

Настоящая предметная область представляет собой подсистему учета материалов предприятия, находящуюся в локальной сети данного предприятия.

Основные сущности: выдача товара, поступление товара, продукция, категория товара, поставщик.

Основными пользователями данной подсистемы являются администратор, заведующий складом, учетчики приема и выдачи продукции (отдельные отделы). Учетчики приема продукции заносят данные в базу с помощью сканера штрих-кода и вводят количество единиц продукции, учетчики выдачи при помощи каталога продукции получают данные о штрих-коде (номере) продукции и её наличии в табели рас, выдают продукцию, вводят данные с помощью штрих-кода и количество продукции, занося данные в табель оперативного расхода материала. Зав. склада выполняет функции контроля над поступлением и расходованием материала, поддерживает необходимое количество материала на складе путем вывода остатков материала, открывает и закрывает табель оперативного расхода материала. Администратор добавляет и удаляет категории материалов, создает табель оперативного расхода материала, контролирует работу заведующего складом путем просмотра табели рас.

Словарь предметной области.

*Персонал:*

Учетчик приема - персонал, занимающийся вводом поступлений продукции товара на склад.

Учетчик выдачи - персонал, занимающийся поиском товара в базе данных и выдачей продукции и введение табеля оперативного расхода материала.

Администратор – руководитель складом.

Зав склада - составитель списка наименований продукции.

*Документы:*

Табель рас – список оставшегося на складе товара.

Каталог продукции – список товара по категориям.

Табель оперативного расхода материала- список учета выдаваемой на складе продукции.

Табель ввода - журнал приема товара на склад.

Список товаров – общий сборник различных наименований продукции

*Рабочий материал:*

Продукция – товар, с которым работает персонал склада.

*Сущности:*

Продукция имеет:

-штрих-код;

-категорию;

-материал;

-массу;

-цена единицы;

-количество (в наличии).

Список товаров – общий сборник различных наименований продукции

Поступление товара – прием и регистрация прибывшего на склад товара.

Поступление имеет:

-номер;

-поступивший товар;

-дата;

-время;

-количество поступившего товара.

Табель ввода - журнал поступлений товара на склад.

Выдача товара – фиксация отданного товара на производство.

Выдача имеет:

-номер;

-поступивший товар;

-дата;

-время;

-количество поступившего товара.

Табель оперативного расхода материала- список учета выдаваемой на складе продукции.

Категория товара - группа схожих товаров.

Категория имеет:

-номер;

-название;

-количество товаров данной категории;

-дату создания;

Каталог продукции – список товара по категориям.

Заявка поставщику – перечень необходимых товаров для склада.

Заявка имеет:

-номер;

-товар;

-количество товара.

**Диаграмма прецедентов**



2 этап.

Диаграмма классов







Диаграмма состояний



**3 Этап**

**Web.config**

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-Курсовик-20181021041825.mdf;Initial Catalog=aspnet-Курсовик-20181021041825;Integrated Security=True"

providerName="System.Data.SqlClient" />

<add name="YourDB" connectionString="Server=.\sqlexpress;Database=Склад;Integrated Security=True"/>

</connectionStrings>

**Контроллеры**

**Homecontroller.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Курсовик.Controllers

{

public class HomeController : Controller

{

// GET: Home

public ActionResult Index()

{

AuthorizeAttribute Role = new AuthorizeAttribute();

if (User.IsInRole("Администратор"))

return RedirectToAction("Index", "Поставщик");

if (User.IsInRole("Заведующий складом"))

return RedirectToAction("Index", "Продукция");

if (User.IsInRole("Учетчик выдачи"))

return RedirectToAction("Index", "Категория");

if (User.IsInRole("Учетчик приема"))

return RedirectToAction("Index", "Поступление");

return RedirectToAction("Index", "Поставщик");

}

// GET: Home/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: Home/Create

public ActionResult Create()

{

return View();

}

// POST: Home/Create

[HttpPost]

public ActionResult Create(FormCollection collection)

{

try

{

// TODO: Add insert logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

// GET: Home/Edit/5

public ActionResult Edit(int id)

{

return View();

}

// POST: Home/Edit/5

[HttpPost]

public ActionResult Edit(int id, FormCollection collection)

{

try

{

// TODO: Add update logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

// GET: Home/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: Home/Delete/5

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

try

{

// TODO: Add delete logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

}

}

**ВыдачаController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Курсовик.Models;

using Курсовик.DAO;

namespace Курсовик.Controllers

{

public class ВыдачаController : Controller

{

ВыдачаDAO ВыдачаDAO = new ВыдачаDAO();

// GET: Выдача

public ActionResult Index()

{

return View(ВыдачаDAO.Журнал\_выдач());

}

// GET: Выдача/Details/5

public ActionResult Details(int id)

{

return View(ВыдачаDAO.Выдача(id));

}

// GET: Выдача/Create

public ActionResult Create()

{

return View();

}

// POST: Выдача/Create

[HttpPost]

public ActionResult Create(String id, Выдача выдача)

{

try

{

// TODO: Add insert logic here

if (ВыдачаDAO.Добавить\_выдачу(id,выдача))

return RedirectToAction("Index","Категория");

else

return View("Create");

}

catch

{

return View();

}

}

// GET: Выдача/Edit/5

public ActionResult Edit(int id)

{

return View(ВыдачаDAO.Выдача(id));

}

// POST: Выдача/Edit/5

[HttpPost]

public ActionResult Edit(int id, Выдача выдача)

{

try

{

if (ВыдачаDAO.Изменить\_выдачу(id, выдача))

return RedirectToAction("Index");

else

return View("Edit");

}

catch

{

return View();

}

}

// GET: Выдача/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: Выдача/Delete/5

[HttpPost]

public ActionResult Delete(int id, Выдача выдача)

{

try

{

if (ВыдачаDAO.Удалить\_выдачу(id, выдача))

return RedirectToAction("Index");

else

return View("Delete");

}

catch

{

return View();

}

}

public ActionResult Index3(String id)

{

Выдача av = new Выдача();

av.Код\_продукции = id;

av.Дата\_и\_время = DateTime.Now;

return View(av);

}

[HttpPost]

public ActionResult Index3(String id, Выдача b)

{

try

{

if (ВыдачаDAO.Добавить\_выдачу(id,b))

return RedirectToAction("Index", "Категория");

else

return View("Index3");

}

catch

{

return View();

}

}

}

}

**КатегорияController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Курсовик.Models;

using Курсовик.DAO;

namespace Курсовик.Controllers

{

[Authorize(Roles = "Администратор,Учетчик выдачи, Заведующий складом")]

public class КатегорияController : Controller

{

КатегорияDAO КатегорияDAO = new КатегорияDAO();

Кат\_продукцияDAO Кат\_ПродукцияDAO = new Кат\_продукцияDAO();

// GET: Категория

public ActionResult Index()

{

return View(КатегорияDAO.Список\_категорий());

}

public ActionResult Index2(Кат\_продукция продукция)

{

return View(Кат\_ПродукцияDAO.Список\_категорий(продукция));

}

// GET: Категория/Details/5

public ActionResult Details(int id)

{

return View(КатегорияDAO.Категория(id));

}

// GET: Категория/Create

public ActionResult Create()

{

return View();

}

// POST: Категория/Create

[HttpPost]

public ActionResult Create(Категория категория)

{

try

{

if (КатегорияDAO.Добавить\_категорию(категория))

return RedirectToAction("Index");

else

return View("Create");

}

catch

{

return View();

}

}

// GET: Категория/Edit/5

public ActionResult Edit(int id)

{

return View(КатегорияDAO.Категория(id));

}

// POST: Категория/Edit/5

[HttpPost]

public ActionResult Edit(int id, Категория категория)

{

try

{

if (КатегорияDAO.Изменить\_категорию(id,категория))

return RedirectToAction("Index");

else

return View("Edit");

}

catch

{

return View();

}

}

// GET: Категория/Delete/5

public ActionResult Delete(int id)

{

return View(КатегорияDAO.Категория(id));

}

// POST: Категория/Delete/5

[HttpPost]

public ActionResult Delete(int id, Категория категория)

{

try

{

if (КатегорияDAO.Удалить\_категорию(id,категория))

return RedirectToAction("Index");

else

return View("Delete");

}

catch

{

return View();

}

}

}

}

**Поставщик\_категорияController.cs**

using System;

using System.Web.Mvc;

using Курсовик.DAO;

using Курсовик.Models;

namespace Курсовик.Controllers

{

public class Поставщик\_категорияController : Controller

{

Поставщик\_категорияDAO Поставщик\_категорияDAO = new Поставщик\_категорияDAO();

// GET: Поставщик\_категория

public ActionResult Index1(int id)

{

String sql = "SELECT \* FROM Список\_категорий " +

"WHERE номер not in (SELECT номер\_категории from Поставщик\_категория WHERE " +

"номер\_поставщика=" + id + ")";

return View(Поставщик\_категорияDAO.Список\_категорий(sql,id));

}

public ActionResult Index2(int id)

{

String sql = "SELECT \* FROM Список\_категорий " +

"WHERE номер in (SELECT номер\_категории from Поставщик\_категория WHERE " +

"номер\_поставщика=" + id + ")";

return View(Поставщик\_категорияDAO.Список\_категорий(sql,id));

}

public ActionResult Index4(int id, int a)

{

Поставщик\_категория av = new Поставщик\_категория();

av.Код\_поставщика = id;

av.Код\_категории = a;

return View(av);

}

[HttpPost]

public ActionResult Index4(int id, int a, Поставщик\_категория b)

{

try

{

if (Поставщик\_категорияDAO.Удалить\_категорию(id, a))

{

return RedirectToAction("Index2", new { id });

}

else

return View("Index4");

}

catch

{

return View();

}

}

public ActionResult Index3(int id, int a)

{

Поставщик\_категория av = new Поставщик\_категория();

av.Код\_поставщика = id;

av.Код\_категории = a;

return View(av);

}

[HttpPost]

public ActionResult Index3(int id, int a,Поставщик\_категория b)

{

try

{

if (Поставщик\_категорияDAO.Добавить\_категорию(id,a))

return RedirectToAction("Index1",new { id });

else

return View("Index3");

}

catch

{

return View();

}

}

public ActionResult Index()

{

return View();

}

// GET: Поставщик\_категория/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: Поставщик\_категория/Create

public ActionResult Create()

{

return View();

}

// POST: Поставщик\_категория/Create

[HttpPost]

public ActionResult Create(FormCollection collection)

{

try

{

// TODO: Add insert logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

// GET: Поставщик\_категория/Edit/5

public ActionResult Edit(int id)

{

return View();

}

// POST: Поставщик\_категория/Edit/5

[HttpPost]

public ActionResult Edit(int id, FormCollection collection)

{

try

{

// TODO: Add update logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

// GET: Поставщик\_категория/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: Поставщик\_категория/Delete/5

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

try

{

// TODO: Add delete logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

}

}

**ПоставщикController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Курсовик.Models;

using Курсовик.DAO;

namespace Курсовик.Controllers

{

[Authorize(Roles = "Администратор")]

public class ПоставщикController : Controller

{

ПоставщикDAO ПоставщикDAO = new ПоставщикDAO();

public ActionResult Index()

{

return View(ПоставщикDAO.Список\_поставщиков());

}

// GET: Home1/Details/5

public ActionResult Details(int id)

{

return View(ПоставщикDAO.Поставщик(id));

}

// GET: Home1/Create

public ActionResult Create()

{

return View();

}

// POST: Home1/Create

[HttpPost]

public ActionResult Create([Bind(Exclude = "код\_поставщика")]Поставщик поставщик)

{

try

{

// TODO: Add insert logic here

if (ПоставщикDAO.Добавить\_поставщика(поставщик))

return RedirectToAction("Index");

else

return View("Create");

}

catch

{

return View();

}

}

// GET: Home1/Edit/5

public ActionResult Edit(int id)

{

return View(ПоставщикDAO.Поставщик(id));

}

// POST: Home1/Edit/5

[HttpPost]

public ActionResult Edit(int id, Поставщик поставщик)

{

try

{

// TODO: Add update logic here

if (ПоставщикDAO.Изменить\_поставщика(id,поставщик))

return RedirectToAction("Index");

else

return View("Edit");

}

catch

{

return View();

}

}

// GET: Home1/Delete/5

public ActionResult Delete(int id)

{

return View(ПоставщикDAO.Поставщик(id));

}

// POST: Home1/Delete/5

[HttpPost]

public ActionResult Delete(int id, Поставщик поставщик)

{

try

{

if (ПоставщикDAO.Удалить\_поставщика(id))

return RedirectToAction("Index");

else

return View("Delete");

}

catch

{

return View();

}

}

}

}

**ПоступлениеController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Курсовик.Models;

using Курсовик.DAO;

namespace Курсовик.Controllers

{

[Authorize(Roles = "Заведующий складом, Учетчик приема")]

public class ПоступлениеController : Controller

{

ПоступлениеDAO ПоступлениеDAO = new ПоступлениеDAO();

// GET: Поступление

public ActionResult Index()

{

return View(ПоступлениеDAO.Журнал\_поступлений());

}

// GET: Поступление/Details/5

public ActionResult Details(int id)

{

return View(ПоступлениеDAO.Поступление(id));

}

// GET: Поступление/Create

public ActionResult Create()

{

return View();

}

// POST: Поступление/Create

[HttpPost]

public ActionResult Create(Поступление поступление)

{

try

{

if (ПоступлениеDAO.Добавить\_поступление(поступление))

return RedirectToAction("Index");

else

return View("Create");

}

catch

{

return View();

}

}

// GET: Поступление/Edit/5

public ActionResult Edit(int id)

{

return View(ПоступлениеDAO.Поступление(id));

}

// POST: Поступление/Edit/5

[HttpPost]

public ActionResult Edit(int id, Поступление поступление)

{

try

{

if (ПоступлениеDAO.Изменить\_поступление(id,поступление))

return RedirectToAction("Index");

else

return View("Edit");

}

catch

{

return View();

}

}

// GET: Поступление/Delete/5

public ActionResult Delete(int id)

{

return View(ПоступлениеDAO.Поступление(id));

}

// POST: Поступление/Delete/5

[HttpPost]

public ActionResult Delete(int id, Поступление поступление)

{

try

{

if (ПоступлениеDAO.Удалить\_поступление(id,поступление))

return RedirectToAction("Index");

else

return View("Delete");

}

catch

{

return View();

}

}

}

}

ПродукцияController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Курсовик.Models;

using Курсовик.DAO;

namespace Курсовик.Controllers

{

[Authorize(Roles = "Заведующий складом,Учетчик выдачи")]

public class ПродукцияController : Controller

{

ПродукцияDAO ПродукцияDAO = new ПродукцияDAO();

Кат\_продукцияDAO Кат\_ПродукцияDAO = new Кат\_продукцияDAO();

// GET: /Home/

public ActionResult Index()

{

String sql = "SELECT \* FROM Список\_продукции";

return View(ПродукцияDAO.Список\_продукции(sql));

}

public ActionResult Index1(int id)

{

String sql = "SELECT \* FROM Список\_продукции WHERE номер\_категории="+id;

return View(ПродукцияDAO.Список\_продукции(sql));

}

public ActionResult Index3(Продукция продукция)

{

if (ПродукцияDAO.Добавить\_категорию(продукция))

{

String sql = "SELECT \* FROM Список\_продукции";

return View(ПродукцияDAO.Список\_продукции(sql));

}

else return RedirectToAction("Сreate","Продукция");

}

// GET: /Home/Details/5

public ActionResult Details(String id)

{

return View(ПродукцияDAO.Продукция(id));

}

//

// GET: /Home/Create

[Authorize(Roles = "Заведующий складом")]

public ActionResult Create()

{

return View();

}

//

// POST: /Home/Create

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult Create(Кат\_продукция продукция)

{

try

{

return RedirectToAction("Index2", "Категория", продукция);

}

catch

{

return View("Create");

}

}

// GET: Home/Create

[Authorize(Roles = "Заведующий складом")]

// GET: Home/Edit/5

public ActionResult Edit(String id)

{

return View(ПродукцияDAO.Продукция(id));

}

// POST: Home/Edit/5

[HttpPost]

public ActionResult Edit(String id, Продукция продукция)

{

try

{

if (ПродукцияDAO.Изменить\_продукцию(id,продукция))

return RedirectToAction("Index");

else

return View("Edit");

}

catch

{

return View();

}

}

[Authorize(Roles = "Заведующий складом")]

// GET: Home/Delete/5

public ActionResult Delete(String id)

{

return View(ПродукцияDAO.Продукция(id));

}

// POST: Home/Delete/5

[HttpPost]

public ActionResult Delete(String id, Продукция продукция)

{

try

{

if (ПродукцияDAO.Удалить\_продукцию(id,продукция))

return RedirectToAction("Index");

else

return View("Delete");

}

catch

{

return View();

}

}

// POST: Категория/Delete/5

public ActionResult Index4(Кат\_продукция id)

{

try

{

Продукция продукция = new Продукция();

продукция.Категория = id.Код\_категории;

продукция.Код\_продукции = id.Код\_продукции;

продукция.Количество = id.Количество;

продукция.Масса = id.Масса;

продукция.Материал = id.Материал;

продукция.Название = id.Название\_продукции;

продукция.Цена\_за\_единицу = id.Цена\_за\_единицу;

if (ПродукцияDAO.Добавить\_продукт(продукция))

return RedirectToAction("Index");

else

return View("Index3");

}

catch(Exception ex)

{

String a = ex.Message;

return View();

}

}

}

}

РасходController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Курсовик.DAO;

using Курсовик.Models;

namespace Курсовик.Controllers

{

[Authorize(Roles = "Заведующий складом, Администратор")]

public class РасходController : Controller

{

РасходDAO РасходDAO = new РасходDAO();

ПродукцияDAO ПродукцияDAO = new ПродукцияDAO();

// GET: Расход

public ActionResult Index()

{

String sql = "SELECT \* FROM Список\_продукции WHERE статус is NUll";

return View(ПродукцияDAO.Список\_продукции(sql));

}

public ActionResult Index1()

{

String sql = "SELECT \* FROM Список\_продукции WHERE статус is not NUll";

return View(ПродукцияDAO.Список\_продукции(sql));

}

public ActionResult Index2()

{

if (РасходDAO.Обновить\_статус1())

return RedirectToAction("Index","Поставщик");

else

return View("Index1");

}

public ActionResult Index3()

{

if (РасходDAO.Обновить\_статус2())

return RedirectToAction("Index", "Поставщик");

else

return View("Index1");

}

public ActionResult Index4()

{

String sql = "SELECT "+

"m.код\_продукции as a, "+

"m.название as b, "+

"m.расход\_по\_норме as c, "+

"m.статус, "+

"sum(u.количество) as d, "+

"u.код\_продукции, "+

"sum(n.количество) as e, "+

"n.код\_продукции "+

"FROM Список\_продукции m "+

"INNER JOIN Журнал\_выдач u "+

"ON m.код\_продукции = u.код\_продукции "+

"INNER JOIN Журнал\_поступлений n "+

"ON m.код\_продукции = n.код\_продукции "+

"WHERE m.статус = 'создан' "+

"Group By u.код\_продукции, m.код\_продукции, "+

"n.код\_продукции, m.название, m.статус, " +

"m.расход\_по\_норме";

return View(РасходDAO.Табель\_расхода(sql));

}

// GET: Расход/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: Расход/Create

public ActionResult Create()

{

return View();

}

// POST: Расход/Create

[HttpPost]

public ActionResult Create(Расход расход)

{

try

{

// TODO: Add insert logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

// GET: Расход/Edit/5

public ActionResult Edit(String id)

{

return View();

}

// POST: Расход/Edit/5

[HttpPost]

public ActionResult Edit(String id, Расход расход)

{

try

{

// TODO: Add update logic here

if (РасходDAO.Обновить\_статус(id, расход))

return RedirectToAction("Index");

else

return View("Create");

}

catch

{

return View();

}

}

// GET: Расход/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: Расход/Delete/5

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

try

{

// TODO: Add delete logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

}

}

**DAO-классы**

**DAO.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using System.Configuration;

namespace Курсовик.DAO

{

public class DAO

{

private string ConnectionString =

ConfigurationManager.ConnectionStrings["YourDB"].ConnectionString;

protected SqlConnection Сonnection { get; set; }

public void Connect()

{

Сonnection = new SqlConnection(ConnectionString);

Сonnection.Open();

}

public void Disconnect()

{

Сonnection.Close();

}

}

}

Выдача.DAO

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class ВыдачаDAO : DAO

{

public List<Выдача> Журнал\_выдач()

{

Connect();

List<Выдача> Журнал\_выдач = new List<Выдача>();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Журнал\_выдач", Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Выдача выдача = new Выдача

{

Код\_выдачи = Convert.ToInt32(reader["номер"]),

Код\_продукции = Convert.ToString(reader["код\_продукции"]),

Дата\_и\_время = Convert.ToDateTime(reader["дата\_и\_время"]),

Количество = Convert.ToInt32(reader["количество"])

};

Журнал\_выдач.Add(выдача);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Журнал\_выдач;

}

public bool Добавить\_выдачу(String id, Выдача выдача)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"INSERT INTO Журнал\_выдач (код\_продукции, дата\_и\_время, " +

"количество) " +

"VALUES (@код\_продукции, @дата\_и\_время, @количество)", Сonnection);

cmd.Parameters.Add(new SqlParameter("@код\_продукции", id));

cmd.Parameters.Add(new SqlParameter("@дата\_и\_время", DateTime.Now));

cmd.Parameters.Add(new SqlParameter("@количество", выдача.Количество));

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

String a = ex.Message;

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Удалить\_выдачу(int id,Выдача выдача)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"DELETE FROM Журнал\_выдач WHERE номер=" + id, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public Выдача Выдача(int id)

{

Connect();

Выдача Поступление = new Выдача();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Журнал\_выдач WHERE номер =" + id, Сonnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

Поступление.Код\_выдачи = Convert.ToInt32(reader["номер"]);

Поступление.Код\_продукции = Convert.ToString(reader["код продукции"]);

Поступление.Дата\_и\_время = Convert.ToDateTime(reader["дата\_и\_время"]);

Поступление.Количество = Convert.ToInt32(reader["количество"]);

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Поступление;

}

public bool Изменить\_выдачу(int id, Выдача выдача)

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Журнал\_выдач SET код\_продукции='{0}', дата\_и\_время='{1}', количество='{2}' " +

"WHERE номер={3}", выдача.Код\_продукции, выдача.Дата\_и\_время, выдача.Количество, id);

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

Кат\_продукцияDAO.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class Кат\_продукцияDAO:DAO

{

public List<Кат\_продукция> Список\_категорий(Кат\_продукция категория1)

{

Connect();

List<Кат\_продукция> Список\_категорий = new List<Кат\_продукция>();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Список\_категорий", Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Кат\_продукция категория = new Кат\_продукция();

категория.Код\_категории = Convert.ToInt32(reader["номер"]);

категория.Название = Convert.ToString(reader["название"]);

категория.Количество\_товара = Convert.ToInt32(reader["количество"]);

категория.Дата\_создания = Convert.ToDateTime(reader["дата"]);

категория.Код\_продукции = категория1.Код\_продукции;

категория.Количество = категория1.Количество;

категория.Масса = категория1.Масса;

категория.Категория = Convert.ToInt32(reader["номер"]);

категория.Материал = категория1.Материал;

категория.Название\_продукции = категория1.Название\_продукции;

категория.Цена\_за\_единицу = категория1.Цена\_за\_единицу;

Список\_категорий.Add(категория);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Список\_категорий;

}

}

}

КатегорияDAO.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class КатегорияDAO:DAO

{

public List<Категория> Список\_категорий()

{

Connect();

List<Категория> Список\_категорий = new List<Категория>();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Список\_категорий", Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Категория категория = new Категория();

категория.Код\_категории = Convert.ToInt32(reader["номер"]);

категория.Название = Convert.ToString(reader["название"]);

категория.Количество\_товара = Convert.ToInt32(reader["количество"]);

категория.Дата\_создания = Convert.ToDateTime(reader["дата"]);

Список\_категорий.Add(категория);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Список\_категорий;

}

public bool Добавить\_категорию(Категория категория)

{

bool result = true;

Connect();

try

{

SqlCommand cmd = new SqlCommand(

"INSERT INTO Список\_категорий(название, количество, дата) VALUES (@название, @количество, @дата)", Сonnection);

cmd.Parameters.Add(new SqlParameter("@название", категория.Название));

cmd.Parameters.Add(new SqlParameter("@количество", '0'));

cmd.Parameters.Add(new SqlParameter("@дата", DateTime.Now));

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

String a=ex.ToString();

result = false;

}

finally

{

Disconnect();

}

return result;

}

public Категория Категория(int id)

{

Connect();

Категория Категория = new Категория();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Список\_категорий WHERE номер =" + id, Сonnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

Категория.Код\_категории = Convert.ToInt32(reader["номер"]);

Категория.Название = Convert.ToString(reader["название"]);

Категория.Количество\_товара = Convert.ToInt32(reader["количество"]);

Категория.Дата\_создания = Convert.ToDateTime(reader["дата"]);

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Категория;

}

public bool Изменить\_категорию(int id, Категория категория)

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Список\_категорий SET название='{0}', дата='{1}' " +

"WHERE номер='{2}'", категория.Название, DateTime.Now, id);

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Удалить\_категорию(int id,Категория категория)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"DELETE FROM Список\_категорий WHERE номер =" + id, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

Поставщик\_категорияDAO.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class Поставщик\_категорияDAO:DAO

{

public List<Поставщик\_категория> Список\_категорий(String sql,int id)

{

Connect();

List<Поставщик\_категория> Список\_категорий = new List<Поставщик\_категория>();

try

{

SqlCommand command = new SqlCommand(sql, Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Поставщик\_категория категория = new Поставщик\_категория();

категория.Код\_поставщика = id;

категория.Код\_категории = Convert.ToInt32(reader["номер"]);

категория.Название = Convert.ToString(reader["название"]);

категория.Количество\_товара = Convert.ToInt32(reader["количество"]);

категория.Дата\_создания = Convert.ToDateTime(reader["дата"]);

Список\_категорий.Add(категория);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Список\_категорий;

}

public bool Удалить\_категорию(int id, int a)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"DELETE FROM Поставщик\_категория WHERE (номер\_поставщика="+id+") and (номер\_категории="+a+")" , Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Добавить\_категорию(int id, int a)

{

bool result = true;

Connect();

try

{

SqlCommand cmd = new SqlCommand(

"INSERT INTO Поставщик\_категория(номер\_категории, номер\_поставщика) VALUES (@номер\_категории, @номер\_поставщика)", Сonnection);

cmd.Parameters.Add(new SqlParameter("@номер\_категории", a));

cmd.Parameters.Add(new SqlParameter("@номер\_поставщика", id));

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

String ac = ex.Message;

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

**ПоставщикDAO.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class ПоставщикDAO : DAO

{

public List<Поставщик> Список\_поставщиков()

{

Connect();

List<Поставщик> Список\_поставщиков = new List<Поставщик>();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Список\_поставщиков", Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Поставщик поставщик = new Поставщик();

поставщик.Код\_поставщика = Convert.ToInt32(reader["код\_поставщика"]);

поставщик.Наименование\_организации = Convert.ToString(reader["наименование\_организации"]);

поставщик.Адрес = Convert.ToString(reader["адрес"]);

поставщик.Почта = Convert.ToString(reader["почта"]);

поставщик.Телефон = Convert.ToString(reader["телефон"]);

Список\_поставщиков.Add(поставщик);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Список\_поставщиков;

}

public bool Добавить\_поставщика(Поставщик поставщик)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"INSERT INTO Список\_поставщиков (наименование\_организации, адрес, " +

"почта, телефон) " +

"VALUES (@наименование\_организации, @адрес, @почта,@телефон)", Сonnection);

cmd.Parameters.Add(new SqlParameter("@наименование\_организации", поставщик.Наименование\_организации));

cmd.Parameters.Add(new SqlParameter("@адрес", поставщик.Адрес));

cmd.Parameters.Add(new SqlParameter("@почта", поставщик.Почта));

cmd.Parameters.Add(new SqlParameter("@телефон", поставщик.Телефон));

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Удалить\_поставщика(int id)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"DELETE FROM Список\_поставщиков WHERE код\_поставщика=" + id, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

String a = ex.Message;

result = false;

}

finally

{

Disconnect();

}

return result;

}

public Поставщик Поставщик(int id)

{

Connect();

Поставщик Поставщик = new Поставщик();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Список\_поставщиков WHERE код\_поставщика ="+id, Сonnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

Поставщик.Код\_поставщика = Convert.ToInt32(reader["код\_поставщика"]);

Поставщик.Наименование\_организации = Convert.ToString(reader["наименование\_организации"]);

Поставщик.Адрес = Convert.ToString(reader["адрес"]);

Поставщик.Почта = Convert.ToString(reader["почта"]);

Поставщик.Телефон = Convert.ToString(reader["телефон"]);

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Поставщик;

}

public bool Изменить\_поставщика(int id, Поставщик поставщик)

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Список\_поставщиков SET наименование\_организации='{0}', адрес='{1}', почта='{2}', телефон='{3}' " +

"WHERE код\_поставщика='{4}'", поставщик.Наименование\_организации, поставщик.Адрес, поставщик.Почта, поставщик.Телефон, id);

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

ПоступлениеDAO.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class ПоступлениеDAO:DAO

{

public List<Поступление> Журнал\_поступлений()

{

Connect();

List<Поступление> Журнал\_поступлений = new List<Поступление>();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Журнал\_поступлений", Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Поступление поступление = new Поступление();

поступление.Код\_поступления = Convert.ToInt32(reader["номер"]);

поступление.Код\_продукции = Convert.ToString(reader["код\_продукции"]);

поступление.Дата\_и\_время = Convert.ToDateTime(reader["дата\_и\_время"]);

поступление.Количество = Convert.ToInt32(reader["количество"]);

Журнал\_поступлений.Add(поступление);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Журнал\_поступлений;

}

public bool Добавить\_поступление(Поступление поступление)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"INSERT INTO Журнал\_поступлений (код\_продукции, дата\_и\_время, количество) " +

"VALUES (@код\_продукции, @дата\_и\_время, @количество)", Сonnection);

cmd.Parameters.Add(new SqlParameter("@код\_продукции", поступление.Код\_продукции));

cmd.Parameters.Add(new SqlParameter("@дата\_и\_время", DateTime.Now));

cmd.Parameters.Add(new SqlParameter("@количество", поступление.Количество));

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

String a = ex.Message;

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Удалить\_поступление(int id, Поступление поступление)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand("DELETE FROM Журнал\_поступлений WHERE номер=" + id, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public Поступление Поступление(int id)

{

Connect();

Поступление Поступление = new Поступление();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Журнал\_поступлений WHERE номер =" + id, Сonnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

Поступление.Код\_поступления = Convert.ToInt32(reader["номер"]);

Поступление.Код\_продукции = Convert.ToString(reader["код\_продукции"]);

Поступление.Дата\_и\_время = Convert.ToDateTime(reader["дата\_и\_время"]);

Поступление.Количество = Convert.ToInt32(reader["количество"]);

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Поступление;

}

public bool Изменить\_поступление(int id,Поступление поступление)

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Журнал\_поступлений SET код\_продукции='{0}', дата\_и\_время='{1}', количество='{2}' "+

"WHERE номер={3}", поступление.Код\_продукции, DateTime.Now, поступление.Количество, id);

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

**ПродукцияDAO.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Data.SqlClient;

using Курсовик.Models;

namespace Курсовик.DAO

{

public class ПоступлениеDAO:DAO

{

public List<Поступление> Журнал\_поступлений()

{

Connect();

List<Поступление> Журнал\_поступлений = new List<Поступление>();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Журнал\_поступлений", Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Поступление поступление = new Поступление();

поступление.Код\_поступления = Convert.ToInt32(reader["номер"]);

поступление.Код\_продукции = Convert.ToString(reader["код\_продукции"]);

поступление.Дата\_и\_время = Convert.ToDateTime(reader["дата\_и\_время"]);

поступление.Количество = Convert.ToInt32(reader["количество"]);

Журнал\_поступлений.Add(поступление);

}

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Журнал\_поступлений;

}

public bool Добавить\_поступление(Поступление поступление)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand(

"INSERT INTO Журнал\_поступлений (код\_продукции, дата\_и\_время, количество) " +

"VALUES (@код\_продукции, @дата\_и\_время, @количество)", Сonnection);

cmd.Parameters.Add(new SqlParameter("@код\_продукции", поступление.Код\_продукции));

cmd.Parameters.Add(new SqlParameter("@дата\_и\_время", DateTime.Now));

cmd.Parameters.Add(new SqlParameter("@количество", поступление.Количество));

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

String a = ex.Message;

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Удалить\_поступление(int id, Поступление поступление)

{

Connect();

bool result = true;

try

{

SqlCommand cmd = new SqlCommand("DELETE FROM Журнал\_поступлений WHERE номер=" + id, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public Поступление Поступление(int id)

{

Connect();

Поступление Поступление = new Поступление();

try

{

SqlCommand command = new SqlCommand("SELECT \* FROM Журнал\_поступлений WHERE номер =" + id, Сonnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

Поступление.Код\_поступления = Convert.ToInt32(reader["номер"]);

Поступление.Код\_продукции = Convert.ToString(reader["код\_продукции"]);

Поступление.Дата\_и\_время = Convert.ToDateTime(reader["дата\_и\_время"]);

Поступление.Количество = Convert.ToInt32(reader["количество"]);

reader.Close();

}

catch (Exception)

{

}

finally

{

Disconnect();

}

return Поступление;

}

public bool Изменить\_поступление(int id,Поступление поступление)

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Журнал\_поступлений SET код\_продукции='{0}', дата\_и\_время='{1}', количество='{2}' "+

"WHERE номер={3}", поступление.Код\_продукции, DateTime.Now, поступление.Количество, id);

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

**РасходDAO.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using Курсовик.Models;

using System.Data.SqlClient;

namespace Курсовик.DAO

{

public class РасходDAO:DAO

{

public List<Расход> Табель\_расхода(String sql)

{

Connect();

List<Расход> Табель\_расхода = new List<Расход>();

try

{

SqlCommand command = new SqlCommand(sql, Сonnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Расход товар = new Расход();

товар.Код\_продукции = Convert.ToString(reader["a"]);

товар.Название = Convert.ToString(reader["b"]);

товар.Приход\_за\_сутки = Convert.ToInt32(reader["c"]);

товар.Расход\_за\_сутки = Convert.ToInt32(reader["d"]);

товар.Расход\_по\_норме = Convert.ToUInt32(reader["e"]);

//товар.Код\_продукции = reader.GetString(0);

//товар.Название = reader.GetString(1);

//товар.Приход\_за\_сутки = reader.GetInt32(6);

//товар.Расход\_за\_сутки = reader.GetInt32(4);

//товар.Расход\_по\_норме= reader.GetFloat(2);

товар.Коэффициент\_расхода = (float )товар.Расход\_за\_сутки/(float)товар.Приход\_за\_сутки;

Табель\_расхода.Add(товар);

}

reader.Close();

}

catch (Exception ex)

{

String a = ex.Message;

a="";

}

finally

{

Disconnect();

}

return Табель\_расхода;

}

public bool Обновить\_статус(String id, Расход расход)

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Список\_продукции SET расход\_по\_норме='{0}', статус= 'черновик'"+

"WHERE код\_продукции='{1}'", расход.Расход\_по\_норме, id);

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Обновить\_статус1()

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Список\_продукции SET статус= 'создан'" +

"WHERE расход\_по\_норме is not NULL");

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

public bool Обновить\_статус2()

{

Connect();

bool result = true;

try

{

String sql = string.Format("UPDATE Список\_продукции SET статус=null, расход\_по\_норме=null");

SqlCommand cmd = new SqlCommand(sql, Сonnection);

cmd.ExecuteNonQuery();

}

catch (Exception)

{

result = false;

}

finally

{

Disconnect();

}

return result;

}

}

}

**Модели данных**

**Выдача.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Выдача

{

public int Код\_выдачи { get; set; }

public String Код\_продукции { get; set; }

public DateTime Дата\_и\_время { get; set; }

public int Количество { get; set; }

public List<Выдача> Список\_выдач { get; set; }

}

}

**Кат\_продукция.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Кат\_продукция

{

public String Код\_продукции { get; set; }

public String Название\_продукции { get; set; }

public int Категория { get; set; }

public String Материал { get; set; }

public float Масса { get; set; }

public float Цена\_за\_единицу { get; set; }

public int Количество { get; set; }

public int Код\_категории { get; set; }

public String Название { get; set; }

public DateTime Дата\_создания { get; set; }

public int Количество\_товара { get; set; }

}

}

Категория.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Категория

{

public int Код\_категории { get; set; }

public String Название { get; set; }

public DateTime Дата\_создания { get; set; }

public int Количество\_товара { get; set; }

public List<Категория> Список\_категорий { get; set; }

}

}

**Поставщик.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Поставщик

{

public int Код\_поставщика { get; set; }

public String Наименование\_организации { get; set; }

public String Адрес { get; set; }

public String Почта { get; set; }

public String Телефон { get; set; }

public List<Поставщик> Список\_поставщиков { get; set; }

}

}

**Поставщик\_категория.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Поставщик\_категория

{

public int Код\_поставщика { get; set; }

public int Код\_категории { get ; set ; }

public String Название { get; set; }

public DateTime Дата\_создания { get; set; }

public int Количество\_товара { get; set; }

}

}

Поступление.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Поступление

{

public int Код\_поступления { get; set; }

public String Код\_продукции { get; set; }

public DateTime Дата\_и\_время { get; set; }

public int Количество { get; set; }

public List<Поступление> Список\_поступлений { get; set; }

}

}

Продукция.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Продукция

{

public String Код\_продукции { get; set; }

public String Название { get; set; }

public int Категория { get; set; }

public String Материал { get; set; }

public float Масса { get; set; }

public float Цена\_за\_единицу { get; set; }

public int Количество { get; set; }

public List<Продукция> Список\_продукции { get; set; }

}

}

**Расход.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Курсовик.Models

{

public class Расход

{

public String Код\_продукции { get; set; }

public String Название { get; set; }

public int Приход\_за\_сутки { get; set; }

public int Расход\_за\_сутки { get; set; }

public float Расход\_по\_норме { get; set; }

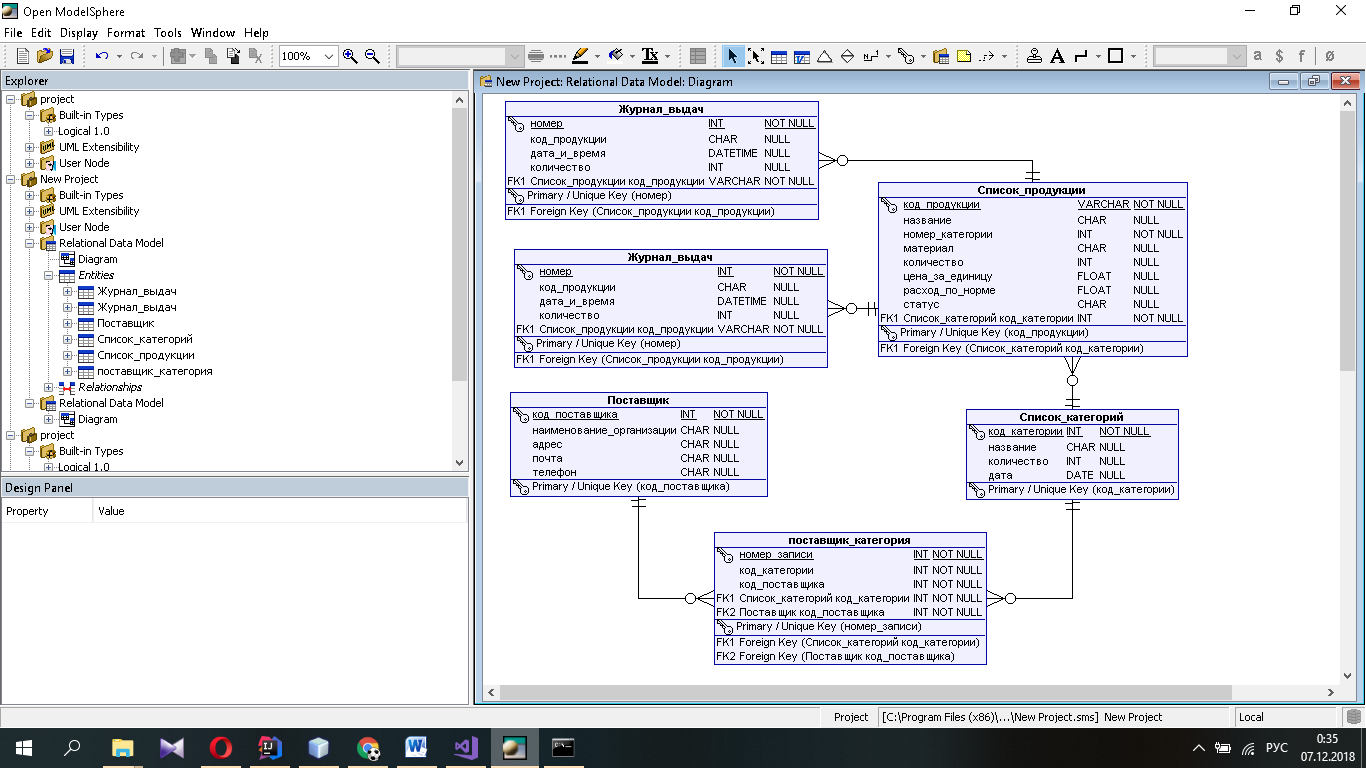
public float Коэффициент\_расхода { get; set; }

}

}

**База данных**

**Физическая схема**



**Скрипты создания таблиц**

**Журнал\_выдач**

CREATE TABLE [dbo].[Журнал\_выдач] (

[номер] INT IDENTITY (1, 1) NOT NULL,

[код\_продукции] NVARCHAR (15) NULL,

[дата\_и\_время] DATETIME NULL,

[количество] INT NULL,

PRIMARY KEY CLUSTERED ([номер] ASC),

CONSTRAINT [FK\_Журнал\_выдач\_ToTable] FOREIGN KEY ([код\_продукции]) REFERENCES [dbo].[Список\_продукции] ([код\_продукции])

);

**Триггер к таблице**

GO

CREATE TRIGGER [Выдать продукцию]

ON [dbo].[Журнал\_выдач]

FOR DELETE, INSERT, UPDATE

AS

BEGIN

DECLARE @s int;

DECLARE @b int;

DECLARE @a nvarchar(15);

SELECT @S =(select количество from inserted);

Update Список\_продукции SET количество=количество-@S WHERE код\_продукции in (select код\_продукции from inserted);

SELECT @b=(select количество from Список\_продукции WHERE код\_продукции in (select код\_продукции from inserted));

IF @b<0

BEGIN

PRINT 'Недостаточно продукции'

ROLLBACK TRANSACTION

END

END

**Журнал\_поступлений**

CREATE TABLE [dbo].[Журнал\_поступлений] (

[номер] INT IDENTITY (1, 1) NOT NULL,

[код\_продукции] NVARCHAR (15) NULL,

[дата\_и\_время] DATETIME NULL,

[количество] INT NULL,

PRIMARY KEY CLUSTERED ([номер] ASC),

CONSTRAINT [FK\_Журнал\_поступлений\_ToTable] FOREIGN KEY ([код\_продукции]) REFERENCES [dbo].[Список\_продукции] ([код\_продукции])

);

**Триггер к таблице**

GO

CREATE TRIGGER [Добавление продукции]

ON [dbo].[Журнал\_поступлений]

AFTER DELETE, INSERT, UPDATE

AS

BEGIN

DECLARE @s int;

DECLARE @a nvarchar(15);

SELECT @S =(select количество from inserted);

Update Список\_продукции SET количество=количество+@S WHERE код\_продукции in (select код\_продукции from inserted);

SELECT @a=(select код\_продукции from inserted);

IF @a not in (SELECT (код\_продукции) from Список\_продукции)

BEGIN

PRINT 'Продукция не найдена'

ROLLBACK TRANSACTION

END

END

**Таблица Поставщик\_категория**

CREATE TABLE [dbo].[Поставщик\_категория] (

[номер\_записи] INT IDENTITY (1, 1) NOT NULL,

[номер\_категории] INT NULL,

[номер\_поставщика] INT NULL,

PRIMARY KEY CLUSTERED ([номер\_записи] ASC),

CONSTRAINT [FK\_Поставщик\_категория\_ToTable] FOREIGN KEY ([номер\_категории]) REFERENCES [dbo].[Список\_категорий] ([номер]),

CONSTRAINT [FK\_Поставщик\_категория\_ToTable\_1] FOREIGN KEY ([номер\_поставщика]) REFERENCES [dbo].[Список\_поставщиков] ([код\_поставщика]) ON DELETE CASCADE

);

**Таблица Список\_категорий**

CREATE TABLE [dbo].[Список\_категорий] (

[номер] INT IDENTITY (1, 1) NOT NULL,

[название] CHAR (50) NULL,

[количество] INT NULL,

[дата] DATETIME NULL,

PRIMARY KEY CLUSTERED ([номер] ASC)

);

**Таблица Список\_поставщиков**

CREATE TABLE [dbo].[Список\_поставщиков] (

[код\_поставщика] INT IDENTITY (1, 1) NOT NULL,

[наименование\_организации] CHAR (50) NULL,

[адрес] CHAR (50) NULL,

[почта] CHAR (50) NULL,

[телефон] CHAR (50) NULL,

PRIMARY KEY CLUSTERED ([код\_поставщика] ASC)

);

**Триггер к таблице**

GO

CREATE TRIGGER [Удаление категорий]

ON [dbo].[Список\_поставщиков]

FOR DELETE

AS

BEGIN

DELETE FROM Поставщик\_категория WHERE номер\_поставщика in (SELECT код\_поставщика from deleted);

END

**Таблица Список\_продукции**

CREATE TABLE [dbo].[Список\_продукции] (

[код\_продукции] NVARCHAR (15) NOT NULL,

[название] CHAR (30) NULL,

[номер\_категории] INT NULL,

[материал] CHAR (20) NULL,

[масса] FLOAT (53) NULL,

[количество] INT NULL,

[цена\_за\_единицу] INT NULL,

[расход\_по\_норме] INT NULL,

[статус] CHAR (20) NULL,

PRIMARY KEY CLUSTERED ([код\_продукции] ASC),

CONSTRAINT [FK\_Список\_продукции\_ToTable] FOREIGN KEY ([номер\_категории]) REFERENCES [dbo].[Список\_категорий] ([номер])

);

**Триггер к таблице**

GO

CREATE TRIGGER [Категории]

ON [dbo].[Список\_продукции]

for DELETE, INSERT, UPDATE

AS

BEGIN

DECLARE @b INt;

declare @a int;

Declare BankCursor Cursor LOCAL for Select sum(количество), номер\_категории from Список\_продукции Group by номер\_категории;

Open BankCursor;

WHILE @@FETCH\_STATUS = 0 BEGIN

FETCH BankCursor INTO @a,@b;

Update Список\_категорий SET количество=@a WHERE номер=@b;

END;

Close BankCursor;

END