Министерство образования и науки Нижегородской области

Государственное бюджетное профессиональное образовательное учреждение

«Нижегородский Губернский колледж»

**ОТЧЕТ**

Тестирование сайта «Расписание НГК»

Студент:\_\_\_\_\_\_\_\_\_\_\_\_\_Э. О. Токарева 19.10.2024

(подпись)

Руководитель:\_\_\_\_\_\_\_\_\_\_\_\_\_Е. П. Голубева 19.10.2024

(подпись)

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2024

**Содержание**

[1. Тест план 3](#_Toc180674339)

[1. Цели и задачи тестирования 3](#_Toc180674340)

[2. Объект тестирования 3](#_Toc180674341)

[3. Стратегия тестирования 3](#_Toc180674342)

[4. Методология тестирования 3](#_Toc180674343)

[5. Инструменты для тестирования 3](#_Toc180674344)

[6. Критерии завершения тестирования 3](#_Toc180674345)

[2. Тестовый набор 4](#_Toc180674346)

[3. Тест-кейсы 4](#_Toc180674347)

[Приложения 5](#_Toc180674348)

# Тест план

## Цели и задачи тестирования

Цель: обеспечить корректность, производительность и безопасность базы данных.

Задачи:

* + Проверка триггеров базы данных
  + Проверка целостности данных
  + Проверка функциональности CRUD (Create, Read, Update, Delete) операций

## Объект тестирования

База данных, содержащая таблицы пользователей, категорий, целей, задач, сфер.

## Стратегия тестирования

Подход к тестированию: Автоматизированное. Создание Unit-тестов для автоматизации проверки функциональности базы данных.

Уровень тестирования: Модульное. Тестирование работы базы данных, без ее взаимодействия с бизнес-логикой будущего мобильного приложения.

## Методология тестирования

Функциональное тестирование: проверка всех функций CRUD и триггеров.

Тестирование на целостность данных: Проверка ограничений целостности (первичные ключи, внешние ключи).

## Инструменты для тестирования

Для создания Unit-тестов используется фреймворк MSTest, язык C#. IDE: Microsoft Visual Studio. СУБД: Supabase

## Критерии завершения тестирования

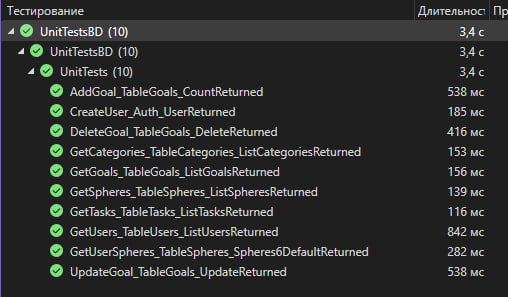
* Все тесты пройдены успешно
* Не найдено уязвимостей

# Тестовый набор

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Тест** | **Описание** | **Ожидаемый результат** |
| 1 | Создание нового пользователя | Добавление нового пользователя в таблицу Auth, функция возвращает нового user | user не равен null |
| 2 | Триггер добавления сфер | Проверка триггера, создающего 6 значений сфер по умолчанию при добавлении нового пользователя | spheres.Count = 6 |
| 3 | Выборка пользователей | Получение листа всех пользователей, функция возвращает лист | users - лист с данными |
| 4 | Выборка категорий | Получение листа всех категорий, функция возвращает лист | categories - лист с данными |
| 5 | Выборка целей | Получение листа всех целей, функция возвращает лист | goals - лист с данными |
| 6 | Выборка сфер | Получение листа всех сфер, функция возвращает лист | spheres - лист с данными |
| 7 | Выборка задач | Получение листа всех задач, функция возвращает лист | tasks - лист с данными |
| 8 | Добавление целей | Добавление цели в таблицу целей. Возвращает количество целей до и после добавления и сравнивает, что цель стала на 1 больше | count1 = count2 |
| 9 | Изменение целей | Изменение статуса цели под определенным идентификатором, функция возвращает статус Http | Успешный статус |
| 10 | Удаление целей | Удаление цели по его идентификатору, функция возвращает сообщение об успешном удалении | Сообщение о успешном результате |

# Тест-кейсы

Разработано 10 тестов, все тесты пройдены успешны.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название** | **Результат** | **Прил.** | **Пройден** |
| 1 | CreateUser\_Auth\_UserReturned | Assert.IsNotNull(user); | [Код теста 1](#_Приложения) | + |
| 2 | GetUserSpheres\_TableSpheres\_Spheres6DefaultReturned | Assert.AreEqual(spheres.Count, 6); | [Код теста 2](#_Приложения) | + |
| 3 | GetUsers\_TableUsers\_ListUsersReturned | Assert.IsNotNull(users); | [Код теста 3](#_Приложения) | + |
| 4 | GetCategories\_TableCategories\_ListCategoriesReturned | Assert.IsNotNull(categories); | [Код теста 4](#_Приложения) | + |
| 5 | GetGoals\_TableGoals\_ListGoalsReturned | Assert.IsNotNull(goals); | [Код теста 5](#_Приложения) | + |
| 6 | GetSpheres\_TableSpheres\_ListSpheresReturned | Assert.IsNotNull(spheres); | [Код теста 6](#_Приложения) | + |
| 7 | GetTasks\_TableTasks\_ListTasksReturned | Assert.IsNotNull(tasks); | [Код теста 7](#_Приложения) | + |
| 8 | AddGoal\_TableGoals\_CountReturned | Assert.AreEqual(count1, count2); | [Код теста 8](#_Приложения) | + |
| 9 | UpdateGoal\_TableGoals\_UpdateReturned | Assert.AreEqual(code, HttpStatusCode.OK); | [Код теста 9](#_Приложения) | + |
| 10 | DeleteGoal\_TableGoals\_DeleteReturned | Assert.IsTrue(result2.IsCompleted); | [Код теста 10](#_Приложения) | + |

# Приложения

Код теста 1

|  |
| --- |
| [TestMethod]  public void CreateUser\_Auth\_UserReturned()  {  string email = "test1234@bk.ru";  string password = "test1234";  var session = CreateUser(email, password);  session.Wait();  var user = session.Result.User;  Assert.IsNotNull(user);  } |

Код теста 2

|  |
| --- |
| [TestMethod]  public void GetUserSpheres\_TableSpheres\_Spheres6DefaultReturned()  {  var session = SignIn("testUpdateDelete@bk.ru", "testUpdateDelete123");  session.Wait();  var user = session.Result.User;  Guid id = Guid.Parse(user.Id);  var temp = GetUserSpheres(id);  temp.Wait();  var spheres = temp.Result.Models;  Assert.AreEqual(spheres.Count, 6);  } |

Код теста 3

|  |
| --- |
| [TestMethod]  public void GetUsers\_TableUsers\_ListUsersReturned()  {  var result = GetUsers();  result.Wait();  var users = result.Result.Models;  Assert.IsNotNull(users);  } |

Код теста 4

|  |
| --- |
| [TestMethod]  public void GetCategories\_TableCategories\_ListCategoriesReturned()  {  var result = GetCategories();  result.Wait();  var categories = result.Result.Models;  Assert.IsNotNull(categories);  } |

Код теста 5

|  |
| --- |
| [TestMethod]  public void GetGoals\_TableGoals\_ListGoalsReturned()  {  var result = GetGoals();  result.Wait();  var goals = result.Result.Models;  Assert.IsNotNull(goals);  } |

Код теста 6

|  |
| --- |
| [TestMethod]  public void GetSpheres\_TableSpheres\_ListSpheresReturned()  {  var result = GetSpheres();  result.Wait();  var spheres = result.Result.Models;  Assert.IsNotNull(spheres);  } |

Код теста 7

|  |
| --- |
| [TestMethod]  public void GetTasks\_TableTasks\_ListTasksReturned()  {  var result = GetTasks();  result.Wait();  var tasks = result.Result.Models;  Assert.IsNotNull(tasks);  } |

Код теста 8

|  |
| --- |
| [TestMethod]  public void AddGoal\_TableGoals\_CountReturned()  {  var session = SignIn("testUpdateDelete@bk.ru", "testUpdateDelete123");  session.Wait();  var result1 = GetGoals();  result1.Wait();  var temp = GetSpheres();  temp.Wait();  Guid id = Guid.Parse(session.Result.User.Id);  var sphere = temp.Result.Models.FirstOrDefault(x => x.IdUser == id).Id;  var goal = new Goals  {  Name = "Тестовая цель",  Description = "цель 1",  IdSphere = sphere,  Status = false,  IdUser = Guid.Parse(session.Result.User.Id)  };  var result2 = AddGoal(goal);  result2.Wait();  int count1 = result1.Result.Models.Count + 1;  int count2 = result2.Result.Models.Count;  Assert.AreEqual(count1, count2);  } |

Код теста 9

|  |
| --- |
| [TestMethod]  public void UpdateGoal\_TableGoals\_UpdateReturned()  {  var session = SignIn("testUpdateDelete@bk.ru", "testUpdateDelete123");  session.Wait();  var user = session.Result.User;  Guid id = Guid.Parse(user.Id);  var result = UpdateGoal(id, true);  result.Wait();  HttpStatusCode code = result.Result.ResponseMessage.StatusCode;    Assert.AreEqual(code, HttpStatusCode.OK);  } |

Код теста 10

|  |
| --- |
| [TestMethod]  public void DeleteGoal\_TableGoals\_DeleteReturned()  {  var session = SignIn("testUpdateDelete@bk.ru", "testUpdateDelete123");  session.Wait();  Guid id = Guid.Parse(session.Result.User.Id);  var goals = GetGoals();  goals.Wait();  Guid firstIdGoals = goals.Result.Models.First(x => x.IdUser == id).Id;  var result2 = DeleteGoal(firstIdGoals);  result2.Wait();  Assert.IsTrue(result2.IsCompleted);  } |