"])
            for cardID in cards_set["cards"]:
                card = self.db["flashcards"].find_one({"_id": cardID})
                cur_set.addFlashcard(classes.Flashcard(card["Question"],
                                                       card["Answer"],
                                                       card["User"],
                                                       cards_set["_id"],
                                                       cardID))
            result_sets.update({cur_set.ID: cur_set})
            del cur_set

    return result_sets
```

Analogiczna funkcja jest wykorzystywana do pobrania wszystkich zestawów z bazy:

```
def all_sets(self):
    result_sets = {}
    for cards_set in self.db["cardssets"].find({"Description": {"$exists": "true"}}):
        cur_set = classes.Set(cards_set["Creator"], cards_set["Description"], cards_set["_id"])
        for cardID in cards_set["cards"]:
            card = self.db["flashcards"].find_one({"_id": cardID})
            cur_set.addFlashcard(classes.Flashcard(card["Question"],
                                                   card["Answer"],
                                                   card["User"],
                                                   cards_set["_id"],
                                                   cardID))
        result_sets.update({cur_set.ID: cur_set})
        del cur_set

    return result_sets
```