QuizPy

Autorzy: Mateusz Mazur, Wojciech Łącki, Mateusz Kleszcz

1. Opis

QuizPy jest aplikacją do quizowania stworzona za pomocą MongoDB oraz Python.

Umożliwia ona przeprowadzanie quizu pojedynczemu użytkownikowi. Aplikacja daje możliwość tworzenia własnych zcustomizowanych quizów. Pytania mogą przybierać różną formę. Gracz może rozpocząć rozgrywkę przez wybór jednego z prywatnych lub udostępnionych publicznie quizów. W trakcie rozgrywki prowadzony będzie ranking graczy, na podstawie poprawności i szybkości udzielanych odpowiedzi. Po zakończonej grze osoby z najwyższymi wynikami będą odpowiednio graficznie wyróżnione.

2. Baza danych

Baza danych zawiera 3 kolekcje:

a) Users


```
{
  "username": "name",
  "password": {
    "$binary": {
      "base64":
"JDJiJDEyJDl4eS44MTlDU3BSc1R6c3BqU0JUYU9odWtYOWRnaGFEaDNMU0h1M0pUMC9DbllUaS9OeS5h",
      "subType": "00"
    }
  }
}
```

b) Quizes


```
{
  "category": "Polska",
  "name": "Nowy quiz",
  "questions": [
    {
      "question": "Ustaw w kolejności",
      "type": "correctOrder",
      "answers": [
        "3",
        "2",
        "1",
        "4"
      ],
      "correct": [
        2,
        1,
        0,
        3
      ]
    },
    {
      "question": "Wybierz jeden",
      "type": "chooseOne",
      "answers": [
        "1",
        "2",
        "3",
        "4"
      ],
      "correct": 0
    }
  ],
  "username": "Student",
  "results": [
    {
      "username": "q",
      "points": 3839,
      "date": "21/06/2022 12:11:47",
    }
  ]
}
```


Pole „answers” przechowuje możliwe odpowiedzi, natomiast „correct” jakąś formę weryfikacji udzielonych odpowiedzi. W „results” przechowywane są wyniki i daty ukończonych podejść do quiz’u wraz z nazwą użytkownika, który zmierzył się z quizem.

c) Categories

```
{
  "name": "Polska"
}
```

3. Funkcje

a) Stworzenie użytkownika

```
def create_user(self, username, password):
    hashed = bcrypt.hashpw(password.encode("utf-8"), bcrypt.gensalt())
    x = self.db.Users.insert_one({"username": username, "password":
hashed})
    if x:
        self.user = username
        return True
    self.user = None
    return False
```

b) Logowanie użytkownika

```
def login_user(self, username, password):
    user = self.db.Users.find_one({"username": username})
    if user:
        if bcrypt.checkpw(password.encode("utf-8"), user["password"]):
            self.user = username
            return True
    self.user = None
    return False
```

c) Sprawdzenie czy użytkownik istnieje

```
def check_if_user_exist(self, username):
    collection = self.db.Users
    if collection.count_documents({"username": username}) == 0:
        return False
    return True
```

d) Zapisanie wyniku po rozwiązaniu quizu

```
def save_score(self, quiz_id, points):
    date = now.strftime("%d/%m/%Y %H:%M:%S")
    new_result = {"username": self.user, "points": points, "date": date}
    self.db.Quizes.update_one({"_id": quiz_id}, {"$push": {"results":
new_result}})
```

e) Stworzenie nowego quizu

```
def save_quiz(self, quiz):
    quiz_obj = quiz.to_json()
    quiz_obj = json.loads(quiz_obj)
    a = self.db.Quizes.insert_one(quiz_obj)
    if a:
        return True
    return False
```

f) Zapytanie do bazy o „k” najlepszych rezultatów quizu o id „quiz_id”

```
def find_k_best_results(self, quiz_id, k):
    collection = self.db.Quizes
    ranking = list(collection.aggregate([
        {"$match": {
            "_id": quiz_id
        }},
        {"$unwind": "$results"},
        {"$sort": {
            "results.points": -1
        }},
        {"$project": {
            "_id": 0,
            "results": 1
        }},
        {"$limit": k}
    ]))
    return ranking
```

Funkcja ta zwraca z bazy „k” najlepszych wyników uzyskanych przez graczy w danym quizie.

g) Odnalezienie quizów z danej kategorii.

```
def find_quizes(self, categoryName):
    collection = self.db.Quizes
    quiz_list = list(collection.find({"category": categoryName},
{"results": 0}))
    return quiz_list
```

h) Pobranie kategorii

```
def find_categories(self):
    collection = self.db.Categories
    categories = list(collection.find({}))
    return categories
```

3. Triggery i Logi

W MongoDB Atlas zostały dodane odpowiednie triggery, które wykrywają dodanie jakiegoś dokumentu w którejś z kolekcji.

Name	Trigger Type	Source	Schedule	Status
user_add_trigger	Database	QuizPy		Enabled
quiz_add_trigger	Database	QuizPy		Enabled
category_add_trigger	Database	QuizPy		Enabled

Jeśli chodzi o kolekcję użytkowników, to zapamiętywana jest operacja jaka jest wykonywana (czyli insert) oraz nazwa użytkownika, który zostaje dodany.

```
console.log(changeEvent.operationType);
console.log(changeEvent.fullDocument.username);
```

W kolekcjach quizów i kategorii, zapisywana jest nazwa danego quizu, czy też kategorii.

```
console.log(changeEvent.operationType);
console.log(changeEvent.fullDocument.name);
```

Poniżej możemy zobaczyć, jak wygląda typowy wpis log'a przy tworzeniu użytkownika. Został stworzony użytkownik „user1".

```
Logs:
[
  "insert",
  "user1"
]
```