Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2018 г.

дата

Отчет

По лабораторной работе

по дисциплине

«ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПО»

ЛР 09.03.04.10.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(и. о., фамилия)

Преподаватель\_\_\_\_\_доцент, к. ф.-м. н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.М. Старолетов

должность, ученое звание (и. о., фамилия)

Барнаул 2019

**Задание.**

* Предоставить проект с написанной документацией для тестирования
* Выполнить модульное тестирование для проекта одного из участника команды

В качестве тестируемого ПО было предоставлено приложение для электронного портфолио студента и преподавателя в рамках автоматизированной системы университета

Исходный код:

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace patternsKursach.dataMapper

{

public class StudentMapper

{

public static string studetTable = "student";

public static string workTable = "work";

public static string studId = "stud\_id";

public static string attestTable = "attestation";

public static string progressTable = "dostizh";

private StorageAdapter adapter;

public StudentMapper(StorageAdapter adapter)

{

this.adapter = adapter;

}

/// <summary>

/// Добавление студента

/// </summary>

/// <param name="column">Колонки</param>

public void addStudent(List<Object> column)

{

adapter.add(studetTable, column);

}

/// <summary>

/// Редактирование студента

/// </summary>

/// <param name="columns">Колонки</param>

/// <param name="id">id записи</param>

public void editStudent(Dictionary<String, Object> columns, int id)

{

adapter.edit(studetTable, columns, id);

}

/// <summary>

/// Поиск студента по id

/// </summary>

/// <param name="id">id студента</param>

/// <returns>Студент</returns>

public Student findStudentById(int id)

{

DataRow data = adapter.findId(id, studetTable);

return new Student(data);

}

/// <summary>

/// Вывод всех студентов

/// </summary>

/// <returns>Список студентов</returns>

public List<Student> findAll()

{

DataRow[] datas = adapter.findAll(studetTable);

List<Student> students = new List<Student>();

foreach(DataRow data in datas){

students.Add(new Student(data));

}

return students;

}

/// <summary>

/// Поиск работы по id

/// </summary>

/// <param name="id">id</param>

/// <returns>Работа</returns>

public Work findByWorkId(int id)

{

DataRow data = adapter.findId(id, workTable);

return new Work(data);

}

/// <summary>

/// Поиск работ по id студента

/// </summary>

/// <param name="id">id студента</param>

/// <returns>Работы студента</returns>

public List<Work> findAllWorkStudent(int id)

{

DataRow[] datas = adapter.findSideKick(workTable, studId, id);

List<Work> works = new List<Work>();

foreach (DataRow data in datas)

{

works.Add(new Work(data));

}

return works;

}

/// <summary>

/// Добавление работы

/// </summary>

/// <param name="column">Поля</param>

public void addWork(List<Object> column)

{

adapter.add(workTable, column);

}

/// <summary>

/// Редактирование работы

/// </summary>

/// <param name="columns">Поля</param>

/// <param name="id">id работы</param>

public void editWork(Dictionary<String, Object> columns, int id)

{

adapter.edit(workTable, columns, id);

}

/// <summary>

/// Поиск аттестации по id студента

/// </summary>

/// <param name="id">id студента</param>

/// <returns>Аттестация</returns>

public List<Attestation> findAllAttestation(int id)

{

DataRow[] datas = adapter.findSideKick(attestTable, studId, id);

List<Attestation> attestation = new List<Attestation>();

foreach (DataRow data in datas)

{

attestation.Add(new Attestation(data));

}

return attestation;

}

/// <summary>

/// Добавление аттестации

/// </summary>

/// <param name="column">поля</param>

public void addAttestation(List<Object> column)

{

adapter.add(attestTable, column);

}

/// <summary>

/// Редактирование атестации

/// </summary>

/// <param name="columns">Поля</param>

/// <param name="id">id аттестации</param>

public void editAttestation(Dictionary<String, Object> columns, int id)

{

adapter.edit(attestTable, columns, id);

}

/// <summary>

/// Поиск достижение по id студента

/// </summary>

/// <param name="id">id студента</param>

/// <returns>Достижения</returns>

public List<Progress> findAllProgres(int id)

{

DataRow[] datas = adapter.findSideKick(progressTable, studId, id);

List<Progress> progress = new List<Progress>();

foreach (DataRow data in datas)

{

progress.Add(new Progress(data));

}

return progress;

}

/// <summary>

/// Добавление достижений

/// </summary>

/// <param name="column">поля</param>

public void addProgress(List<Object> column)

{

adapter.add(progressTable, column);

}

/// <summary>

/// Редактирование достижений

/// </summary>

/// <param name="columns">поля</param>

/// <param name="id">id достижения</param>

public void editProgress(Dictionary<String, Object> columns, int id)

{

adapter.edit(progressTable, columns, id);

}

/// <summary>

/// Удаление записи

/// </summary>

/// <param name="table">таблица</param>

/// <param name="id">id записи</param>

public void delete(String table, int id)

{

adapter.delete(table, id);

}

}

}

В качестве ПО для тестирования досталось приложения для электронного портфолио студента и преподавателя в рамках автоматизированной системы университета

using System;

using System.Collections.Generic;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using MySql.Data.MySqlClient;

using patternsKursach;

using patternsKursach.dataMapper;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

private StorageAdapter adapter = new StorageAdapter();

private StudentMapper studentMapper;

private TeacherMapper teacherMapper;

public UnitTest1()

{

studentMapper = new StudentMapper(adapter);

teacherMapper = new TeacherMapper(adapter);

}

[TestMethod]

public void AddStudentTest()

{

List<Object> columns = new List<object>();

columns.Add("sdfsdf");

columns.Add("sdfsdf");

columns.Add("sdfsdf");

studentMapper.addStudent(columns);

}

[TestMethod]

public void AllStudentTet()

{

List<Student> students = studentMapper.findAll();

Assert.AreEqual(true, students.Count > 0);

}

[TestMethod]

public void FindStudentTest()

{

Student student = studentMapper.findStudentById(15);

Assert.AreEqual(15, student.Id);

}

[TestMethod]

public void editStudentTest()

{

Dictionary<String, object> columns = new Dictionary<string, object>();

columns.Add("name", "чебур");

studentMapper.editStudent(columns, 15);

Assert.AreEqual("чебур", studentMapper.findStudentById(15).Name);

}

[TestMethod]

public void findWorksTest()

{

List<Work> works = studentMapper.findAllWorkStudent(2);

Assert.AreEqual(21, works.Count);

}

[TestMethod]

public void findAttestationTest()

{

List<Attestation> attestations = studentMapper.findAllAttestation(2);

Assert.AreEqual(82, attestations.Count);

}

[TestMethod]

public void addTeacher()

{

List<object> column = new List<object>

{

"gjh",

"ghfh",

"jhgjuh"

};

teacherMapper.addTeacher(column);

}

[TestMethod]

[ExpectedException(typeof(MySqlException),

"Column count doesn't match value count at row 1")]

public void failAddTeacher()

{

List<object> column = new List<object>

{

"gjh",

"jhgjuh"

};

teacherMapper.addTeacher(column);

}

[TestMethod]

public void findTeachersTest()

{

List<Teacher> teachers = teacherMapper.findAll();

Assert.AreEqual(true, teachers.Count > 0);

}

[TestMethod]

public void findMetodickaTest()

{

List<Metodichka> metodichkas = teacherMapper.findAllMetodichkaTeacher(3);

Assert.AreEqual(67, metodichkas.Count);

}

[TestMethod]

public void failFindMetodickaTest()

{

List<Metodichka> metodichkas = teacherMapper.findAllMetodichkaTeacher(-3);

Assert.IsNotNull(metodichkas);

}

}

}