**Содержание**

Введение…………………………………………………………………….............

Анализ проблемы…………………………………………………………………..

Анализ функциональных требований……………………...……………………..

Предсказание на основе экстраполяции значений продаж за предыдущие месяцы……………………………….………………………………………….......

Используемые технологии…..………………………………………………..........

Реализация……………………………………………………………………..........

Заключение…………………………………………………………………….........

Список использованных источников……………………………………………...

**Введение**

На сегодняшний день применение баз данных и бизнес-приложений приобрело весьма важное значение для многих организаций, которые для упрощения своей работы применяют компьютерные технологии.

Большинство, если не все бизнес-приложения предназначены для обработки бизнес-данных. Самые первые бизнес-решения так и назывались — программы обработки данных*.* Эффективное хранение, обработка и взаимодействие с данными — только важная составляющая управления предприятием, в тоже время компании инвестируют значительные средства в разработку компьютеризированных системы для эффективного решения этих задач.

В настоящее время, несмотря на повышение компьютеризации общества, в сфере бизнеса и торговли до сих пор нет средств, позволяющих в достаточной мере автоматизировать процесс пополнения склада товарами. Одной из основных задач можно рассматривать проблему предсказания предложения, а также оперативную корректировку данных при возникновении необходимости в этом.

О своевременности и **актуальности** рассматриваемой проблемы говорит тот факт, что большую часть своего времени администрация магазина или завода тратит на анализ спроса предыдущих месяцев. Огромное количество магазинов и отсутствие предложений в данной сфере гарантирует высокую потребность в данном продукте.

**Объектом** для создания бизнес-приложения я решил взять бетонный завод. Данная программа предлагает хранение данных о покупателях и их покупках.

Базу данных будут использовать не только администрация бетонного завода, а также клиенты, совершая свои покупки используя отдельное веб-приложение.

**Целью** данного курсового проекта является создание приложения, которое может хранить данные о покупках и анализировать их, помогая администрации предсказывать какое количество товаров необходимо завезти на склад.

**Основные** **задачи**:

1. Прогнозирование спроса на бетон с целью повышения эффективности деятельности бетонного завода.
2. Упрощение и повышение эффективности работы администрации завода.

В данной курсовой работе будет использована специальная объектно-ориентированная технология на базе фреймворка .NET для работы с данными — Entity Framework, которая будет взаимодействовать с SQL Server 2014. Приложение для администрирования написано на Windows Application Forms C#, а веб-приложение для клиентов предприятия реализовано на технологии ASP .NET MVC 5.

**1 Анализ проблемы**

Моделирование деятельности бетонного предприятия нужно начать с клиентского приложения. Для заказа пользователь будет вводить свои данные для своей идентификации и выбирать понравившийся товар, а также его количество. В соответствие с количеством и ценой товара за штуку будет формироваться сумма заказа. Это приложение необходимо будет распространять, так что веб-приложение здесь подходит больше всего. Все покупки будут хранятся в базе данных, для этого я завёл отдельную таблицу Purchases в SQL Server 2014.

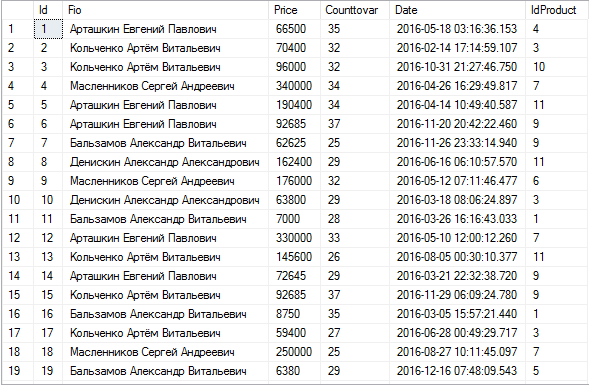


Рисунок 1 — Таблица в SQL Server Purchases

Id — идентификатор покупки. Fio — фамилия, имя, отчество покупателя. Counttovar — количество купленного товара. Date — дата покупки. IdProduct — идентификатор купленного товара.

Руководству бетонного завода нужны функции просмотра и редактирования товаров, которые есть на складе. Все товары будут хранятся в базе данных, для этого я завёл отдельную таблицу Products в SQL Server 2014.

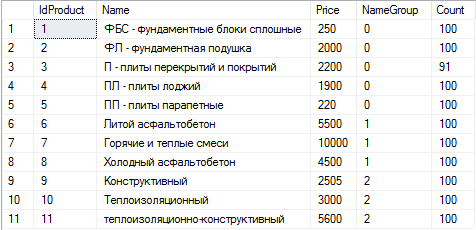


Рисунок 2 — Таблица в SQL Server Products

IdProduct — идентификатор товара. Name — наименование товара. Price — цена за штуку товара. NameGroup — группа товара (железобетон, асфальтобетон или керамзитобетон). Count — количество товара на складе.

Таблицы dbo.Products и dbo.Purchases связаны по ключевому полю IdProduct связью один ко многим.

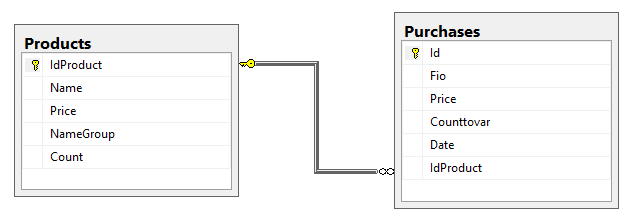


Рисунок 3 — Связь таблицы Purchases и Products

Анализ спроса и его последующее предсказание будет производится с помощью таблицы Purchases.

Для предсказаний с использованием экстраполяции нужны функции, встроенные в MATLAB.

**2 Анализ функциональных требований**

Для реализации поставленной задачи мне необходимы два приложения. Первое это приложение для клиентов бетонного завода, а второе для администрирования.

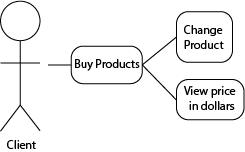


Рисунок 4 — Действия клиента

Для клиента доступны функции:

1. Покупка товаров. Для покупки пользователь будет вводить свои данные и количество товара.
   1. Выбор из ассортимента товаров, так как обычно бетонные заводы производят сразу несколько товаров.
   2. Возможность просмотра цены в Долларах.

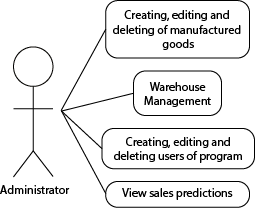


Рисунок 5 — Действия администратора

Для администратора доступны функции:

1. Создание, редактирование и удаление производимых товаров. В случае, когда завод начнёт производить новые товары не нужно будет вносить из-под SQL Server, это можно будет сделать прямо из приложения.
2. Пополнение склада товаров, а также просмотр статистики и истории продаж для отдельно взятого товара.
3. Создание, редактирование и удаление пользователей программы. Это дополнительная возможность завести несколько учётных записей на нескольких сотрудников.
4. Просмотр предсказаний продаж на следующий месяц.

**Предсказание на основе экстраполяции значений продаж за предыдущие месяцы**

Экстраполяция — это особый тип аппроксимации, при котором функция аппроксимируется вне заданного интервала, а не между заданными значениями.

Иными словами, экстраполяция — приближённое определение значений функции в точках , лежащих вне отрезка , по её значениям в точках .

Для экстраполирования значений я использовал MATLAB R2016a, а если конкретнее использовал функцию interp1 со значением extrap. Для использования функций MATLAB в приложение на C# нам потребуется интегрировать написанную функцию в C#.

Чтобы собрать dll-библиотеку MATLAB'а для интеграции в C#.NET, нужно настроить линкер, т.е. какой средой мы будем собирать проект. Для начала нужно установить среду выполнения MCR (MATLAB Component Runtime). Это набор dll-библиотек для полной поддержки языка MATLAB. Установочный файл можно найти:

...\MATLAB\R2016a\toolbox\compiler\deploy\win32\MCRInstaller

Для настройки линкера в командном окне MATLAB'а набираем mbuild -setup. Со всем соглашаемся и выбираем нужную нам среду, в нашем случае это Microsoft Visual Studio 2015. Получаем что-то похожее:

Please choose your compiler for building standalone MATLAB applications:

Would you like mbuild to locate installed compilers [y]/n? y

Select a compiler:

[1] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2016a\sys\lcc

[2] Microsoft Visual C++ 2008 SP1 in C:\Program Files\Microsoft Visual Studio 15

[0] None

Compiler: 2

Please verify your choices:

Compiler: Microsoft Visual C++ 2008 SP1

Location: C:\Program Files\Microsoft Visual Studio 15

Are these correct [y]/n? y

Следующий шаг — это написание самой функции, компилятор MATLAB'а понимает только функции т.е., каждый сценарий должен начинаться с function (желательно заканчиваться end) и быть отдельным m-файлом.

function vq3 = hint(x,v)

xq = length(x)+1;

vq3 = interp1(x,v,xq,'pchip','extrap');

end

После этого мы уже можем получить динамическую библиотеку. Набираем в командном окне MATLAB'a deploytool. Создаем новый .NET Assembly проект, указываем размещение. Далее создаем класс, в моём случае он называется MLHint, добавляем в него m-файл и нажимаем кнопку build.

После компиляции создается интересующая нас папка с установочным файлом для нашей библиотеки. На эту библиотеку в дальнейшем и будет ссылаться наша программа.

**Используемые технологии**

**ASP .NET MVC 5**

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

**Entity Framework**

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

**Windows Forms**

Windows Forms представляет простейшую технологию для создания графических настольных программ на платформе .NET. И в этом разделе выкладываются все материалы, посвященные созданию оконных приложений с использованием технологии Windows Forms, в том числе по работе различными элементами управления, двухмерной и трехмерной графикой.

**SQL Server 2014**

SQL Server 2014 представляет собой мощный полнофункциональный сервер баз данных, отличающийся высокой производительностью быстротой освоения и удобным интерфейсом администрирования. Под его управлением могут работать базы данных в широком диапазоне от уровня среднего звена предприятия до распределённых баз масштаба корпорации.

**Реализация**

Данный курсовой проект состоит из 3 приложений — Data, ClientWebApi и Admin.

Data — это приложение, которое содержит в себе описание моделей и контекста данных, для работы с базами данных через Entity Framework. А также это приложение работает как инициализатор и, если база данных не существует на компьютере, Data автоматически создаст ее. Основной код описан в классе Program.

using System;

using System.Collections.Generic;

using System.Data.Entity;

namespace Data

{

class Program

{

static void Main(string[] args)

{

try

{

using (BaseContext db = new BaseContext())

{

Database.SetInitializer(new DbDeleteandInit());

Console.WriteLine("Удаление и создание базы");

//Железобетон

db.Products.Add(new Product { Name = "ФБС - фундаментные блоки сплошные", NameGroup = Namegroup.IronBeton, Price = 250, Count = 100 });

db.Products.Add(new Product { Name = "ФЛ - фундаментная подушка", NameGroup = Namegroup.IronBeton, Price = 2000, Count = 100 });

db.Products.Add(new Product { Name = "П - плиты перекрытий и покрытий", NameGroup = Namegroup.IronBeton, Price = 2200, Count = 100 });

db.Products.Add(new Product { Name = "ПЛ - плиты лоджий", NameGroup = Namegroup.IronBeton, Price = 1900, Count = 100 });

db.Products.Add(new Product { Name = "ПП - плиты парапетные", NameGroup = Namegroup.IronBeton, Price = 220, Count = 100 });

//асфальтобетон

db.Products.Add(new Product { Name = "Литой асфальтобетон", NameGroup = Namegroup.AsphaltBeton, Price = 5500, Count = 100 });

db.Products.Add(new Product { Name = "Горячие и теплые смеси", NameGroup = Namegroup.AsphaltBeton, Price = 10000, Count = 100 });

db.Products.Add(new Product { Name = "Холодный асфальтобетон", NameGroup = Namegroup.AsphaltBeton, Price = 4500, Count = 100 });

//керамзитобетон

db.Products.Add(new Product { Name = "Конструктивный", NameGroup = Namegroup.KeramzBeton, Price = 2505, Count = 100 });

db.Products.Add(new Product { Name = "Теплоизоляционный", NameGroup = Namegroup.KeramzBeton, Price = 3000, Count = 100 });

db.Products.Add(new Product { Name = "теплоизоляционно-конструктивный", NameGroup = Namegroup.KeramzBeton, Price = 5600, Count = 100 });

db.SaveChanges();

//покупки

Random rnd = new Random();

List<string> fio = new List<string>();

fio.Add("Масленников Сергей Андреевич");

fio.Add("Бальзамов Александр Витальевич");

fio.Add("Кольченко Артём Витальевич");

fio.Add("Арташкин Евгений Павлович");

fio.Add("Денискин Александр Александрович");

//Имена пользователей

DateTime st = new DateTime(2016,1,1);

DateTime en = new DateTime(2017,1,1);

for (var it = 0; it < 1000; it++)

{

Console.WriteLine(it);

var tempid = rnd.Next(1, 12);

var tempcount = rnd.Next(25, 40);

db.Purchases.Add(new Purchase { Fio = fio[rnd.Next(0, 5)], IdProduct = tempid, Price = db.Products.Find(tempid).Price \* tempcount, Counttovar = tempcount, Date =GetRandomDate(st,en) });

}

//пользователи Admin

db.UsersAdmin.Add(new User { Login = "sergey", Password = "yeti" });

db.SaveChanges();

Console.WriteLine("Изменения сохранены");

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

static readonly Random rnd = new Random();

public static DateTime GetRandomDate(DateTime from, DateTime to)

{

var range = to - from;

var randTimeSpan = new TimeSpan((long)(rnd.NextDouble() \* range.Ticks));

return from + randTimeSpan;

}

}

public class DbDeleteandInit : DropCreateDatabaseAlways<BaseContext>

{

protected override void Seed(BaseContext db)

{

base.Seed(db);

}

}

public class DbInit : DropCreateDatabaseIfModelChanges<BaseContext>

{

protected override void Seed(BaseContext db)

{

base.Seed(db);

}

}

}

ClientWebApi — это приложение, ориентированное на пользователей. С помощью него клиенты завода будут покупать товары.

Вначале клиенту необходимо определиться с выбором товара, для этого я создал представление Index.

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<link rel="stylesheet" href="~/Content/bootstrap.min.css" />

<link rel="stylesheet" href="~/Content/app/MainIndexCss.css" />

</head>

<body><h1>Товары бетонного завода</h1>

<div class="flex-container">

<label><img src="~/fonts/asfbeton.png" /><a href="/Home/Buy/1" target="\_blank">Железобетон</a></label>

<label><img src="~/fonts/Beton.jpg" /><a href="/Home/Buy/2" target="\_blank">Асфальтобетон</a></label>

<label><img src="~/fonts/keramzit.jpg" /><a href="/Home/Buy/3" target="\_blank">Керамзитобетон</a></label>

</div>

</body>

</html>

Данная форма представлена на рисунке 6.

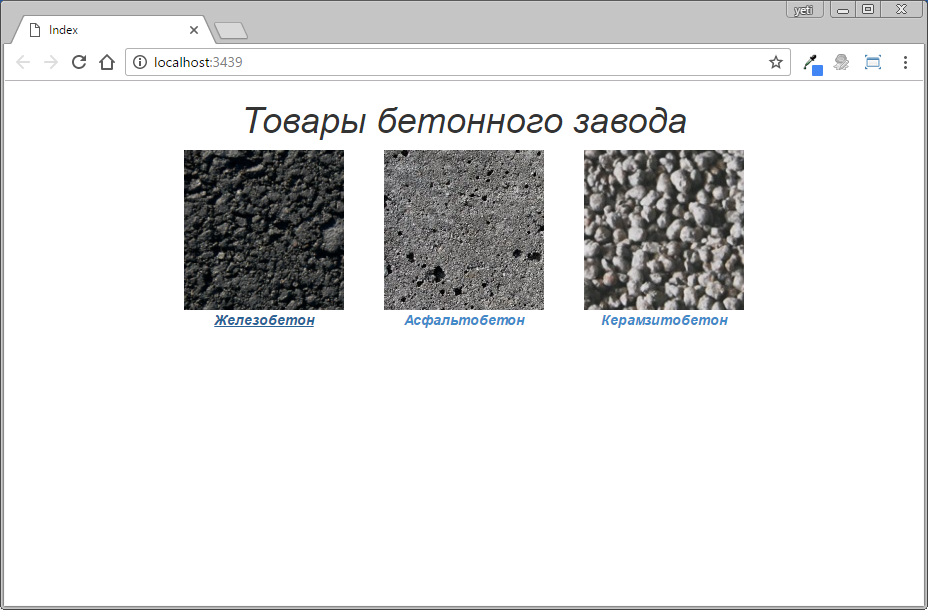


Рисунок 6 —

Далее для покупки и ввода персональных данных пользователя я сделал представление Buy, которое показано на рисунке 7. Чтобы следить за полями форм я использовал Knockout.js.

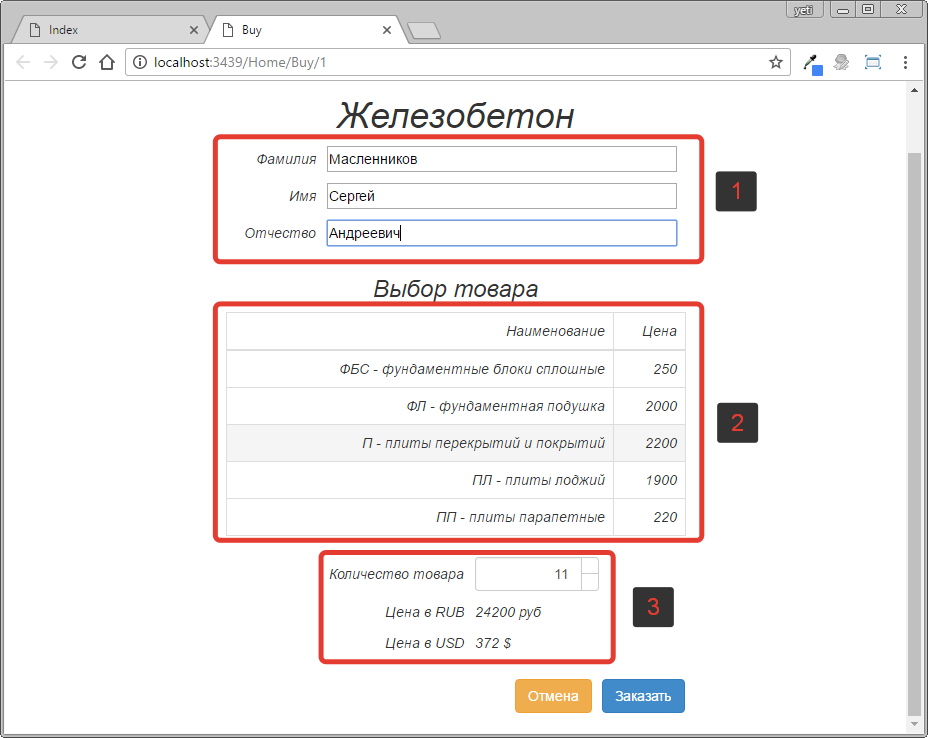


Рисунок 7 —

1. Поля для ввода фамилии, имени и отчества. Для идентификации пользователя.
2. Выбор конкретного товара
3. Выбор количества товара.

Knockout.js — это [свободный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) JavaScript каркас веб-приложений, реализующий Model-View-ViewModel шаблон c образцами. Основными принципами являются:

* чёткое разделение доменных данных, компонентов отображения и отображаемых данных
* наличие чётко определённого слоя специализированного кода, задающего отношение компонентов отображения.

Основной JavaScript код написан в файле buycustom.

var ViewModel = {

ListProducts: ko.observableArray(),

Name: ko.observable(),

f: ko.observable(""),

i: ko.observable(""),

o: ko.observable(""),

Price: ko.observable(),

actioncontroller: ko.observable(),

MainVid: ko.observable(),

priceRUB: ko.observable(0),

priceUSD: ko.observable(0)

}

function Connect() {

$('#changetable > tbody > tr').click(function () {

if ($(this).hasClass('selected1')) {

$('tbody tr').removeClass('selected1');

IdProd = "0";

Price = "0";

} else {

$('tbody tr').removeClass('selected1');

$(this).addClass('selected1');

IdProd = this.id;

Price = this.innerHTML.match(/Price\"\S+</)[0].replace("Price\">", "").replace("<", "");

ViewModel.priceRUB(0);

ViewModel.priceUSD(0);

$('.spinner input').val(0);

}

});

};

function GetTovarsById(id) {

$.ajax({

url: '/api/BuyProd/?id=' + id,

contentType: 'application/json; charset=UTF-8',

success: function (data) {

ViewModel.ListProducts(data);

Connect();

},

error: function (e) {

alert(e.responseText);

}

});

}

function GetCurs() {

$.ajax({

url: '/api/BuyProd/',

contentType: 'application/json; charset=UTF-8',

success: function (data) {

DollarCurs = data;

},

error: function (e) {

alert(e.responseText);

}

});

}

$(function () {

ko.applyBindings(ViewModel);

window.IdProd = "0";

window.Price = 0;

var url = location.href;

var array = url.split('/');

window.DollarCurs;

GetCurs();

GetTovarsById(array[array.length - 1]);

if (array[array.length - 1] == "1") {

ViewModel.MainVid("Железобетон");

}

if (array[array.length - 1] == "2") {

ViewModel.MainVid("Асфальтобетон");

}

if (array[array.length - 1] == "3") {

ViewModel.MainVid("Керамзитобетон");

}

$('.spinner .btn:first-of-type').on('click', function () {

if (IdProd !== "0") {

$('.spinner input').val(parseInt($('.spinner input').val(), 10) + 1);

ViewModel.priceRUB(Number($('.spinner input').val(), 10) \* Number(Price.toString(), 10) + " руб");

ViewModel.priceUSD(parseInt(parseInt($('.spinner input').val(), 10) \* parseInt(Price, 10) / parseInt(DollarCurs, 10), 10) + " $");

}

});

$('.spinner .btn:last-of-type').on('click', function () {

if (IdProd !== "0") {

if (parseInt($('.spinner input').val(), 10) - 1 > 0) {

$('.spinner input').val(parseInt($('.spinner input').val(), 10) - 1);

ViewModel.priceRUB(parseInt($('.spinner input').val(), 10) \* parseInt(Price, 10) + " руб");

ViewModel.priceUSD(parseInt(parseInt($('.spinner input').val(), 10) \* parseInt(Price, 10) / parseInt(DollarCurs, 10), 10) + " $");

}

}

});

$("#cancele").click(function () {

window.open("/Home/Index", "\_self");

});

$("#buytovar").click(function () {

if (IdProd !== "0" && $("#spinnercount").val() !== "0") {

var pur = {

Fio: ViewModel.f() + " " + ViewModel.i() + " " + ViewModel.o(),

Price: parseInt(ViewModel.priceRUB()),

Counttovar: $("#spinnercount").val(),

Date: null,

IdProduct: parseInt(IdProd, 10)

}

$.ajax({

type: 'POST',

url: '/api/BuyProd',

data: JSON.stringify(pur),

contentType: 'application/json; charset=UTF-8',

success: function (data) {

ViewModel.actioncontroller(data);

$("#continuebuy").hide();

$("#Modal\_Returner\_Success").modal("show");

},

error: function (e) {

ViewModel.actioncontroller(e.responseText);

$("#Modal\_Returner\_Success").modal("show");

}

});

} else {

ViewModel.actioncontroller("Заполните все поля!");

$("#Modal\_Returner\_Success").modal("show");

}

});

});

Для взаимодействия формы с базой данных я использовал ApiController BuyProdController. В нём также написан метод Get(), который возвращает текущий курс доллара.

using Data;

using System;

using System.IO;

using System.Linq;

using System.Web.Http;

using System.Xml;

namespace ClientWebApi.Controllers

{

public class BuyProdController : ApiController

{

[HttpGet]

public IHttpActionResult Get()

{

try

{

System.Net.WebRequest req = System.Net.WebRequest.Create("http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml");

System.Net.WebResponse resp = req.GetResponse();

Stream stream = resp.GetResponseStream();

StreamReader sr = new StreamReader(stream);

string xmlString = sr.ReadToEnd();

XmlDocument xml = new XmlDocument();

xml.LoadXml(xmlString);

XmlNamespaceManager nsmgr = new XmlNamespaceManager(xml.NameTable);

nsmgr.AddNamespace("ecb", "http://www.ecb.int/vocabulary/2002-08-01/eurofxref");

nsmgr.AddNamespace("gesmes", "http://www.gesmes.org/xml/2002-08-01");

XmlNode currencyNode = xml.SelectSingleNode("descendant::ecb:Cube[@currency='RUB']", nsmgr);

string rate = currencyNode.Attributes.GetNamedItem("rate").Value.Trim().Replace(".", ",");

var drate = Convert.ToInt32(Math.Round(float.Parse(rate)));

sr.Close();

return Ok(drate);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

[HttpGet]

public IHttpActionResult Get([FromUri]int id)

{

if (id == 1)

{

using (BaseContext db = new BaseContext())

{

var arr = db.Products

.Where(e => e.NameGroup == Namegroup.IronBeton).Select(e => new { IdProd = e.IdProduct, Name = e.Name, Price = e.Price }).ToList();

return Ok(arr);

};

}

if (id == 2)

{

using (BaseContext db = new BaseContext())

{

var arr = db.Products

.Where(e => e.NameGroup == Namegroup.AsphaltBeton).Select(e => new { IdProd = e.IdProduct, Name = e.Name, Price = e.Price }).ToList();

return Ok(arr);

};

}

if (id == 3)

{

using (BaseContext db = new BaseContext())

{

var arr = db.Products

.Where(e => e.NameGroup == Namegroup.KeramzBeton).Select(e => new { IdProd = e.IdProduct, Name = e.Name, Price = e.Price }).ToList();

return Ok(arr);

};

}

return BadRequest("Товар не найден");

}

// POST: api/BuyProd

public IHttpActionResult Post([FromBody]Purchase pur)

{

using (BaseContext db = new BaseContext())

{

try

{

pur.Date = DateTime.Now;

db.Purchases.Add(pur);

var temp = db.Products.Find(pur.IdProduct);

if (temp.Count - pur.Counttovar > 0)

{

temp.Count = temp.Count - pur.Counttovar;

db.SaveChanges();

return Ok("Заказ успешно совершён!");

}

else {

return BadRequest("Выбранного товара недостаточно на складе. Масимальное количество товара: " + temp.Count);

}

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

}

}

}

Admin — это приложение для администрации завода. Главная форма приложения показана на рисунке 8.

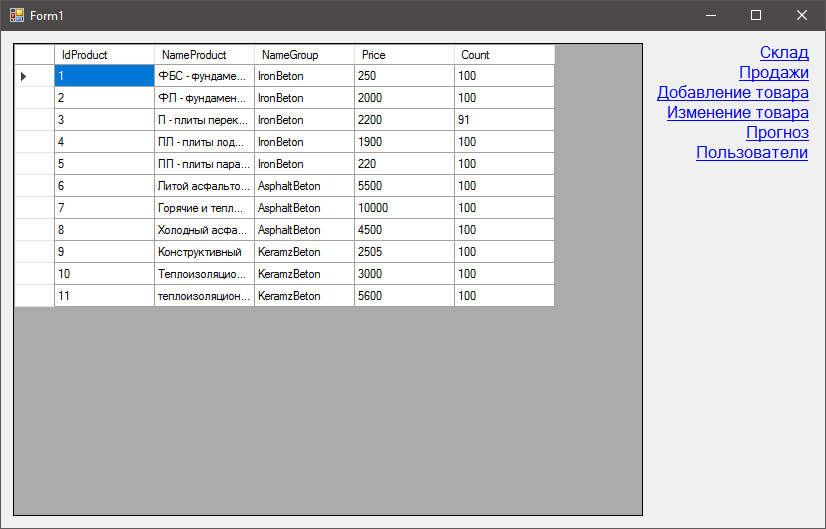


Рисунок 8 —

Form1 содержит панель навигации, из нее можно перейти к добавлению или редактированию товара (предварительно выбрав нужный товар из таблицы). Примерное использование формы Sales показано на рисунке 9.

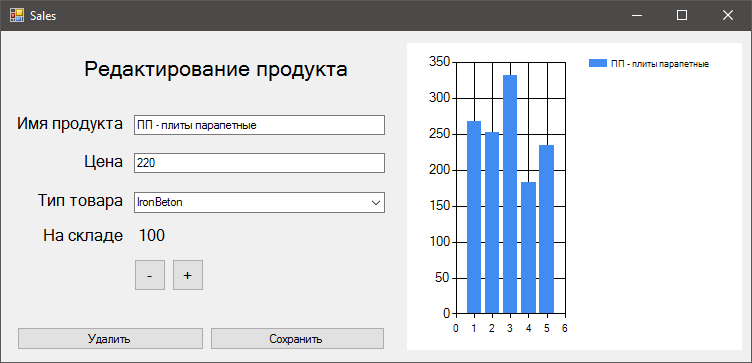


Рисунок 9 —

Слева на форме расположены основные данные продукта, которые при желании можно отредактировать. Справа расположен график продаж товара по месяцам.

Основной код формы Sales:

using Data;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace Admin

{

public partial class Sales : Form

{

private int IdentNum;

public Sales(int Id)

{

IdentNum = Id;

InitializeComponent();

comboBox1.DataSource = System.Enum.GetNames(typeof(Namegroup)).ToList(); ;

if (Id == -1)

{

label4.Text = "Создание продукта";

button2.Text = "Закрыть";

}

else {

button2.Text = "Удалить";

label4.Text = "Редактирование продукта";

using (BaseContext db = new BaseContext())

{

var temp = db.Products.Find(Id);

textBox1.Text = temp.Name;

textBox2.Text = temp.Price.ToString();

comboBox1.SelectedIndex = temp.NameGroup.GetHashCode();

label5.Text = temp.Count.ToString();

//статистика

var list = db.Purchases.Where(p => p.IdProduct == IdentNum).ToList();

var temparr = from i in list

group i by new { i.Date.Month,i.Date.Year } into grp

select new { Month = grp.Key, Count = grp.Sum(i => i.Counttovar) };

var res = temparr.ToList().OrderBy(u => u.Month.Month).OrderBy(u => u.Month.Year).Take(5);

Series series = chart1.Series.Add(temp.Name);

foreach (var u in res)

{

series.Points.Add(u.Count);

}

}

}

}

private void button1\_Click(object sender, System.EventArgs e)

{

if (IdentNum == -1)

{

using (BaseContext db = new BaseContext())

{

db.Products.Add(new Product

{

Name = textBox1.Text,

Price = int.Parse(textBox2.Text),

NameGroup = (Namegroup)comboBox1.SelectedIndex,

Count = int.Parse(label5.Text)

});

db.SaveChanges();

}

Close();

Form1.selfRef.UpdateGrid();

}

else {

using (BaseContext db = new BaseContext())

{

var temp = db.Products.Find(IdentNum);

temp.Name = textBox1.Text;

temp.Price = int.Parse(textBox2.Text);

temp.NameGroup = (Namegroup)comboBox1.SelectedIndex;

temp.Count = int.Parse(label5.Text);

db.SaveChanges();

}

Close();

Form1.selfRef.UpdateGrid();

}

}

private void button2\_Click(object sender, System.EventArgs e)

{

if (IdentNum == -1)

{

Close();

}

else {

using (BaseContext db = new BaseContext())

{

Product temp = db.Products.First(x => x.IdProduct == IdentNum);

db.Products.Remove(temp);

db.SaveChanges();

}

Close();

Form1.selfRef.UpdateGrid();

}

}

private void button4\_Click(object sender, System.EventArgs e)

{

label5.Text = (int.Parse(label5.Text)+1).ToString();

}

private void button3\_Click(object sender, System.EventArgs e)

{

label5.Text = (int.Parse(label5.Text) - 1).ToString();

}

}

}

Из Form1 можно перейти на форму предсказаний HintForm, она показана на рисунке 10.

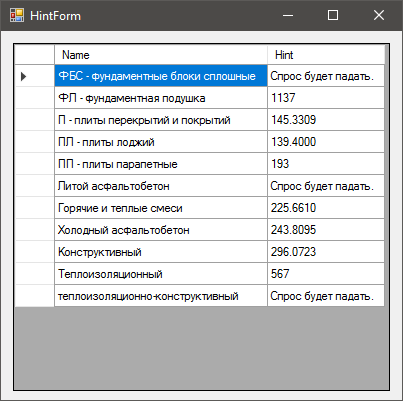


Рисунок 10 —

Основной код формы HintForm:

using System.Windows.Forms;

using MathWorks.MATLAB.NET.Arrays;

using hint;

using System;

using Data;

using System.Linq;

using System.Collections.Generic;

namespace Admin

{

public partial class HintForm : Form

{

public HintForm()

{

InitializeComponent();

ProgressForm.selfRef.SetProgress(10);

MLHint res = new MLHint();

ProgressForm.selfRef.SetProgress(30);

List<HelperTable> table = new List<HelperTable>();

using (BaseContext db = new BaseContext())

{

var tempprod = db.Products;

int incofprogr = 60/tempprod.Count();

foreach (var io in tempprod)

{

var list = db.Purchases.Where(p => p.IdProduct == io.IdProduct).ToList();

var temparr = from i in list

group i by new { i.Date.Month, i.Date.Year } into grp

select new

{

Month = grp.Key,

Count = grp.Sum(i => i.Counttovar)

};

int[] reslinq = temparr

.ToList()

.OrderBy(u => u.Month.Month)

.OrderBy(u => u.Month.Year)

.Select(u => u.Count)

.ToArray();

int[] tt1 = new int[reslinq.Count()];

for (var it = 0; it < reslinq.Count(); it++)

{

tt1[it] = it+1;

}

MWArray mas = null;

MWNumericArray arr1 = tt1;

mas = arr1;

MWArray mas2 = null;

MWNumericArray arr2 = reslinq;

mas2 = arr2;

try

{

var tempres = res.hint(mas, mas2).ToString();

if (tempres.Contains('-') == true)

{

table.Add(new HelperTable(io.Name, "Спрос будет падать."));

}

else {

table.Add(new HelperTable(io.Name, tempres));

}

}

catch (Exception e)

{

MessageBox.Show(e.Message);

}

ProgressForm.selfRef.SetProgress(incofprogr);

}

}

dataGridView1.DataSource = table;

ProgressForm.selfRef.SetAll();

ProgressForm.selfRef.ShowHideProgress();

}

private class HelperTable

{

public HelperTable(string nm, string hnt)

{

Name = nm;

Hint = hnt;

}

public string Name { get; set; }

public string Hint { get; set; }

}

}

}

**Заключение**

1 —

2 —