Wydział Informatyki Politechniki Białostockiej Przedmiot: Komunikacja Człowiek-Komputer	Data oddania projektu: 27.11.2018 r.
Zespół: 1. Michał Kuc 2. Marek Ciborowski	Prowadzący: mgr inż. Łukasz Gadomer Ocena:

1. **Opis projektu.**

Projekt ma na celu stworzenie platformy do wypełniania ankiet, który tworzą użytkownicy. Każdy ma możliwość założenia konta, stworzenia ankiety (która zawiera pytania jednokrotnego, wielokrotnego wyboru, umożliwia tworzenie własnych odpowiedzi), przeglądania jej wyników czy obserwowania użytkowników (lista obserwowanych użytkowników może nam ułatwić znalezienie ankiet, które interesują nas najbardziej).

2. **Opis funkcjonalności.**

RepositoryLayer:

Każde repozytorium zawiera podstawowe metody do tworzenia poszczególnych modelów, dodawania do bazy, usuwania z bazy, wyszukiwania poszczególnych danych na podstawie głównie numeru ID (rzadko inne parametry są wykorzystywane do wyszukiwania np. po loginie i haśle). Repozytorium może zawierać metody wyspecjalizowane np. walidację loginu, hasła, emailu itp. Te zostaną opisane poniżej:

a) **AccountRepository.cs:** klasa zawierająca funkcjonalność użytkownika powiązaną z bazą danych oraz walidacją konta
   i) **DidFillSurvey(int? accountID, int? surveyID)** - metoda sprawdzająca, czy użytkownik o danym nr ID wypełniał ankietę o danym nr ID;
   ii) **IsEmailCorrect(string email)** - walidacja adresu email podanego przy rejestracji konta (sprawdzanie, czy format jest poprawny (wykorzystuje metodę prywatną **IsValidEmail(string email)** oraz czy nie jest zajęty przez innego użytkownika);
   iii) **IsNicknameCorrect(string nick)** - sprawdzenie, czy nick ma odpowiednią ilość znaków oraz czy nie jest zajęty przez innego użytkownika;
   iv) **IsFollowed(int followerId, int followedId)** - sprawdzanie, czy podany w pierwszym parametrze użytkownik nie obserwuje użytkownika podanego w drugim parametrze;
   v) **hashPassword(string password)** - metoda do szyfrowania hasła (cel: bezpieczeństwo);

b) **AnswerRepository.cs**
   i) **GetAccountsVoters(int? answerID)** - zwraca listę kont, która głosowała na daną odpowiedź;
   ii) **GetQuantityOfVotes(int? answerID)** - zwraca liczbę głosów na daną odpowiedź;

c) **SurveyRepository.cs**

i) **IsAnonymous(int? id)** - zwraca wartość true, gdy ankieta o podanym nr ID jest anonimowa, w przeciwnym wypadku false;

d) **QuestionRepository.cs:**
   i) **GetQuestionCategory(bool canAddOwnAnswer, bool isSingleChoice)** - na podstawie wartości boolowskich zwraca odpowiednią kategorię pytania o podanych parametrach (istnieją tylko 4 kategorie);

e) **UserSecurityRepository.cs:**
   i) **IsLoginFree(string login)** - walidacja loginu przy rejestracji konta;

3. **Opis realizacji zmiany warstwy prezentacji z wyszczególnieniem oddzielenia od niej logiki aplikacji**

   Do warstwy prezentacji został utworzony nowy projekt w solucji korzystający z .NET Frameworka w architekturze MVC. Do widoków został użyty silnik ASP.NET Razor.

4. **Szczególnie interesujące zagadnienia projektowe:**
   a) sortowanie i wyszukiwanie ankiet i użytkowników z użyciem pakietu PagedList.
   b) przechowywanie obiektu aktualnie zalogowanego konta w sesji (sprawdzane w autoryzacji)
   c) metoda CreateSurvey:
   d)

```
[HttpPost]
[ValidateAntiForgeryToken]
          public ActionResult CreateSurvey(CreateSurveyVM
createSurveyVM, string button)
{
    if (button == "AddQuestion")
    {
          if (createSurveyVM.newQuestion.questionValue ==
null || createSurveyVM.newQuestion.questionValue == "")
        {

ModelState.AddModelError(string.Format("newQuestion.questionValue"
), "You can't add empty question.");
            return View(createSurveyVM);
        }
            createSurveyVM.newQuestion.questionValueCopy =
createSurveyVM.newQuestion.questionValue;

createSurveyVM.questions.Add(createSurveyVM.newQuestion);
        return View(createSurveyVM);
    }
    // Create Survey
    else if (button == "Confirm")
```

```
            {
                if (ModelState.IsValid)
                {
                                                    Account account =
(Account)Session["CurrentUser"];
                    int index = 0;
                    bool isOk = true;
                              if (createSurveyVM.title == null ||
createSurveyVM.title == "")
                    {

ModelState.AddModelError(string.Format("title"), "You can't add
survey without title.");
                        isOk = false;
                    }
                         if (createSurveyVM.description == null ||
createSurveyVM.description == "")
                    {

ModelState.AddModelError(string.Format("description"), "You can't
add survey without description.");
                        isOk = false;
                    }
                       foreach (CreateSurveyVM.NewQuestion question
in createSurveyVM.questions)
                    {
                        if (question.answers.Count < 2)
                        {

ModelState.AddModelError(string.Format("questions[{0}].questionVal
ue", index), "All questions must have 2 or more answers.");
                            isOk = false;
                        }
                        int i = 0;
                                        foreach (string answer in
question.answers)
                        {
                            if (answer == "")
                            {

ModelState.AddModelError(string.Format("questions[{0}].answers[{1}
]", index, i), "You can't add empty answer.");
                                isOk = false;
                            }
                            i++;
                        }
```

```
                            if (question.questionValue == null ||
question.questionValue == "")
                        {

ModelState.AddModelError(string.Format("questions[{0}].questionVal
ue", index), "You can't add empty question.");
                            isOk = false;
                        }
                        index++;
                    }
                    if (!isOk)
                        return View(createSurveyVM);

                            ICollection<Question> questions = new
List<Question>();
                      foreach (CreateSurveyVM.NewQuestion question
in createSurveyVM.questions)
                    {
                                ICollection<Answer> answers = new
List<Answer>();
                                        foreach (string answer in
question.answers)
                        {
                                            Answer answerCreated =
answerRepository.CreateAnswer(answer);
                            answers.Add(answerCreated);
                        }
                                        Question questionCreated =
questionRepository.CreateQuestion(question.questionValue,
question.canAddOwnAnswers, question.isSingleChoice, answers);
                        questions.Add(questionCreated);
                    }
                                                Survey survey =
surveyRepository.CreateSurvey(createSurveyVM.title,
createSurveyVM.description,             createSurveyVM.isAnonymous,
questions);
                    surveyRepository.AddSurvey(survey, account);
                    return RedirectToAction("Index", "Home");
                }
                return View(createSurveyVM);
            }
            // Delete Answer
            else if (button.Contains("DeleteAnswer"))
            {
                string indexI = "", indexJ = "";
                int i = 0;
```

```
                for (i = 13; i < button.Length; i++)
                {
                    string slice = button.Substring(i, 1);
                    if (string.Equals(slice, " "))
                        break;
                    indexI = slice;
                }
                indexJ = button.Substring(13 + indexI.Length + 1);
                i = Int32.Parse(indexI);
                int j = Int32.Parse(indexJ);
                createSurveyVM.questions[i].answers.RemoveAt(j);

createSurveyVM.questions[i].answersCopy.RemoveAt(j);
            }

            // Delete Question
            else if (button.Contains("DeleteQuestion"))
            {
                string index = button.Substring(15);
                int i = Int32.Parse(index);
                createSurveyVM.questions.RemoveAt(i);
            }

            // Add Answer and validation
            else
            {
                int index = Int32.Parse(button);
                                    string  newAnswerValue  =
createSurveyVM.questions[index].newAnswer;
                bool isOk = true;
                if (string.IsNullOrEmpty(newAnswerValue))
                {

ModelState.AddModelError(string.Format("questions[{0}].newAnswer",
index), "You can't add empty answer.");
                    isOk = false;
                }
                if (isOk)
                {

createSurveyVM.questions[index].answers.Add(newAnswerValue);

createSurveyVM.questions[index].answersCopy.Add(newAnswerValue);
                }
                isOk = true;
                int i = 0;
```

```
                                    foreach (string answer in
createSurveyVM.questions[index].answers)
                {
                    if (answer == "")
                    {

ModelState.AddModelError(string.Format("questions[{0}].answers[{1}
]", index, i), "You can't add empty answer.");
                        isOk = false;
                    }
                    i++;
                }
                  if (createSurveyVM.questions[index].questionValue
== null)
                {

ModelState.AddModelError(string.Format("questions[{0}].questionVal
ue", index), "You can't add empty question.");
                    isOk = false;
                }
                if (!isOk)
                {

createSurveyVM.questions[index].answers.Clear();

createSurveyVM.questions[index].answers.AddRange(createSurveyVM.qu
estions[index].answersCopy);
                     createSurveyVM.questions[index].questionValue
=

createSurveyVM.questions[index].questionValueCopy;
                }
                else
                {

createSurveyVM.questions[index].answersCopy.Clear();

createSurveyVM.questions[index].answersCopy.AddRange(createSurveyV
M.questions[index].answers);

createSurveyVM.questions[index].questionValueCopy =

createSurveyVM.questions[index].questionValue;
                }
            }
            return View(createSurveyVM);
```

```
}
```

5. **Instrukcja użytkownika:**

Nowy użytkownik ma możliwość założenia własnego konta po podaniu swoich danych. Następnie ma do wyboru wiele opcji, m.in.: dodawanie własnych ankiet, przeglądanie ich wyników, obserwowanie innych użytkowników, wypełnianie ankiet, zmianę hasła jak również wylogowanie się. Istnieje również możliwość usunięcia konta.

6. **Wnioski:**

Mieliśmy możliwość dowiedzieć się kilku nowych rzeczy z zakresu technologii .NET Framework. Ponadto poszerzyliśmy naszą wiedzę z zakresu Entity Framework, a także frontendu.

7. **Samoocena: 4.5**