Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**РАЗРАБОТКА ПРОГРАММЫ**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Корпоративная информационная система Pro-complect»**

Выполнил:

студент гр.ИСТ-19-2б

Лебедева А.А.

Принял:

ст. преподаватель, Банников Р.Ю.

**Пермь 2023**

**РЕАЛИЗАЦИЯ ПРОГРАММЫ**

**Цель работы:** запрограммировать обозначенный в ТЗ функционал с учётом созданной БД и разработанных структур классов.

**Задачи:**

1. Запрограммировать обозначенный в ТЗ функционал с учётом созданной БД и разработанных структур классов;
2. Привести программный код, согласно функциям.
3. **Разработка программы**

Пройдемся по основному функционалу системы:

1. В системе должна быть функция авторизации

Внешний интерфейс для функции авторизации представлен на рисунке 1. Здесь нужно ввести логин и пароль для того, чтобы войти в приложение.

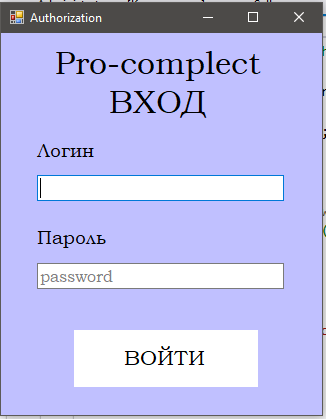


Рисунок 1 – Интерфейс функции авторизации

Программный код для разделения прав пользователей:

switch (user\_role)

{

case "1": db.CloseConnection(); this.Visible = false; Administrator a = new Administrator(id, fio, login, password, user\_role); a.Show(); break;

case "2": db.CloseConnection(); this.Visible = false; Seller s = new Seller(id, fio, login, password, user\_role); s.Show(); break;

}

1. В системе должна быть функция, обеспечивающая добавление, изменение и удаление товаров магазина

Интерфейс ИС «Pro-complect» для добавления, удаления и изменения товара (рис. 2-5).

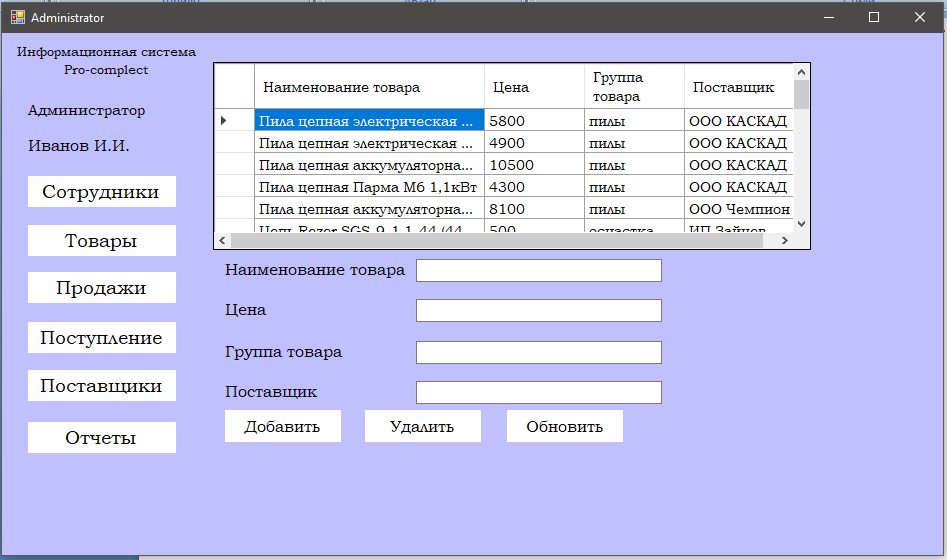


Рисунок 2 – Интерфейс для радела «Товары»

Для добавления и удаления товара необходимо ввести все значения товара. Для удаления нужно щёлкнуть по строке, которую нужно удалить. Для изменения необходимо щёлкнуть по нужному значению из таблицы и данные переносятся в поля.

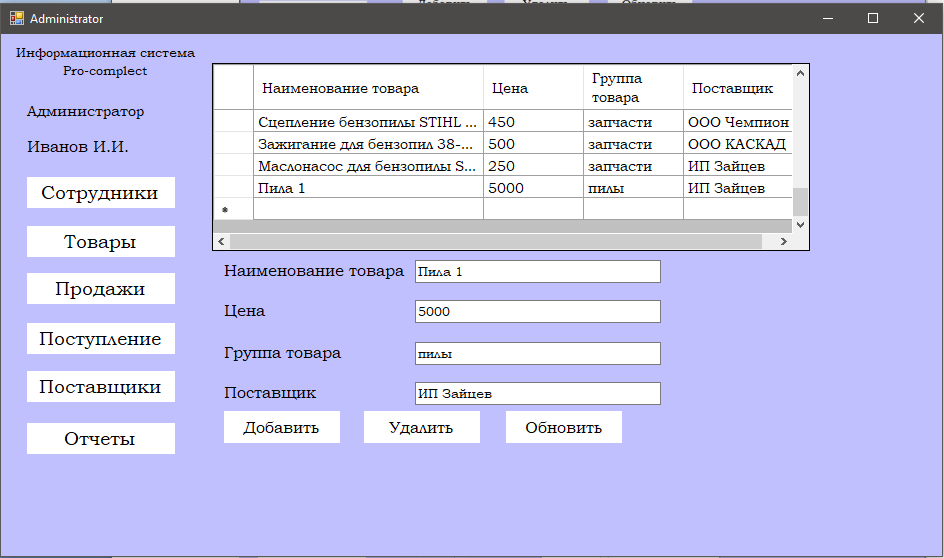


Рисунок 3 – Интерфейс для функции добавления товара

Для того чтобы добавлять товар с наименованиями, а не с идентификаторами нужно написать процедуру добавления товара, а затем программный код.

SQL-код для добавления товара:

PROCEDURE `insert\_tovar`(t\_n VARCHAR(100), pr int, na\_gr\_tov VARCHAR(45), na\_prov VARCHAR(45))

BEGIN

DECLARE id\_gruppa\_tov int;

DECLARE id\_prov int;

SELECT gruppa\_tovar.id\_gruppa\_tovar into id\_gruppa\_tov FROM gruppa\_tovar WHERE na\_gr\_tov = gruppa\_tovar\_name;

SELECT provider.id\_provider into id\_prov FROM provider WHERE na\_prov = provider\_name;

INSERT INTO tovar(tovar\_name, price, gruppa\_tovar\_id, provider\_id)

VALUES

(t\_n, pr, id\_gruppa\_tov, id\_prov);

END

Программный код для добавления товара:

db.OpenConnection();

string procedure\_name = "insert\_tovar";

MySqlCommand comm\_Add = new MySqlCommand(procedure\_name, db.GetConnection());

comm\_Add.CommandType = CommandType.StoredProcedure;

MySqlParameter t\_n\_param = new MySqlParameter

{

ParameterName = "t\_n",

Value = textBox\_tovar\_name.Text

};

MySqlParameter pr\_param = new MySqlParameter

{

ParameterName = "pr",

Value = Convert.ToInt32(textBox\_price.Text)//цена

};

MySqlParameter na\_gr\_tov\_param = new MySqlParameter

{

ParameterName = "na\_gr\_tov",

Value = textBox\_gruppa\_tovar\_id.Text

};

MySqlParameter na\_prov\_param = new MySqlParameter

{

ParameterName = "na\_prov",

Value = textBox\_provider\_id.Text

};

comm\_Add.Parameters.Add(t\_n\_param);

comm\_Add.Parameters.Add(pr\_param);

comm\_Add.Parameters.Add(na\_gr\_tov\_param);

comm\_Add.Parameters.Add(na\_prov\_param);

comm\_Add.ExecuteNonQuery();

db.CloseConnection();

ShowDataTovar();

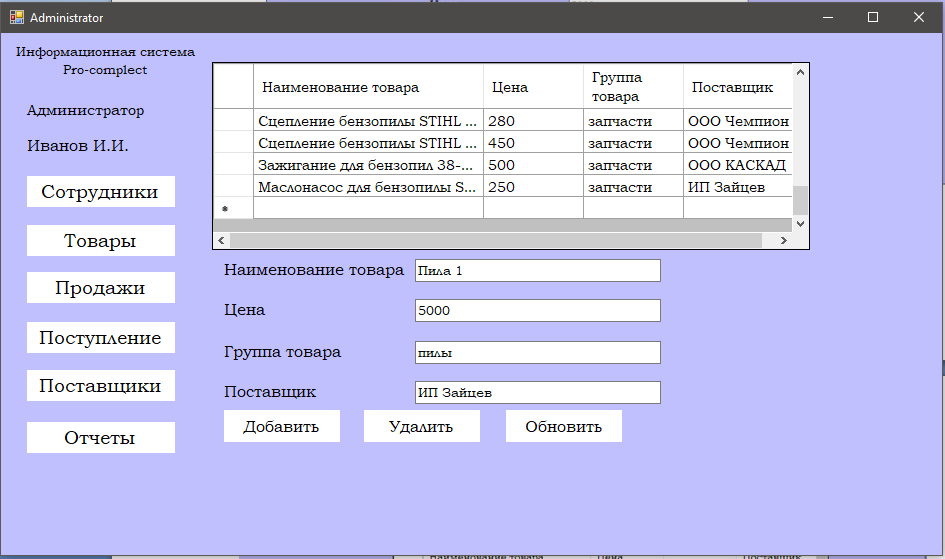


Рисунок 4 – Интерфейс для функции удаления товара

Для того чтобы удалять товар с наименованиями, а не с идентификаторами нужно написать процедуру удаления товара, а затем программный код.

SQL-код для удаления товара:

PROCEDURE `delete\_tovar`(t\_n VARCHAR(100), pr int, na\_gr\_tov VARCHAR(45), na\_prov VARCHAR(45))

BEGIN

DECLARE id\_grt int;

DECLARE id\_pro int;

SELECT gruppa\_tovar.id\_gruppa\_tovar into id\_grt FROM gruppa\_tovar WHERE na\_gr\_tov = gruppa\_tovar\_name;

SELECT provider.id\_provider into id\_pro FROM provider WHERE na\_prov = provider\_name;

DELETE FROM tovar WHERE t\_n = tovar\_name AND

pr = price AND

id\_grt = gruppa\_tovar\_id AND

id\_pro = provider\_id;

END

Программный код для удаления товара:

db.OpenConnection();

string procedure\_name = "delete\_tovar";

MySqlCommand comm\_Del = new MySqlCommand(procedure\_name, db.GetConnection());

comm\_Del.CommandType = CommandType.StoredProcedure;

MySqlParameter t\_n\_param = new MySqlParameter

{

ParameterName = "t\_n",

Value = textBox\_tovar\_name.Text

};

MySqlParameter p\_param = new MySqlParameter

{

ParameterName = "pr",

Value = Convert.ToInt32(textBox\_price.Text)//цена

};

MySqlParameter na\_gr\_tov\_param = new MySqlParameter

{

ParameterName = "na\_gr\_tov",

Value = textBox\_gruppa\_tovar\_id.Text

};

MySqlParameter na\_prov\_param = new MySqlParameter

{

ParameterName = "na\_prov",

Value = textBox\_provider\_id.Text

};

comm\_Del.Parameters.Add(t\_n\_param);

comm\_Del.Parameters.Add(p\_param);

comm\_Del.Parameters.Add(na\_gr\_tov\_param);

comm\_Del.Parameters.Add(na\_prov\_param);

comm\_Del.ExecuteNonQuery();

db.CloseConnection();

ShowDataTovar();

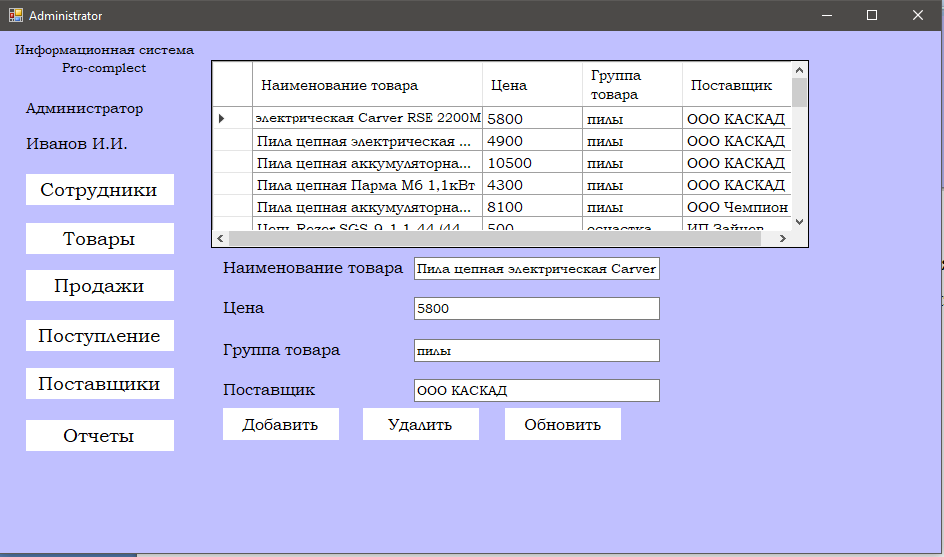


Рисунок 5 – Интерфейс для функции изменения товара

Для того чтобы изменить товар с наименованиями, а не с идентификаторами нужно написать процедуру изменения товара, а затем программный код.

SQL-код для изменения товара:

PROCEDURE `update\_tovar`(id INT, new\_t\_n VARCHAR(45), new\_p INT, new\_g\_t\_i VARCHAR(45), new\_p\_i VARCHAR(45))

BEGIN

DECLARE new\_id\_gruppa\_tov int;

DECLARE new\_id\_prov int;

SELECT gruppa\_tovar.id\_gruppa\_tovar into new\_id\_gruppa\_tov FROM gruppa\_tovar WHERE gruppa\_tovar\_name = new\_g\_t\_i;

SELECT provider.id\_provider into new\_id\_prov FROM provider WHERE provider\_name = new\_p\_i;

UPDATE tovar

SET tovar\_name = new\_t\_n,

price = new\_p,

gruppa\_tovar\_id = new\_id\_gruppa\_tov,

provider\_id = new\_id\_prov

WHERE id\_tovar = id;

END

Программный код для изменения товара:

db.OpenConnection();

string procedure\_name = "update\_tovar";

MySqlCommand comm\_Upd = new MySqlCommand(procedure\_name, db.GetConnection());

comm\_Upd.CommandType = CommandType.StoredProcedure;

string value = dataGridView1.CurrentRow.Cells[0].Value.ToString();

MySqlParameter id\_param = new MySqlParameter

{

ParameterName = "id",

Value = value

};

MySqlParameter new\_t\_n\_param = new MySqlParameter

{

ParameterName = "new\_t\_n",

Value = textBox\_tovar\_name.Text

};

MySqlParameter new\_p\_param = new MySqlParameter

{

ParameterName = "new\_p",

Value = Convert.ToInt32(textBox\_price.Text)//цена

};

MySqlParameter new\_g\_t\_i\_param = new MySqlParameter

{

ParameterName = "new\_g\_t\_i",

Value = textBox\_gruppa\_tovar\_id.Text

};

MySqlParameter new\_p\_i\_param = new MySqlParameter

{

ParameterName = "new\_p\_i",

Value = textBox\_provider\_id.Text

};

comm\_Upd.Parameters.Add(id\_param);

comm\_Upd.Parameters.Add(new\_t\_n\_param);

comm\_Upd.Parameters.Add(new\_p\_param);

comm\_Upd.Parameters.Add(new\_g\_t\_i\_param);

comm\_Upd.Parameters.Add(new\_p\_i\_param);

comm\_Upd.ExecuteNonQuery();

db.CloseConnection();

ShowDataTovar();

1. В системе должна быть функция, обеспечивающая добавление, изменение и удаление информации о поставщиках.

Интерфейс ИС «Pro-complect» для добавления, удаления и изменения поставщиков (рис. 6-9).

Для добавления и удаления поставщика необходимо ввести все значения поставщика. Для удаления нужно щёлкнуть по строке, которую нужно удалить. Для изменения необходимо щёлкнуть по нужному значению из таблицы и данные переносятся в поля.

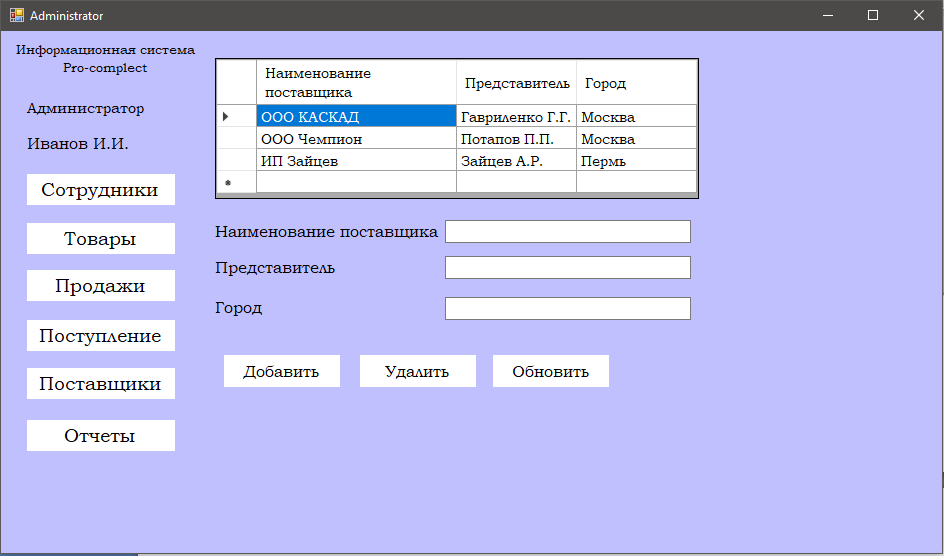


Рисунок 6 – Интерфейс для радела «Поставщики»

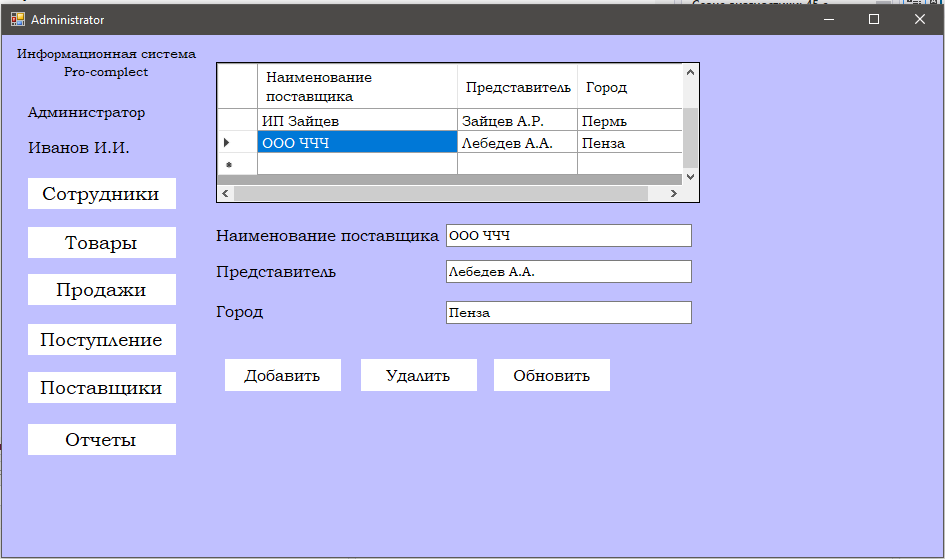


Рисунок 7 – Интерфейс для функции добавления поставщика

Программный код для добавления поставщика:

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "INSERT INTO provider (provider\_name, predstavitel, city) VALUES ('" + textBox\_provider\_name.Text + "', '"+ textBox\_predstavitel.Text + "', '"+ textBox\_city.Text + "');";

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

db.CloseConnection();

ShowProvider();

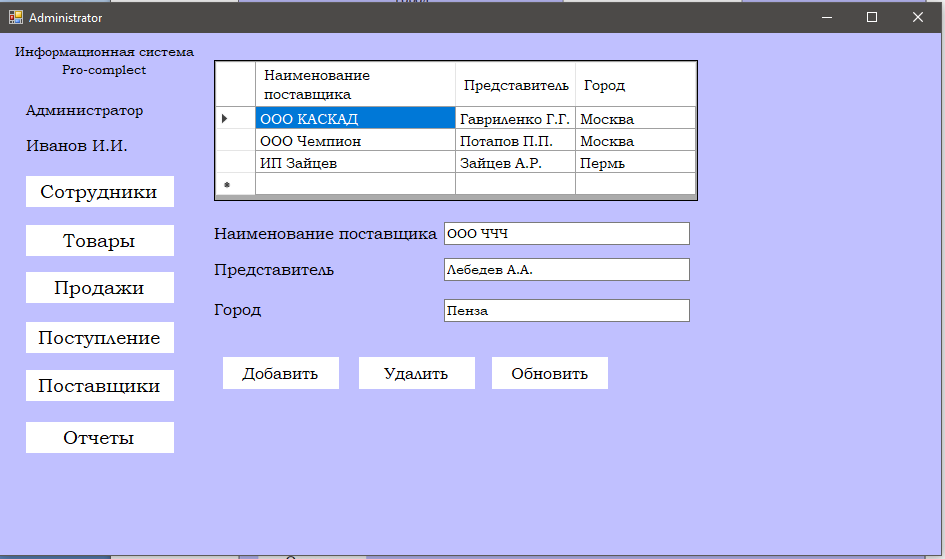


Рисунок 8 – Интерфейс для функции удаления поставщика

Программный код для удаления поставщика:

string value = dataGridView6.CurrentRow.Cells[0].Value.ToString();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "DELETE FROM provider WHERE id\_provider = " + value;

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

db.CloseConnection();

ShowProvider();

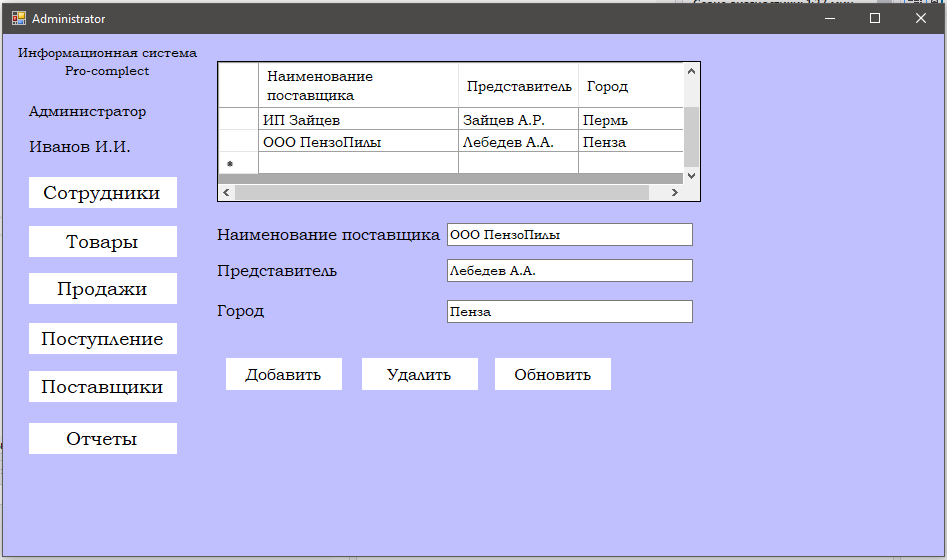


Рисунок 9 – Интерфейс для функции изменения поставщика

Программный код для изменения поставщика:

string value = dataGridView6.CurrentRow.Cells[0].Value.ToString();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "UPDATE provider SET provider\_name = '" + textBox\_provider\_name.Text + "', predstavitel = '" + textBox\_predstavitel.Text + "', city = '" + textBox\_city.Text + "' WHERE id\_provider = " + value;

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

db.CloseConnection();

ShowProvider();

1. В системе должна быть функция, предоставляющая сведения о цене товаров в магазине.

В разделе «Товары» в таблице представлена цена каждого товара (рис.10), а также наименование, группа товара и поставщик.

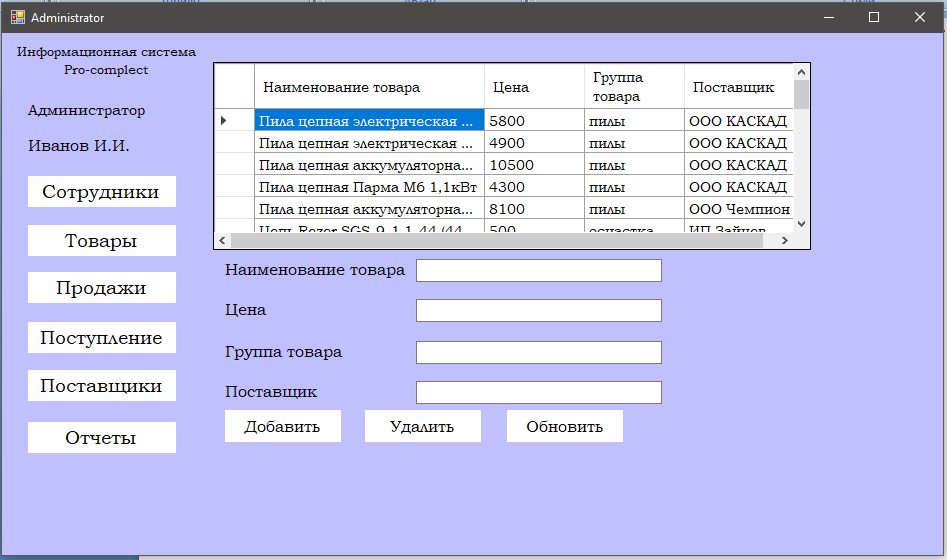


Рисунок 10 – Интерфейс для радела «Товары»

Программный код для сведений о цене товаров:

dataGridView1.Rows.Clear();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "SELECT id\_tovar, tovar\_name, price, (SELECT gruppa\_tovar\_name FROM gruppa\_tovar WHERE gruppa\_tovar.id\_gruppa\_tovar = tovar.gruppa\_tovar\_id), (SELECT provider\_name FROM provider WHERE provider.id\_provider = tovar.provider\_id) FROM tovar;";

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[5]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

1. В системе должна быть функция, предоставляющая сведения о поставщиках.

Интерфейс магазина «Pro-complect» раздела «Поставщики» изображен на рисунке 11. В таблице указаны наименования поставщиков, их представитель и местонахождение организации, а именно город.

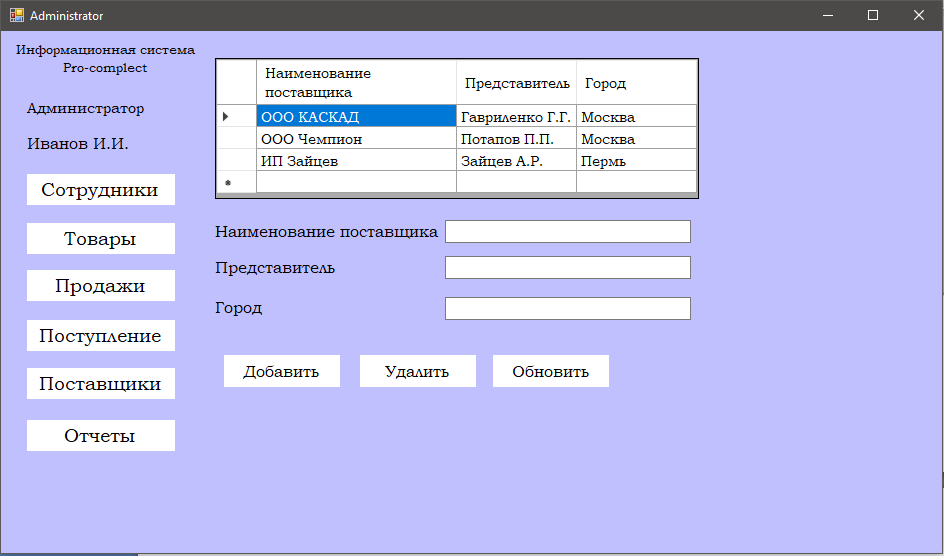


Рисунок 11 – Интерфейс для радела «Поставщики»

Программный код для сведений о поставщиках:

dataGridView6.Rows.Clear();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "SELECT \* FROM provider;";

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[4]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView6.Rows.Add(s);

1. В системе должна быть функция добавления, удаления и изменения Пользователя - сотрудников системы.
2. В системе должна быть функция, предоставляющая сведения о сотрудниках.

Интерфейс магазина «Pro-complect» раздела «Сотрудники» представлен на рисунке 12. Реализация функций добавления, удаления и изменения пользователя изображена на рисунках 13-15. В таблице можно увидеть ФИО, логин, пароль и роль пользователей. А также рекомендации, как заполнять поле ФИО.

Для добавления и удаления поставщика необходимо ввести все значения поставщика. Для удаления нужно щёлкнуть по строке, которую нужно удалить. Для изменения необходимо щёлкнуть по нужному значению из таблицы и данные переносятся в поля.

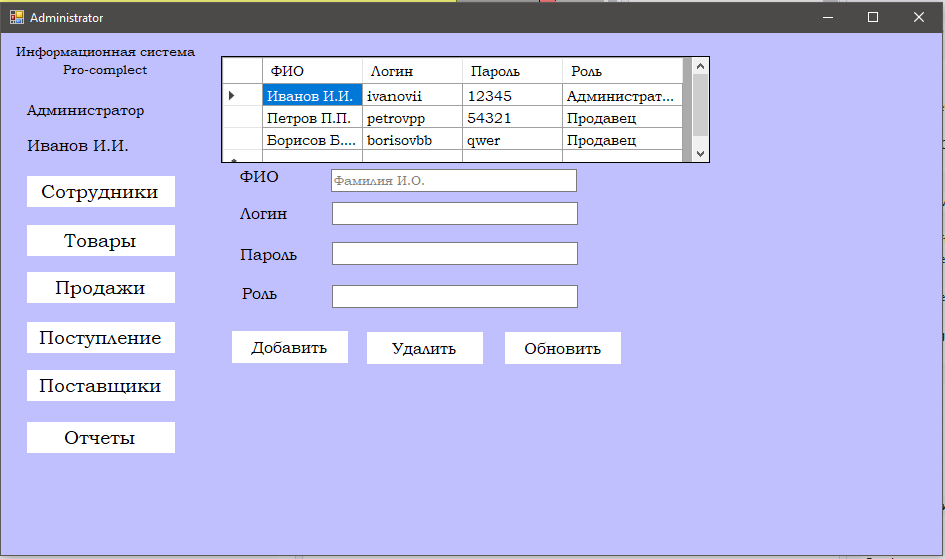


Рисунок 11 – Интерфейс для радела «Сотрудники»

Программный код о сведениях сотрудниках:

dataGridView4.Rows.Clear();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "SELECT id\_users, FIO, login, password, (SELECT user\_role\_name FROM user\_role WHERE user\_role.id\_user\_role = users.user\_role\_id) FROM users;";

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[5]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView4.Rows.Add(s);

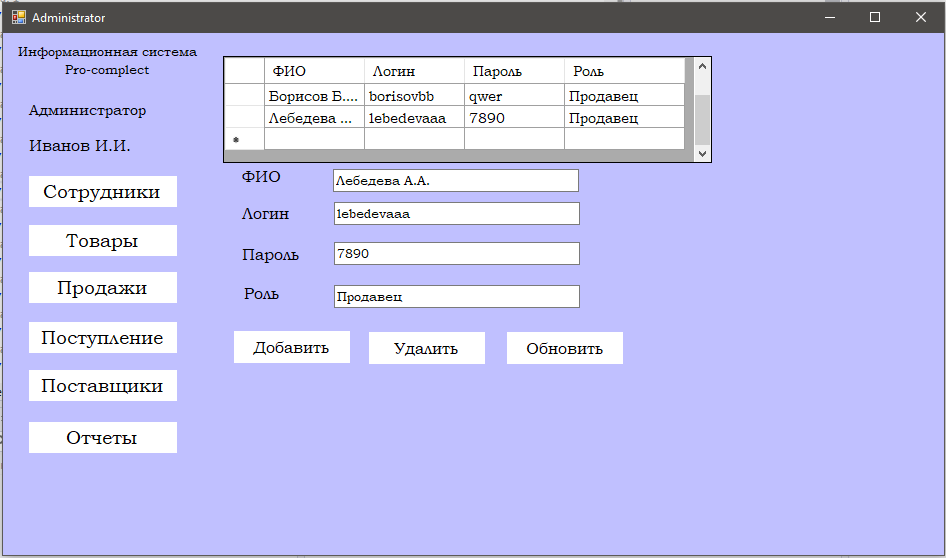


Рисунок 12 – Интерфейс для функции добавления сотрудника

Для того чтобы добавить сотрудника с наименованиями, а не с идентификаторами нужно написать процедуру добавления сотрудника, а затем программный код.

SQL-код для добавления сотрудника:

PROCEDURE `insert\_users`(f\_i\_o VARCHAR(45), log VARCHAR(45), pass VARCHAR(45), na\_us\_ro VARCHAR(45))

BEGIN

DECLARE id\_use\_rol int;

SELECT user\_role.id\_user\_role into id\_use\_rol FROM user\_role WHERE na\_us\_ro = user\_role\_name;

INSERT INTO users(FIO, login, password, user\_role\_id)

VALUES

(f\_i\_o, log, pass, id\_use\_rol);

END

Программный код для добавления сотрудника:

db.OpenConnection();

string procedure\_name = "insert\_users";

MySqlCommand comm\_Add = new MySqlCommand(procedure\_name, db.GetConnection());

comm\_Add.CommandType = CommandType.StoredProcedure;

MySqlParameter f\_i\_o\_param = new MySqlParameter

{

ParameterName = "f\_i\_o",

Value = textBoxFIO.Text

};

MySqlParameter log\_param = new MySqlParameter

{

ParameterName = "log",

Value = textBoxLogin.Text

};

MySqlParameter pass\_param = new MySqlParameter

{

ParameterName = "pass",

Value = textBoxPassword.Text

};

MySqlParameter na\_us\_ro\_param = new MySqlParameter

{

ParameterName = "na\_us\_ro",

Value = textBoxUserRole.Text

};

comm\_Add.Parameters.Add(f\_i\_o\_param);

comm\_Add.Parameters.Add(log\_param);

comm\_Add.Parameters.Add(pass\_param);

comm\_Add.Parameters.Add(na\_us\_ro\_param);

comm\_Add.ExecuteNonQuery();

db.CloseConnection();

ShowDataUsers();

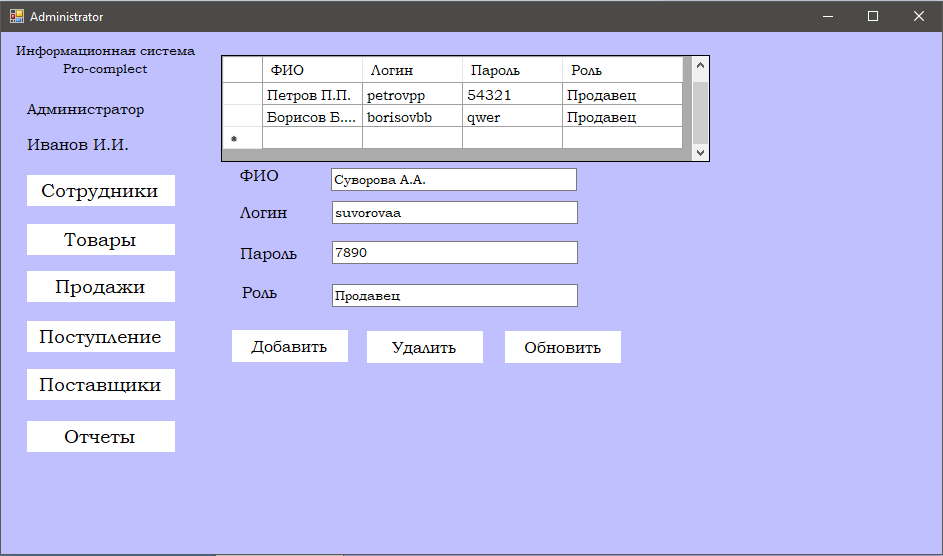


Рисунок 13 – Интерфейс для функции удаления сотрудника

Для того чтобы удалить сотрудника с наименованиями, а не с идентификаторами нужно написать процедуру удаления сотрудника, а затем программный код.

SQL-код для удаления сотрудника:

PROCEDURE `delete\_users`(f\_i\_o VARCHAR(45), log VARCHAR(45), pass VARCHAR(45), na\_us\_ro VARCHAR(45))

BEGIN

DECLARE id\_use\_rol int;

SELECT user\_role.id\_user\_role into id\_use\_rol FROM user\_role WHERE na\_us\_ro = user\_role\_name;

DELETE FROM users WHERE f\_i\_o = FIO AND

log = login AND

pass = password AND

id\_use\_rol = user\_role\_id;

END

Программный код для удаления сотрудника:

db.OpenConnection();

string procedure\_name = "delete\_users";

MySqlCommand comm\_Del = new MySqlCommand(procedure\_name, db.GetConnection());

comm\_Del.CommandType = CommandType.StoredProcedure;

MySqlParameter f\_i\_o\_param = new MySqlParameter

{

ParameterName = "f\_i\_o",

Value = textBoxFIO.Text

};

MySqlParameter log\_param = new MySqlParameter

{

ParameterName = "log",

Value = textBoxLogin.Text

};

MySqlParameter pass\_param = new MySqlParameter

{

ParameterName = "pass",

Value = textBoxPassword.Text

};

MySqlParameter na\_us\_ro\_param = new MySqlParameter

{

ParameterName = "na\_us\_ro",

Value = textBoxUserRole.Text

};

comm\_Del.Parameters.Add(f\_i\_o\_param);

comm\_Del.Parameters.Add(log\_param);

comm\_Del.Parameters.Add(pass\_param);

comm\_Del.Parameters.Add(na\_us\_ro\_param);

comm\_Del.ExecuteNonQuery();

db.CloseConnection();

ShowDataUsers();

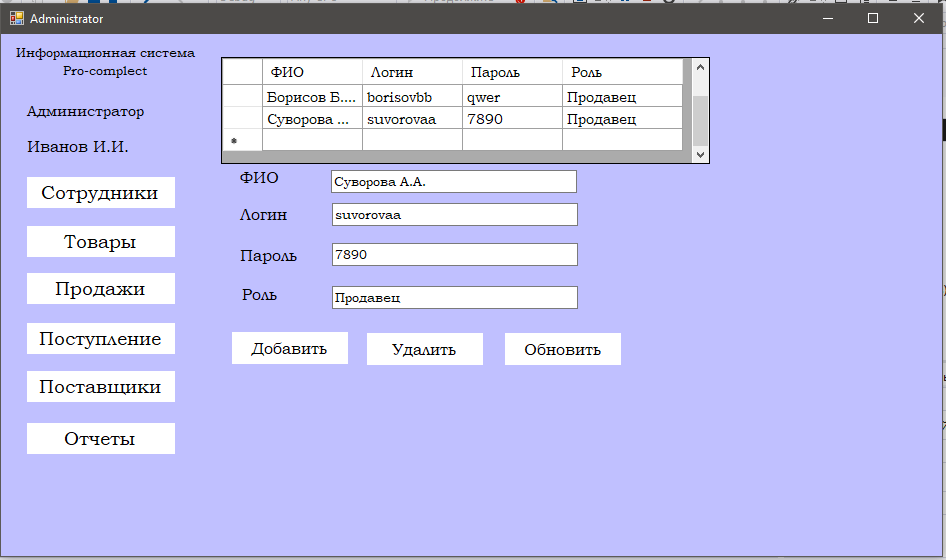


Рисунок 14 – Интерфейс для функции изменения сотрудника

Для того чтобы изменить сотрудника с наименованиями, а не с идентификаторами нужно написать процедуру изменения сотрудника, а затем программный код.

SQL-код для изменения сотрудника:

PROCEDURE `update\_users`(id INT, new\_f\_i\_o VARCHAR(45), new\_log VARCHAR(45), new\_pass VARCHAR(45), new\_na\_us\_ro VARCHAR(45))

BEGIN

DECLARE new\_id\_use\_rol int;

SELECT user\_role.id\_user\_role into new\_id\_use\_rol FROM user\_role WHERE user\_role\_name = new\_na\_us\_ro;

UPDATE users

SET FIO = new\_f\_i\_o,

login = new\_log,

password = new\_pass,

user\_role\_id = new\_id\_use\_rol

WHERE id\_users = id;

END

Программный код для изменения сотрудника:

db.OpenConnection();

string procedure\_name = "update\_users";

MySqlCommand comm\_Upd = new MySqlCommand(procedure\_name, db.GetConnection());

comm\_Upd.CommandType = CommandType.StoredProcedure;

string value = dataGridView4.CurrentRow.Cells[0].Value.ToString();

MySqlParameter id\_param = new MySqlParameter

{

ParameterName = "id",

Value = value

};

MySqlParameter f\_i\_o\_param = new MySqlParameter

{

ParameterName = "new\_f\_i\_o",

Value = textBoxFIO.Text

};

MySqlParameter log\_param = new MySqlParameter

{

ParameterName = "new\_log",

Value = textBoxLogin.Text

};

MySqlParameter pass\_param = new MySqlParameter

{

ParameterName = "new\_pass",

Value = textBoxPassword.Text

};

MySqlParameter na\_us\_ro\_param = new MySqlParameter

{

ParameterName = "new\_na\_us\_ro",

Value = textBoxUserRole.Text

};

comm\_Upd.Parameters.Add(id\_param);

comm\_Upd.Parameters.Add(f\_i\_o\_param);

comm\_Upd.Parameters.Add(log\_param);

comm\_Upd.Parameters.Add(pass\_param);

comm\_Upd.Parameters.Add(na\_us\_ro\_param);

comm\_Upd.ExecuteNonQuery();

db.CloseConnection();

ShowDataUsers();

1. В системе должна быть функция, обеспечивающая добавление, изменение и удаление о поступающих товарах.
2. Функция, предоставляющая сведения о поступлениях.

Интерфейс магазина «Pro-complect» раздела «Поступление» представлен на рисунке 15. В нем реализованы функции добавления, удаления и изменения пользователя. В таблице можно увидеть дату поступления, количество и наименование. А также рекомендации, как заполнять поле даты – «гггг-мм-дд».

Для добавления и удаления поступления товара необходимо ввести все значения. Для удаления нужно щёлкнуть по строке, которую нужно удалить. Для изменения необходимо щёлкнуть по нужному значению из таблицы и данные переносятся в поля.

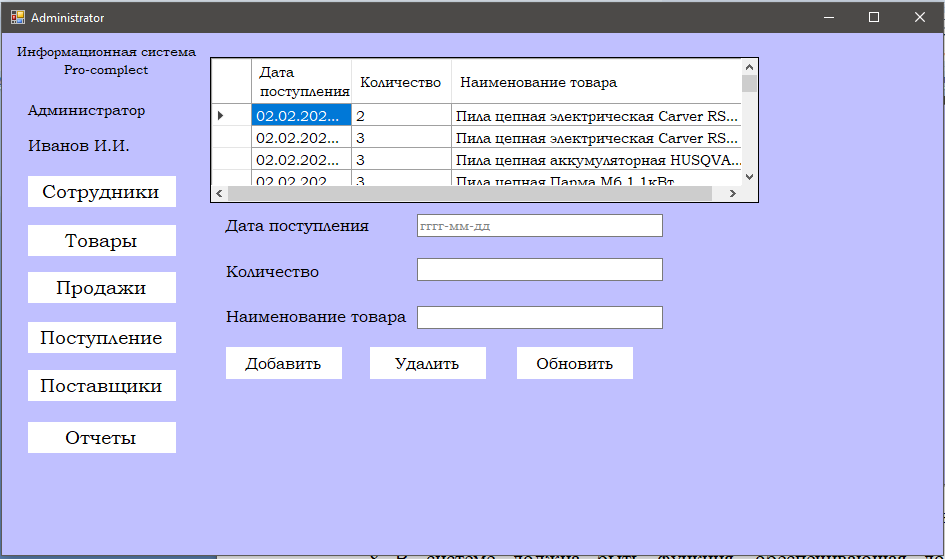


Рисунок 15 – Интерфейс для раздела «Поступление»

Программный код о сведениях поступления:

dataGridView3.Rows.Clear();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "SELECT id\_prihod, prihod\_date, prihod\_colvo, (SELECT tovar\_name FROM tovar WHERE tovar.id\_tovar = prihod.tovar\_id) FROM prihod; ";

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[4]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView3.Rows.Add(s);

1. Функция, обеспечивающая добавление, изменение и удаление о проданных товарах.
2. Функция, предоставляющая сведения о продажах.

Интерфейс магазина «Pro-complect» раздела «Продажи» представлен на рисунке 16. В нем реализованы функции добавления, удаления и изменения пользователя. В таблице можно увидеть дату продажи, количество и наименование и сотрудника. А также рекомендации, как заполнять поле даты – «гггг-мм-дд», и ФИО – «Фамилия И.О.»

Для добавления и удаления поступления товара необходимо ввести все значения. Для удаления нужно щёлкнуть по строке, которую нужно удалить. Для изменения необходимо щёлкнуть по нужному значению из таблицы и данные переносятся в поля.

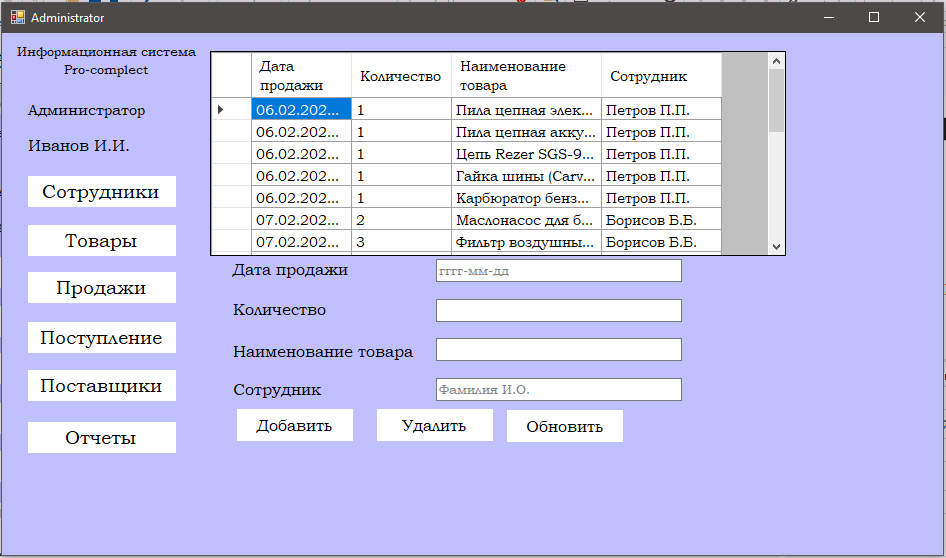


Рисунок 16 – Интерфейс для раздела «Продажи»

Программный код о сведениях продаж:

dataGridView2.Rows.Clear();

MySqlDataAdapter msda = new MySqlDataAdapter();

string sql = "SELECT id\_rashod, rashod\_date, rashod\_colvo, (SELECT tovar\_name FROM tovar WHERE tovar.id\_tovar = rashod.tovar\_id), (SELECT FIO FROM users WHERE users.id\_users = rashod.users\_id) FROM rashod; ";

DataTable dt = new DataTable();

db.OpenConnection();

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[5]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView2.Rows.Add(s);

1. Функция генерации отчетов – не менее пяти.

В системе присутствует 5 видов отчета (рис.17-). Отчеты формируются на основе веденных данных – периода, за который требуется получить отчет (в формате гггг-мм-дд). Для последнего отчета дополнительным вводимым данным должно быть сотрудник ФИО сотрудника.



Рисунок 17 – Интерфейс для раздела «Отчеты»

Программный код генерации отчетов:

db.OpenConnection();

if (comboBox1.Text == "По продажам товара за период")

{

dataGridView5.Rows.Clear();

dataGridView5.ColumnCount = 5;

dataGridView5.Columns[0].HeaderText = "Наименование товара";

dataGridView5.Columns[1].HeaderText = "Количество";

dataGridView5.Columns[2].HeaderText = "Цена";

dataGridView5.Columns[3].HeaderText = "Сумма";

dataGridView5.Columns[4].HeaderText = "Дата продажи";

string sql1 = "SELECT t.tovar\_name, r.rashod\_colvo, t.price, (r.rashod\_colvo \* t.price), r.rashod\_date FROM (titan.rashod r JOIN titan.tovar t) WHERE ((r.rashod\_date >= '" + textBox\_date\_ot.Text + "') and(r.rashod\_date <= '" + textBox\_date\_do.Text + "') and (r.tovar\_id = t.id\_tovar));";

MySqlCommand command1 = new MySqlCommand(sql1, db.GetConnection());

MySqlDataReader reader1 = command1.ExecuteReader();

List<string[]> data1 = new List<string[]>();

while (reader1.Read())

{

data1.Add(new string[5]);

data1[data1.Count - 1][0] = reader1[0].ToString();

data1[data1.Count - 1][1] = reader1[1].ToString();

data1[data1.Count - 1][2] = reader1[2].ToString();

data1[data1.Count - 1][3] = reader1[3].ToString();

data1[data1.Count - 1][4] = reader1[4].ToString();

}

reader1.Close();

db.CloseConnection();

foreach (string[] s in data1)

dataGridView5.Rows.Add(s);

}

else if (comboBox1.Text == "По остаткам товара за период")

{

dataGridView5.Rows.Clear();

dataGridView5.ColumnCount = 3;

dataGridView5.Columns[0].HeaderText = "Наименование товара";

dataGridView5.Columns[1].HeaderText = "Цена";

dataGridView5.Columns[2].HeaderText = "Количество";

string sql2 = "SELECT t.tovar\_name, t.price, (p.prihod\_colvo - r.rashod\_colvo) FROM((titan.rashod r JOIN titan.tovar t) JOIN titan.prihod p) WHERE((r.rashod\_date >= '" + textBox\_date\_ot.Text + "') and (r.rashod\_date <= '" + textBox\_date\_do.Text + "') and(r.tovar\_id = t.id\_tovar) and(t.id\_tovar = p.tovar\_id));";

MySqlCommand command2 = new MySqlCommand(sql2, db.GetConnection());

MySqlDataReader reader2 = command2.ExecuteReader();

List<string[]> data2 = new List<string[]>();

while (reader2.Read())

{

data2.Add(new string[5]);

data2[data2.Count - 1][0] = reader2[0].ToString();

data2[data2.Count - 1][1] = reader2[1].ToString();

data2[data2.Count - 1][2] = reader2[2].ToString();

}

reader2.Close();

db.CloseConnection();

foreach (string[] s in data2)

dataGridView5.Rows.Add(s);

}

else if (comboBox1.Text == "О заказанных у поставщиков товара за период")

{

dataGridView5.Rows.Clear();

dataGridView5.ColumnCount = 5;

dataGridView5.Columns[0].HeaderText = "Наименование товара";

dataGridView5.Columns[1].HeaderText = "Количество";

dataGridView5.Columns[2].HeaderText = "Цена";

dataGridView5.Columns[3].HeaderText = "Поставщик";

dataGridView5.Columns[4].HeaderText = "Дата продажи";

string sql1 = "SELECT t.tovar\_name, p.prihod\_colvo, t.price, pr.provider\_name, p.prihod\_date FROM ((titan.prihod p JOIN titan.tovar t) JOIN titan.provider pr) WHERE ((p.prihod\_date >= '" + textBox\_date\_ot.Text + "') and (p.prihod\_date <= '" + textBox\_date\_do.Text + "') and (pr.id\_provider = t.provider\_id) and (t.id\_tovar = p.tovar\_id));";

MySqlCommand command1 = new MySqlCommand(sql1, db.GetConnection());

MySqlDataReader reader1 = command1.ExecuteReader();

List<string[]> data1 = new List<string[]>();

while (reader1.Read())

{

data1.Add(new string[5]);

data1[data1.Count - 1][0] = reader1[0].ToString();

data1[data1.Count - 1][1] = reader1[1].ToString();

data1[data1.Count - 1][2] = reader1[2].ToString();

data1[data1.Count - 1][3] = reader1[3].ToString();

data1[data1.Count - 1][4] = reader1[4].ToString();

}

reader1.Close();

db.CloseConnection();

foreach (string[] s in data1)

dataGridView5.Rows.Add(s);

}

else if (comboBox1.Text == "По продажам сотрудников за период")

{

dataGridView5.Rows.Clear();

dataGridView5.ColumnCount = 5;

dataGridView5.Columns[0].HeaderText = "Сотрудник";

dataGridView5.Columns[1].HeaderText = "Наименование товара";

dataGridView5.Columns[2].HeaderText = "Количество";

dataGridView5.Columns[3].HeaderText = "Сумма";

dataGridView5.Columns[4].HeaderText = "Дата продажи";

string sql1 = "SELECT u.FIO, t.tovar\_name, r.rashod\_colvo, (r.rashod\_colvo \* t.price), r.rashod\_date FROM ((titan.users u JOIN titan.tovar t) JOIN titan.rashod r) WHERE ((r.rashod\_date >= '" + textBox\_date\_ot.Text + "') and (r.rashod\_date <= '" + textBox\_date\_do.Text + "') and (u.id\_users = r.users\_id) and (t.id\_tovar = r.tovar\_id));";

MySqlCommand command1 = new MySqlCommand(sql1, db.GetConnection());

MySqlDataReader reader1 = command1.ExecuteReader();

List<string[]> data1 = new List<string[]>();

while (reader1.Read())

{

data1.Add(new string[5]);

data1[data1.Count - 1][0] = reader1[0].ToString();

data1[data1.Count - 1][1] = reader1[1].ToString();

data1[data1.Count - 1][2] = reader1[2].ToString();

data1[data1.Count - 1][3] = reader1[3].ToString();

data1[data1.Count - 1][4] = reader1[4].ToString();

}

reader1.Close();

db.CloseConnection();

foreach (string[] s in data1)

dataGridView5.Rows.Add(s);

}

//comboBox1.Items.Add("По продажам по каждому сотруднику за период");

else

{

dataGridView5.Rows.Clear();

dataGridView5.ColumnCount = 4;

dataGridView5.Columns[0].HeaderText = "Наименование товара";

dataGridView5.Columns[1].HeaderText = "Количество";

dataGridView5.Columns[2].HeaderText = "Сумма";

dataGridView5.Columns[3].HeaderText = "Дата продажи";

string sql1 = "SELECT t.tovar\_name, r.rashod\_colvo, (r.rashod\_colvo \* t.price), r.rashod\_date FROM ((titan.users u JOIN titan.tovar t) JOIN titan.rashod r) WHERE ((r.rashod\_date >= '" + textBox\_date\_ot.Text + "') and (r.rashod\_date <= '" + textBox\_date\_do.Text + "') and (u.FIO = '" + comboBox2.Text + "') and (u.id\_users = r.users\_id) and (t.id\_tovar = r.tovar\_id));";

MySqlCommand command1 = new MySqlCommand(sql1, db.GetConnection());

MySqlDataReader reader1 = command1.ExecuteReader();

List<string[]> data1 = new List<string[]>();

while (reader1.Read())

{

data1.Add(new string[4]);

data1[data1.Count - 1][0] = reader1[0].ToString();

data1[data1.Count - 1][1] = reader1[1].ToString();

data1[data1.Count - 1][2] = reader1[2].ToString();

data1[data1.Count - 1][3] = reader1[3].ToString();

}

reader1.Close();

db.CloseConnection();

foreach (string[] s in data1)

dataGridView5.Rows.Add(s);

}

db.CloseConnection();

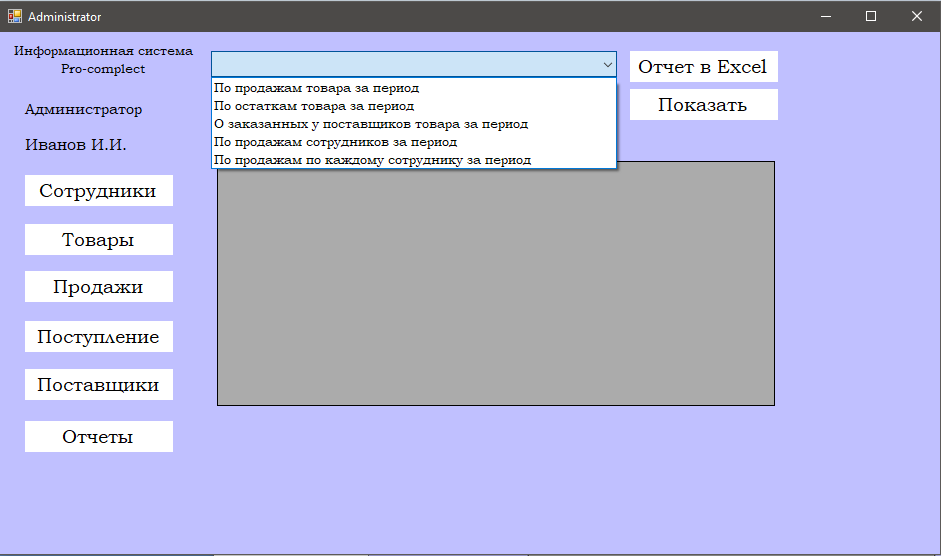


Рисунок 18 – Выбор отчета в разделе «Отчеты»

Отчет по продажам товара за период 2023-02-06 – 2023-02-07 представлен на рисунке 19.

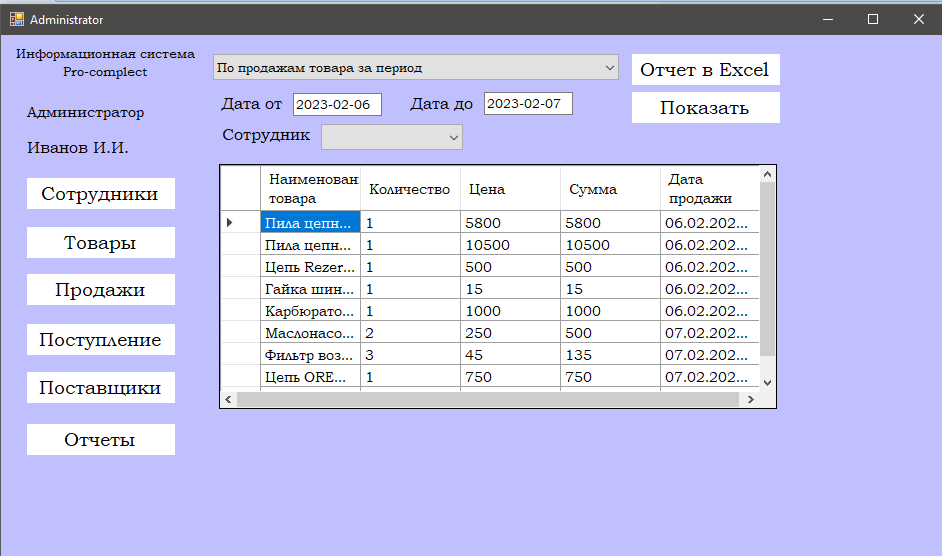


Рисунок 19 – Отчет по продажам товара за период

Отчет по остаткам товара за период 2023-02-06 – 2023-02-07 представлен на рисунке 20.

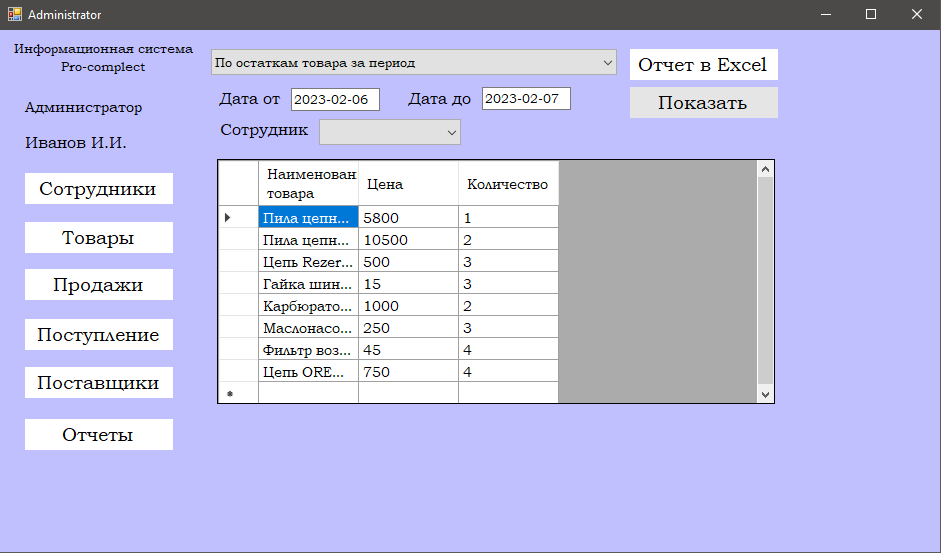


Рисунок 20 – Отчет по остаткам товара за период

Отчет о заказанных товаров у поставщиков за период 2023-02-02 – 2023-02-03 представлен на рисунке 21.

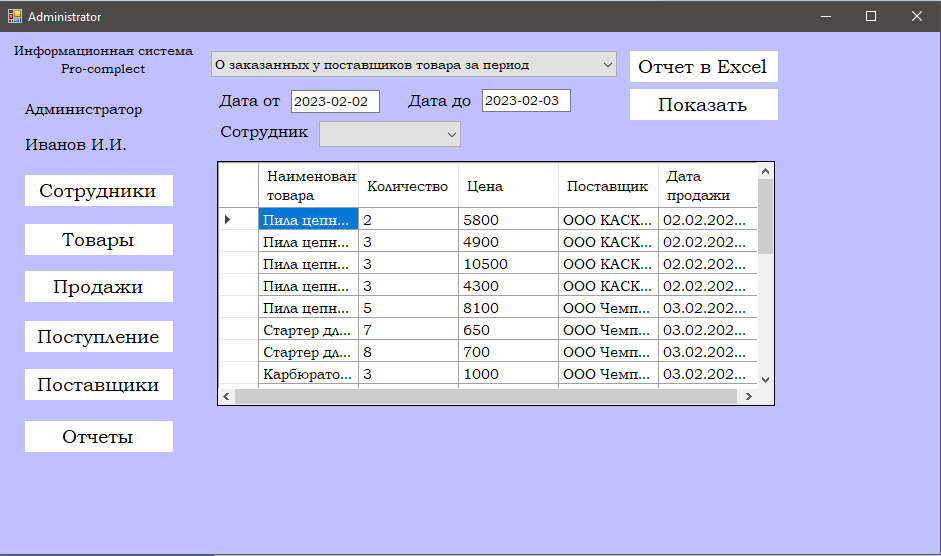


Рисунок 21 – Отчет о заказанных товаров у поставщиков за период

Отчет по продажам товаров сотрудников за период 2023-02-07 – 2023-02-08 представлен на рисунке 22.

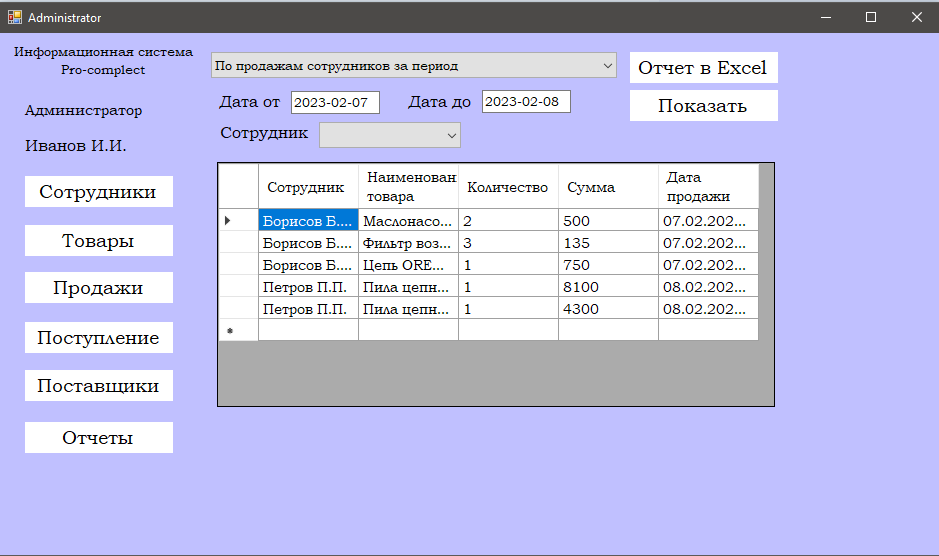


Рисунок 22 – Отчет по продажам товаров сотрудников за период

Отчет по продажам товаров сотрудников за период 2023-02-07 – 2023-02-08 представлен на рисунке 23. Для этого отчета дополнительным выбранным данным должно быть сотрудник ФИО сотрудника.

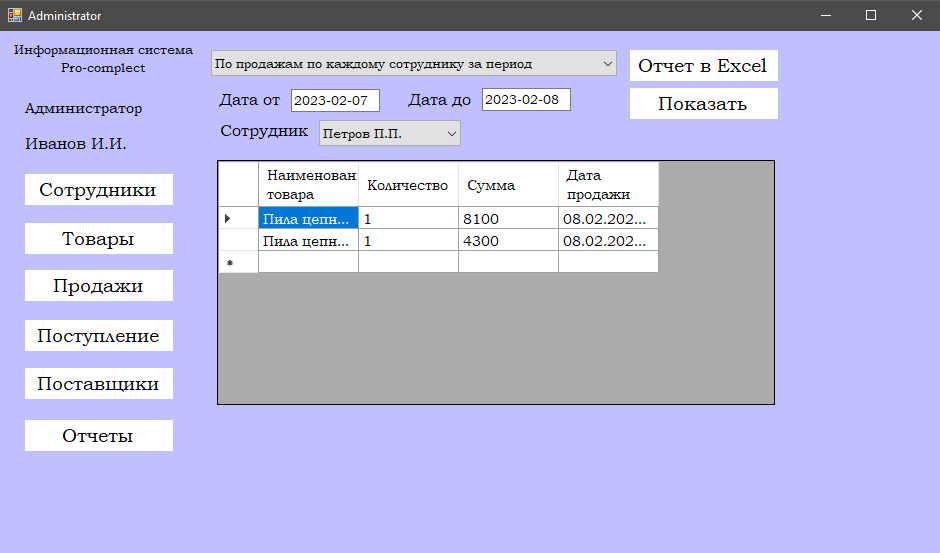
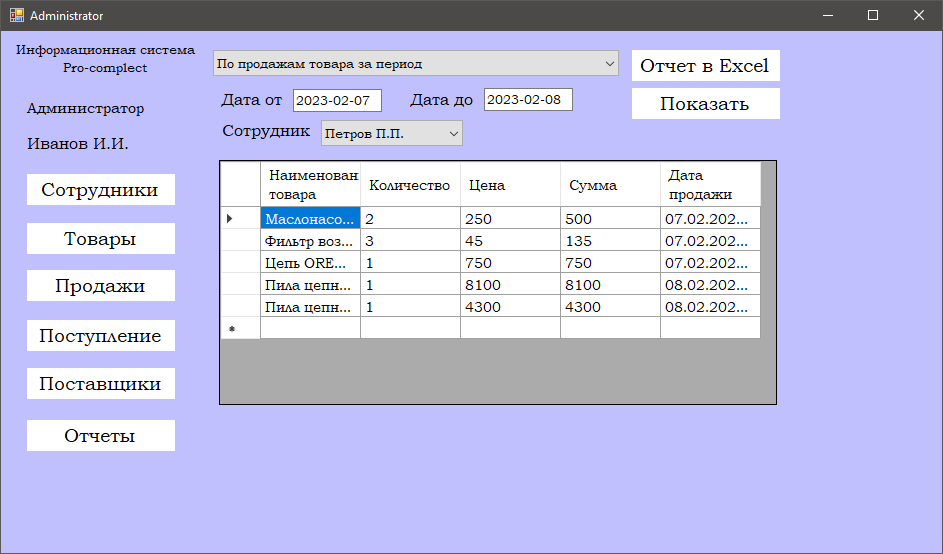


Рисунок 22 – Отчет по продажам товаров сотрудника за период

1. Формирование отчетов в формате Excel.

Все отчеты формируются в формате Excel. Для этого нужно выбрать название отчета, дату (если отчет по продажам товаров конкретного сотрудника за период, то выбрать сотрудника). Откроется путь, сформируется имя отчета: «Название отчета» + выбранный период (рис. 23-25).

Рисунок 23 – Отчет по продажам товаров за 2023-02-07 – 2023-02-08

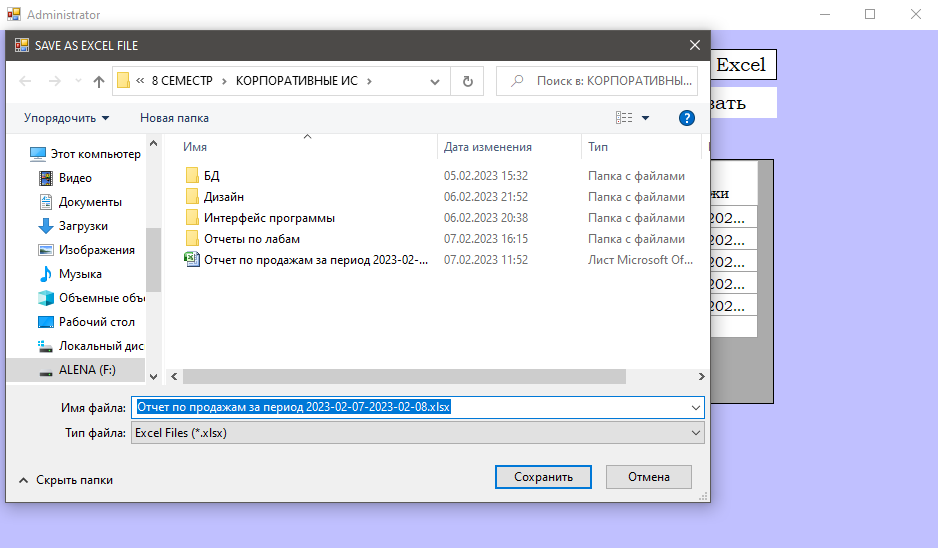


Рисунок 24 – Сохранение отчета



Рисунок 25 – Просмотр отчета в формате Excel

Программный код формирования отчета в Excel:

saveFileDialog1.InitialDirectory = "F:\\УНИВЕР\\4 КУРС\\8 СЕМЕСТР\\КОРПОРАТИВНЫЕ ИС";

saveFileDialog1.Title = "SAVE AS EXCEL FILE";

saveFileDialog1.Filter = "Excel Files (\*.xlsx)|\*.xlsx";

saveFileDialog1.FileName = file\_name;

if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)

{

Cursor.Current = Cursors.WaitCursor;

Excel.Application exApp = new Excel.Application();

exApp.Application.Workbooks.Add(Type.Missing);

exApp.Columns.ColumnWidth = 28;

for (int i = 1; i < dataGridView5.Columns.Count + 1; i++)

{

exApp.Cells[1, i] = dataGridView5.Columns[i - 1].HeaderText;

}

for (int i = 0; i < dataGridView5.Rows.Count; i++)

{

for (int j = 0; j < dataGridView5.Columns.Count; j++)

{

exApp.Cells[i + 2, j + 1] = dataGridView5.Rows[i].Cells[j].Value.ToString();

}

}

exApp.ActiveWorkbook.SaveAs(saveFileDialog1.FileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive, Type.Missing, Type.Missing, Type.Missing, Type.Missing);

exApp.ActiveWorkbook.Saved = true;

exApp.Quit();

}

Cursor.Current = Cursors.Default;

1. Реакция системы на ошибочный ввод.

При неправильном вводе выдается сообщение при аутентификации пользователя (рис.27), при не правильном типе данных выводится надпись (рис.28).

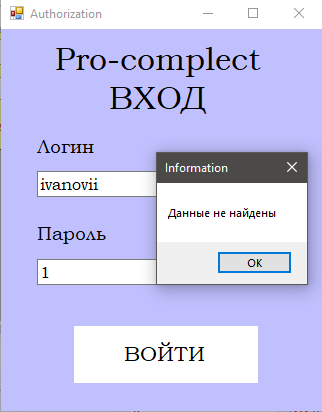


Рисунок 26 – Интерфейс реакция на ошибку в аунтетификации

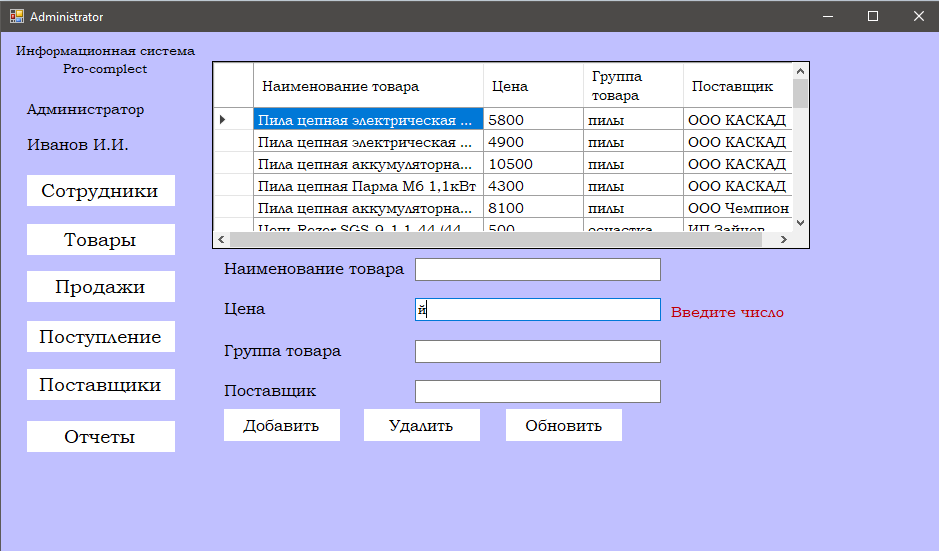


Рисунок 27 – Интерфейс реакция на ошибку в типе данных

Программный код реакции на ошибочный ввод:

char number = e.KeyChar;

if (Char.IsDigit(number))

{

//e.Handled = true;

label19.Visible = false;

}

else

{

label19.Visible = true;

}