Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

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

Факультет информационных технологий

Кафедра прикладной математики

(наименование кафедры)

Отчет защищен с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Шальнев (подпись) (и.о., фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г.

(дата)

Расчетное задание

по теме Проектирование и реализация системы учета рабочего времени

Пояснительная записка

по дисциплине

«Сервис-ориентированные технологии разработки программных систем»

наименование дисциплины

РЗ 09.04.04.27.000 ПЗ

обозначение документа

Работу выполнил

студент группы 8ПИ-61 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.О. Корней

(подпись) (и.о., фамилия)

Руководитель

проекта старший преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Шальнев

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

Барнаул 2016

# Реферат

Работа посвящена проектированию и реализации системы ведения домашней бухгалтерии. В основе системы лежит трехуровневая архитектура. Рассматриваются этапы построения слоя данных, бизнес-логики и клиентской части.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

РЗ 09.04.04.27.000 ПЗ

Разраб.

Корней А.О.

Пров.

Шальнев А.А.

Реценз.

Н. контр.

Утв.

Кантор С.А.

Система управления домашней библиотекой

Лит.

Листов

АлтГТУ ФИТ гр. 8ПИ-61

У 2 35

Содержание

[Введение 6](#_Toc469696225)

[1 Проектирование базы данных 7](#_Toc469696226)

[2 Разработка ПО 8](#_Toc469696227)

[2.1 Проектирование модели и слоя данных 8](#_Toc469696228)

[2.2 Проектирование бизнес-логики 11](#_Toc469696229)

[2.2.1 Доступ к БД 11](#_Toc469696230)

[2.2.2 Бизнес-логика 11](#_Toc469696231)

[2.2.3 Запуск web-сервиса 15](#_Toc469696232)

[2.3 Создание клиентской части 18](#_Toc469696233)

[Заключение 34](#_Toc469696234)

[Список использованных источников 35](#_Toc469696235)

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

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

УТВЕРЖДАЮ

Заведующий кафедрой ПМ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** 2016 г.

ЗАДАНИЕ№27

|  |  |
| --- | --- |
| По направлению | «Программная инженерия» |
| Студенту группы | 8ПИ-61 |
|  | Корней Алене Олеговне |
|  | фамилия, имя, отчество |

|  |  |
| --- | --- |
| Тема: | «Система ведения домашней бухгалтерии» |
|  | |
|  | |

Срок исполнения расчетного задания 17.12.2016

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Корней А.О. /

подпись фамилия, имя, отчество

БАРНАУЛ 2016

1 Исходные данные

Для полученной темы разработать программное обеспечение, построенное на трехуровневой архитектуре, включающей в себя отдельные слои для доступа к данным, бизнес-логики и клиентской части.

2 Содержание разделов пояснительной записки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование разделов | Содержание работ по разделу | Трудоемкость, % | Срок выпол  нения | Консультант (Ф.И.О., подпись) |
| 1 | 2 | 3 | 4 | 5 |
| Введение | Кратко изложить теоретические основы трехуровневой архитектуры |  |  | Шальнев А.А. |
| Проектирование БД | Спроектировать базу данных для полученной темы. Реализовать спроектированную базу в выбранной СУБД, отобразить ее на программную модель данных. |  |  | Шальнев А.А. |
| Разработка ПО | Разработать и реализовать классы для доступа к данным. Реализовать бизнес-логику. Создать web-service для удаленного доступа к данным. Реализовать клиентскую часть. |  |  | Шальнев А.А. |
| Заключение | Общие выводы по работе. |  |  | Шальнев А.А. |

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шальнев А.А.

подпись Фамилия И. О.

# Введение

В последнее время люди стали внимательнее относиться к собственным тратам. Приложения для ведения домашней бухгалтерии весьма популярны, и пользователю есть из чего выбирать. Однако все они обладают схожими чертами – позволяют распределять доходы и траты по различным категориям, вести статистику, создавать простейшие отчеты, на основании которых можно сделать выводы о наиболее затратных статьях семейного бюджета.

Для удобства ведения домашней бухгалтерии необходимо, чтобы пользователь мог вносить данные о покупках с различных устройств, то есть речь в данном случае идет о возможности создания различных клиентских приложений для доступа к единственной базе, содержащей данные определенного пользователя.

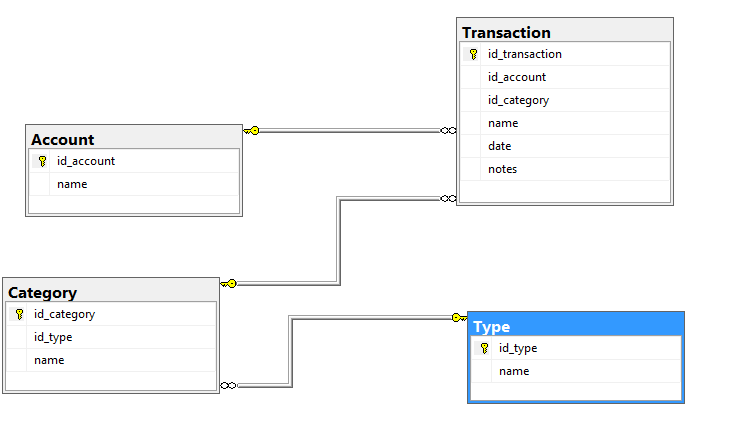
Вот почему систему для ведения домашней бухгалтерии лучше всего строить на основе трехуровневой архитектуры. При использовании такого подхода система получается гибкой, масштабируемой. Отслеживание и исправление ошибок, создание новых клиентских приложений становится простым и понятным.

Работа посвящена проектированию и реализации системы ведения домашней бухгалтерии. Последовательно рассмотрены этапы проектирования базы данных, слоя доступа к данным, бизнес-логики и клиентской части.

# 1 Проектирование базы данных

В рамках данной работы предполагается вести учет доходов и расходов в виде журнала транзакций. Каждая транзакция относится к определенной категории и связана со счетом, с которого она выполнена.

В качестве СУБД выбран MSSQL 2016 Enterprise. Схема базы данных приведена на рис.1.



*Рисунок 1 – схема базы данных для системы ведения домашней бухгалтерии*

# 2 Разработка ПО

## 2.1 Проектирование модели и слоя данных

Для работы с базой данных было решено применить ORM Entity Framework. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции, создавать и сопровождать приложения, ориентированные на данные, используя меньше кода, чем в традиционных приложениях.  Поскольку Entity Framework является компонентом .NET Framework, приложения Entity Framework могут работать на любом компьютере, где установлена платформа .NET Framework, начиная с версии 3.5 с пакетом обновления 1 (SP1).

Для корректной работы с данными с использованием Entity Framework были написаны классы модели. Для отображения на реальную базу использовались атрибуты DataAnnotations. Получение соответсвующего программного кода возможно и с помощью генератора, однако в рамках данной работы было решено реализовать модель вручную для более полного понимания принципов работы Entity Framework.

В связи с тем, что для создания web-сервиса будет использована платформа WCF, а модель необходимо использовать как на стороне сервера, так и на стороне клиента, классы модели и их члены имеют дополнительные атрибуты. Эти атрибуты необходимы для того, чтобы клиент «знал» классы модели, известные серверу и «умел» их сериализовать и десериализовать при передаче данных.

Атрибуты [Key], [Column], [ForeignKey] используются для отображения модели на реальную базу данных. Их значение соответсвует терминам проектирования БД. Однако, данные атрибуты не привязаны к конкретной СУБД и необходимы для формирования логической модели данных.

Атрибуты [DataContract], [DataMember] использованы для работы через WCF-сервис.

**Программный код:**

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Runtime.Serialization;

namespace ExpenseTracker.Model

{

[Table("Type")]

[DataContract]

public class CategoryType

{

[Key]

[Column("id\_type")]

[DataMember]

public int IdType { get; set; }

[Column("name")]

[DisplayName("Type name")]

[DataMember]

public string Name { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace ExpenseTracker.Model

{

[Table("Category")]

[DataContract]

public class Category

{

[Key]

[Column("id\_category")]

[DataMember]

public int IdCategory { get; set; }

[Column("id\_type")]

[DataMember]

public int FKType { get; set; }

[Column("name")]

[DisplayName("Category name")]

[DataMember]

public string Name { get; set; }

[ForeignKey("FKType")]

[DisplayName("Type")]

[DataMember]

public CategoryType Type { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Runtime.Serialization;

using System.ComponentModel;

namespace ExpenseTracker.Model

{

[Table("Account")]

[DataContract]

public class Account

{

[Key]

[Column("id\_account")]

[DataMember]

public int IdAccount { get; set; }

[Required]

[MaxLength(300)]

[Column("name")]

[DisplayName("Account name")]

[DataMember]

public string Name { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace ExpenseTracker.Model

{

[Table("Transaction")]

[DataContract]

public class Transaction

{

[Key]

[Column("id\_transaction")]

[DataMember]

public int IdTransaction { get; set; }

[Column("id\_account")]

[DataMember]

public int FKAccount { get; set; }

[Column("id\_category")]

[DataMember]

public int FkCategory { get; set; }

[Column("name")]

[DataMember]

public string Name { get; set; }

[Column("date")]

[DataMember]

public System.DateTime Date { get; set; }

[Column("notes")]

[DataMember]

public string Notes { get; set; }

[ForeignKey("FKAccount")]

[DisplayName("Account")]

[DataMember]

public Account Account { get; set; }

[ForeignKey("FkCategory")]

[DisplayName("Category")]

[DataMember]

public Category Category { get; set; }

}

}

## 2.2 Проектирование бизнес-логики

### 2.2.1 Доступ к БД

Для работы с базой данных в Entity Framework принято создавать DbContext. При создании контекста в конструктор базового класса передается имя строки подключения (ConnectionString). Строка с таким именем должна обязательно быть в конфигурационном файле приложения, в рамках которого запускается веб-сервис.

**Программный код:**

using ExpenseTracker.Model;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

namespace ExpenseTracker.BusinessLogic

{

public class ExpenseTrackerContext:DbContext

{

public ExpenseTrackerContext() : base("name=ExpenseTracker")

{

}

public virtual DbSet<Account> Accounts { get; set; }

public virtual DbSet<Category> Categories { get; set; }

public virtual DbSet<Transaction> Transactions { get; set; }

public virtual DbSet<CategoryType> Types { get; set; }

}

}

### 2.2.2 Бизнес-логика

В данной работе использован веб-сервис на базе технологии WCF. Учитывая специфику данной технологии можно реализовать бизнес-логику приложения непосредственно в коде сервиса, а затем пометить необходимые методы атрибутом [OperationContract], а сам класс – атрибутом [ServiceContract]. Без использования данных атрибутов получим обычный слой бизнес-логики.

Entity Framework предоставляет широкие возможности для удобной работы с данными. В частности, есть возможность использовать LINQ, что значительно повышает скорость написания запросов и удобство чтения кода.

Ниже приведен код сервиса.

**Программный код:**

using ExpenseTracker.BusinessLogic;

using ExpenseTracker.Model;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace ExpenseTracker.Services

{

[ServiceContract]

public class ExpenseTrackerService

{

#region CategoryType

[OperationContract]

public IList<CategoryType> GetCategoryTypes()

{

using (var db = new ExpenseTrackerContext())

{

return db.Types.ToList();

}

}

[OperationContract]

public CategoryType GetCategoryTypeById(int id)

{

using (var db = new ExpenseTrackerContext())

{

var categoryType = db.Types.Find(id);

return categoryType;

}

}

[OperationContract]

public void AddCategoryType(CategoryType categoryType)

{

using (var db = new ExpenseTrackerContext())

{

db.Types.Add(categoryType);

db.SaveChanges();

}

}

[OperationContract]

public void UpdateCategoryType(CategoryType categoryType)

{

using (var db = new ExpenseTrackerContext())

{

db.Entry(categoryType).State = EntityState.Modified;

db.SaveChanges();

}

}

[OperationContract]

public void DeleteCategoryTypeById(int id)

{

using (var db = new ExpenseTrackerContext())

{

var categoryType = db.Types.Find(id);

db.Types.Remove(categoryType);

db.SaveChanges();

}

}

#endregion

#region Category

[OperationContract]

public IList<Category> GetCategories()

{

using (var db = new ExpenseTrackerContext())

{

var categories = db.Categories.Include(c => c.Type);

return categories.ToList();

}

}

[OperationContract]

public Category GetCategoryById(int id)

{

using (var db = new ExpenseTrackerContext())

{

var category = db.Categories.Find(id);

return category;

}

}

[OperationContract]

public void AddCategory(Category category)

{

using(var db = new ExpenseTrackerContext())

{

db.Categories.Add(category);

db.SaveChanges();

}

}

[OperationContract]

public void UpdateCategory(Category category)

{

using (var db = new ExpenseTrackerContext())

{

db.Entry(category).State = EntityState.Modified;

db.SaveChanges();

}

}

[OperationContract]

public void DeleteCategoryById(int id)

{

using (var db = new ExpenseTrackerContext())

{

Category category = db.Categories.Find(id);

db.Categories.Remove(category);

db.SaveChanges();

}

}

#endregion

#region Transaction

[OperationContract]

public IList<Transaction> GetTransactions()

{

using (var db = new ExpenseTrackerContext())

{

var transactions = db.Transactions.Include(t => t.Account).Include(t => t.Category);

return transactions.ToList();

}

}

[OperationContract]

public Transaction GetTransactionById(int id)

{

using (var db = new ExpenseTrackerContext())

{

Transaction transaction = db.Transactions.Find(id);

return transaction;

}

}

[OperationContract]

public void AddTransaction(Transaction transaction)

{

using (var db = new ExpenseTrackerContext())

{

db.Transactions.Add(transaction);

db.SaveChanges();

}

}

[OperationContract]

public void UpdateTransaction(Transaction transaction)

{

using(var db = new ExpenseTrackerContext())

{

db.Entry(transaction).State = EntityState.Modified;

db.SaveChanges();

}

}

[OperationContract]

public void DeleteTransactionById(int id)

{

using (var db = new ExpenseTrackerContext())

{

Transaction transaction = db.Transactions.Find(id);

db.Transactions.Remove(transaction);

db.SaveChanges();

}

}

#endregion

#region Account

[OperationContract]

public IList<Account> GetAccounts()

{

using (var db = new ExpenseTrackerContext())

{

return db.Accounts.ToList();

}

}

[OperationContract]

public Account GetAccountById(int id)

{

using (var db = new ExpenseTrackerContext())

{

Account account = db.Accounts.Find(id);

return account;

}

}

[OperationContract]

public void AddAccount(Account account)

{

using (var db = new ExpenseTrackerContext())

{

db.Accounts.Add(account);

db.SaveChanges();

}

}

[OperationContract]

public void UpdateAccount(Account account)

{

using (var db = new ExpenseTrackerContext())

{

db.Entry(account).State = EntityState.Modified;

db.SaveChanges();

}

}

[OperationContract]

public void DeleteAccountById(int id)

{

using (var db = new ExpenseTrackerContext())

{

Account account = db.Accounts.Find(id);

db.Accounts.Remove(account);

db.SaveChanges();

}

}

#endregion

}

}

### 2.2.3 Запуск web-сервиса

WCF сервис можно запускать под управлением IIS или с использованием WAS (Windows Activation Service). Поскольку возможности IIS избыточны для данной работы, наиболее приемлемым было бы размещение веб-сервиса в виде службы windows. Однако в рамках выполения рассчетного задания было решено запускать WCF-сервис с использованием консольного приложения, код которого приведен ниже:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace ExpenseTracker.Service.ConsoleHost

{

class Program

{

static void Main(string[] args)

{

var expensesHost = new ServiceHost(typeof(ExpenseTracker.Services.ExpenseTrackerService));

expensesHost.Open();

Console.WriteLine("serviceStarted");

Console.ReadKey();

expensesHost.Close();

}

}

}

Для корректной работы сервиса необходим конфигурационный файл, содержащий необходимую информацию. Для подключения к нужной базе данных необходимо создать именованную строку подключения, указав так же имя сборки, которая соответствует нужному провайдеру. Результаты запуска сервиса представлены на рис.2-3

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

</configSections>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />

</startup>

<connectionStrings>

<add name="ExpenseTracker" providerName="System.Data.SqlClient" connectionString="Server=.\;Database=ExpenseTracker;Integrated Security=True;" />

</connectionStrings>

<system.serviceModel>

<services>

<service name="ExpenseTracker.Services.ExpenseTrackerService" behaviorConfiguration="basicBehaviorConfiguration">

<host>

<baseAddresses>

<add baseAddress="http://localhost:9999/ExpenseTrackerService" />

</baseAddresses>

</host>

<endpoint address="" binding="basicHttpBinding" contract="ExpenseTracker.Services.ExpenseTrackerService">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

</service>

</services>

<bindings>

<!--<webHttpBinding>

<binding name="webHttpBindingWoSecurity" maxReceivedMessageSize="10485760"/>

</webHttpBinding>-->

</bindings>

<behaviors>

<serviceBehaviors>

<behavior name="basicBehaviorConfiguration">

<serviceMetadata httpGetEnabled="True" />

<serviceDebug includeExceptionDetailInFaults="True" />

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

<parameters>

<parameter value="mssqllocaldb" />

</parameters>

</defaultConnectionFactory>

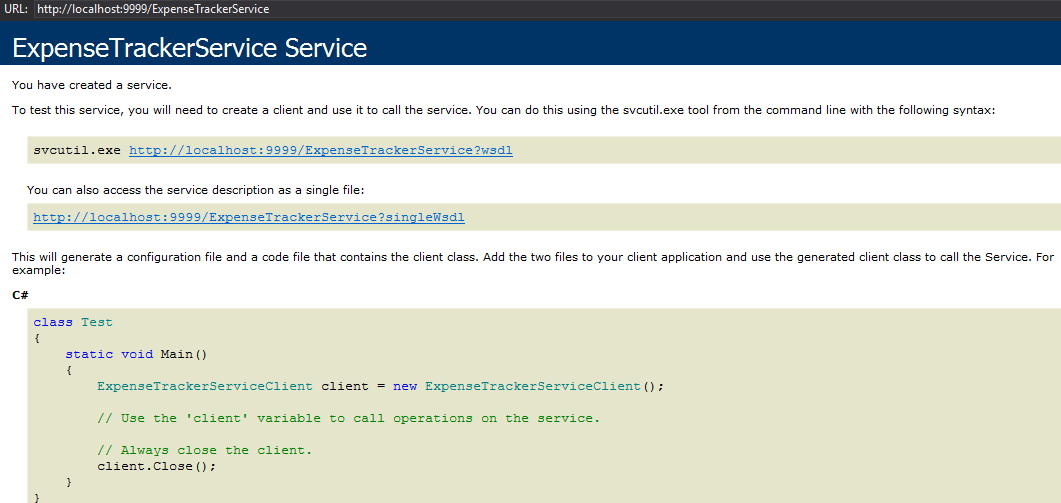
<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

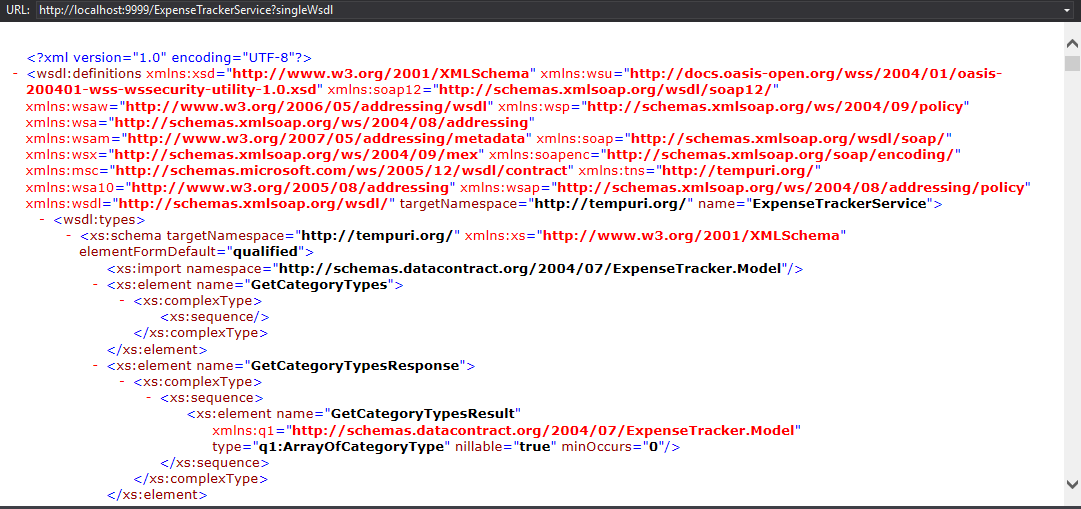
</providers>

</entityFramework>

</configuration>



*Рисунок 2 – результат обращения к WCF-сервису*



*Рисунок 3 – предоставленные WCF-сервисом метаданные*

# 2.3 Создание клиентской части

Для создания клиентской части было решено использовать ASP.NET MVC. Создание веб-клиента для ведения домашней бухгалтерии является хорошим решением, поскольку гарантирует пользователю доступ к журналу доходов и расходов с любого устройства без установки дополнительных приложений и пакетов.

Клиентская часть работает с веб-сервисом (контроллеры MVC делегируют сервису запросы на чтение/модификацию данных).

Код контроллеров, а так же разметка, использованная для создания представлений, приведены ниже.

**Программный код:**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using ExpenseTracker.Model;

namespace ExpenseTracker.Server.MVC.Controllers

{

public class TransactionsController : Controller

{

//private ExpenseTrackerContext db = new ExpenseTrackerContext();

private ExpensesService.ExpenseTrackerServiceClient \_serviceClient

= new ExpensesService.ExpenseTrackerServiceClient();

// GET: Transactions

public ActionResult Index()

{

var transactions = \_serviceClient.GetTransactions();//db.Transactions.Include(t => t.Account).Include(t => t.Category);

return View(transactions.ToList());

}

// GET: Transactions/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Transaction transaction = \_serviceClient.GetTransactionById(id.Value);//db.Transactions.Find(id);

if (transaction == null)

{

return HttpNotFound();

}

return View(transaction);

}

// GET: Transactions/Create

public ActionResult Create()

{

ViewBag.FKAccount = new SelectList(\_serviceClient.GetAccounts(), "IdAccount", "Name");

ViewBag.FkCategory = new SelectList(\_serviceClient.GetCategories(), "IdCategory", "Name");

return View();

}

// POST: Transactions/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "IdTransaction,FKAccount,FkCategory,Name,Date,Notes")] Transaction transaction)

{

if (ModelState.IsValid)

{

//db.Transactions.Add(transaction);

//db.SaveChanges();

\_serviceClient.AddTransaction(transaction);

return RedirectToAction("Index");

}

ViewBag.FKAccount = new SelectList(\_serviceClient.GetAccounts(), "IdAccount", "Name", transaction.FKAccount);

ViewBag.FkCategory = new SelectList(\_serviceClient.GetCategories(), "IdCategory", "Name", transaction.FkCategory);

return View(transaction);

}

// GET: Transactions/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Transaction transaction = \_serviceClient.GetTransactionById(id.Value);//db.Transactions.Find(id);

if (transaction == null)

{

return HttpNotFound();

}

ViewBag.FKAccount = new SelectList(\_serviceClient.GetAccounts(), "IdAccount", "Name", transaction.FKAccount);

ViewBag.FkCategory = new SelectList(\_serviceClient.GetCategories(), "IdCategory", "Name", transaction.FkCategory);

return View(transaction);

}

// POST: Transactions/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "IdTransaction,FKAccount,FkCategory,Name,Date,Notes")] Transaction transaction)

{

if (ModelState.IsValid)

{

//db.Entry(transaction).State = EntityState.Modified;

//db.SaveChanges();

\_serviceClient.UpdateTransaction(transaction);

return RedirectToAction("Index");

}

ViewBag.FKAccount = new SelectList(\_serviceClient.GetAccounts(), "IdAccount", "Name", transaction.FKAccount);

ViewBag.FkCategory = new SelectList(\_serviceClient.GetCategories(), "IdCategory", "Name", transaction.FkCategory);

return View(transaction);

}

// GET: Transactions/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Transaction transaction = \_serviceClient.GetTransactionById(id.Value);//db.Transactions.Find(id);

if (transaction == null)

{

return HttpNotFound();

}

return View(transaction);

}

// POST: Transactions/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

//Transaction transaction = db.Transactions.Find(id);

//db.Transactions.Remove(transaction);

//db.SaveChanges();

\_serviceClient.DeleteTransactionById(id);

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

//db.Dispose();

}

base.Dispose(disposing);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using ExpenseTracker.Model;

namespace ExpenseTracker.Server.MVC.Controllers

{

public class CategoriesController : Controller

{

private ExpensesService.ExpenseTrackerServiceClient \_serviceClient

= new ExpensesService.ExpenseTrackerServiceClient();

// GET: Categories

public ActionResult Index()

{

var categories = \_serviceClient.GetCategories();

return View(categories.ToList());

}

// GET: Categories/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = \_serviceClient.GetCategoryById(id.Value);//db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

// GET: Categories/Create

public ActionResult Create()

{

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name");

return View();

}

// POST: Categories/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "IdCategory,FKType,Name")] Category category)

{

if (ModelState.IsValid)

{

\_serviceClient.AddCategory(category);

return RedirectToAction("Index");

}

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name", category.FKType);

return View(category);

}

// GET: Categories/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = \_serviceClient.GetCategoryById(id.Value);//db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name", category.FKType);

return View(category);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "IdCategory,FKType,Name")] Category category)

{

if (ModelState.IsValid)

{

\_serviceClient.UpdateCategory(category);

return RedirectToAction("Index");

}

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name", category.FKType);

return View(category);

}

// GET: Categories/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = \_serviceClient.GetCategoryById(id.Value);//db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

// POST: Categories/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

\_serviceClient.DeleteCategoryById(id);

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

\_serviceClient.Close();

}

base.Dispose(disposing);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using ExpenseTracker.Model;

namespace ExpenseTracker.Server.MVC.Controllers

{

public class CategoryTypesController : Controller

{

private ExpensesService.ExpenseTrackerServiceClient \_serviceClient

= new ExpensesService.ExpenseTrackerServiceClient();

// GET: CategoryTypes

public ActionResult Index()

{

return View(\_serviceClient.GetCategoryTypes());

}

// GET: CategoryTypes/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

CategoryType categoryType = \_serviceClient.GetCategoryTypeById(id.Value);//db.Types.Find(id);

if (categoryType == null)

{

return HttpNotFound();

}

return View(categoryType);

}

// GET: CategoryTypes/Create

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "IdType,Name")] CategoryType categoryType)

{

if (ModelState.IsValid)

{

\_serviceClient.AddCategoryType(categoryType);

return RedirectToAction("Index");

}

return View(categoryType);

}

// GET: CategoryTypes/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

CategoryType categoryType = \_serviceClient.GetCategoryTypeById(id.Value);//db.Types.Find(id);

if (categoryType == null)

{

return HttpNotFound();

}

return View(categoryType);

}

// POST: CategoryTypes/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "IdType,Name")] CategoryType categoryType)

{

if (ModelState.IsValid)

{

\_serviceClient.UpdateCategoryType(categoryType);

return RedirectToAction("Index");

}

return View(categoryType);

}

// GET: CategoryTypes/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

CategoryType categoryType = \_serviceClient.GetCategoryTypeById(id.Value);//db.Types.Find(id);

if (categoryType == null)

{

return HttpNotFound();

}

return View(categoryType);

}

// POST: CategoryTypes/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

\_serviceClient.DeleteCategoryTypeById(id);

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

//db.Dispose();

\_serviceClient.Close();

}

base.Dispose(disposing);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using ExpenseTracker.Model;

namespace ExpenseTracker.Server.MVC.Controllers

{

public class CategoriesController : Controller

{

private ExpensesService.ExpenseTrackerServiceClient \_serviceClient

= new ExpensesService.ExpenseTrackerServiceClient();

// GET: Categories

public ActionResult Index()

{

var categories = \_serviceClient.GetCategories();

return View(categories.ToList());

}

// GET: Categories/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = \_serviceClient.GetCategoryById(id.Value);//db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

// GET: Categories/Create

public ActionResult Create()

{

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name");

return View();

}

// POST: Categories/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "IdCategory,FKType,Name")] Category category)

{

if (ModelState.IsValid)

{

\_serviceClient.AddCategory(category);

return RedirectToAction("Index");

}

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name", category.FKType);

return View(category);

}

// GET: Categories/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = \_serviceClient.GetCategoryById(id.Value);//db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name", category.FKType);

return View(category);

}

// POST: Categories/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "IdCategory,FKType,Name")] Category category)

{

if (ModelState.IsValid)

{

\_serviceClient.UpdateCategory(category);

return RedirectToAction("Index");

}

ViewBag.FKType = new SelectList(\_serviceClient.GetCategoryTypes(),//db.Types,

"IdType", "Name", category.FKType);

return View(category);

}

// GET: Categories/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Category category = \_serviceClient.GetCategoryById(id.Value);//db.Categories.Find(id);

if (category == null)

{

return HttpNotFound();

}

return View(category);

}

// POST: Categories/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

\_serviceClient.DeleteCategoryById(id);

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

//db.Dispose();

\_serviceClient.Close();

}

base.Dispose(disposing);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using ExpenseTracker.Model;

namespace ExpenseTracker.Server.MVC.Controllers

{

public class AccountsController : Controller

{

//private ExpenseTrackerContext db = new ExpenseTrackerContext();

private ExpensesService.ExpenseTrackerServiceClient \_serviceClient

= new ExpensesService.ExpenseTrackerServiceClient();

// GET: Accounts

public ActionResult Index()

{

return View(\_serviceClient.GetAccounts());

}

// GET: Accounts/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Account account = \_serviceClient.GetAccountById(id.Value);

if (account == null)

{

return HttpNotFound();

}

return View(account);

}

// GET: Accounts/Create

public ActionResult Create()

{

return View();

}

// POST: Accounts/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "IdAccount,Name")] Account account)

{

if (ModelState.IsValid)

{

\_serviceClient.AddAccount(account);

return RedirectToAction("Index");

}

return View(account);

}

// GET: Accounts/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Account account = \_serviceClient.GetAccountById(id.Value);

if (account == null)

{

return HttpNotFound();

}

return View(account);

}

// POST: Accounts/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "IdAccount,Name")] Account account)

{

if (ModelState.IsValid)

{

\_serviceClient.UpdateAccount(account);

return RedirectToAction("Index");

}

return View(account);

}

// GET: Accounts/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Account account = \_serviceClient.GetAccountById(id.Value);

if (account == null)

{

return HttpNotFound();

}

return View(account);

}

// POST: Accounts/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

\_serviceClient.DeleteAccountById(id);

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

\_serviceClient.Close();

}

base.Dispose(disposing);

}

}

}

**Разметка, на примере работы с таблицей категорий:**

Просмотр списка:

@model IEnumerable<ExpenseTracker.Model.Category>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Type.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Type.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.IdCategory }) |

@Html.ActionLink("Details", "Details", new { id=item.IdCategory }) |

@Html.ActionLink("Delete", "Delete", new { id=item.IdCategory })

</td>

</tr>

}

</table>

Редактирование:

@model ExpenseTracker.Model.Category

@{

ViewBag.Title = "Edit";

}

<h2>Edit</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Category</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.IdCategory)

<div class="form-group">

@Html.LabelFor(model => model.FKType, "FKType", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("FKType", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.FKType, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Создание новой записи:

@model ExpenseTracker.Model.Category

@{

ViewBag.Title = "Create";

}

<h2>Create</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Category</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.FKType, "FKType", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("FKType", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.FKType, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Удаление:

@model ExpenseTracker.Model.Category

@{

ViewBag.Title = "Delete";

}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Category</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Type.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Type.Name)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Name)

</dd>

</dl>

@using (Html.BeginForm()) {

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<input type="submit" value="Delete" class="btn btn-default" /> |

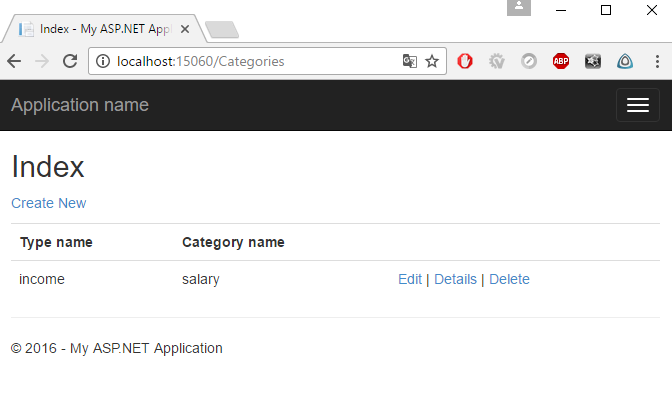
@Html.ActionLink("Back to List", "Index")

</div>

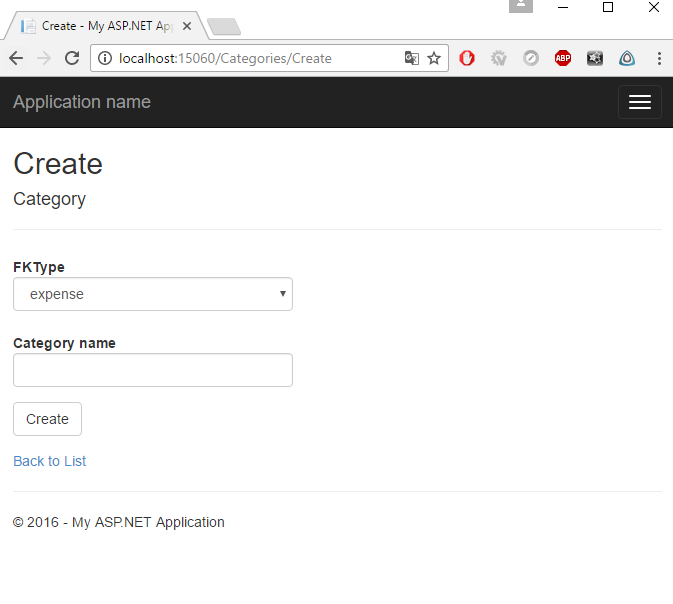
}

</div>

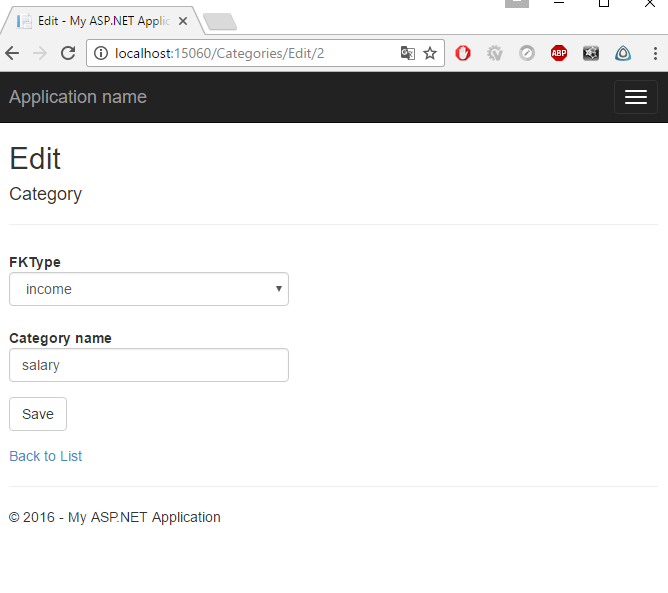
Внешний вид представлений (на примере списка категорий) показан на рисунках 4-7.



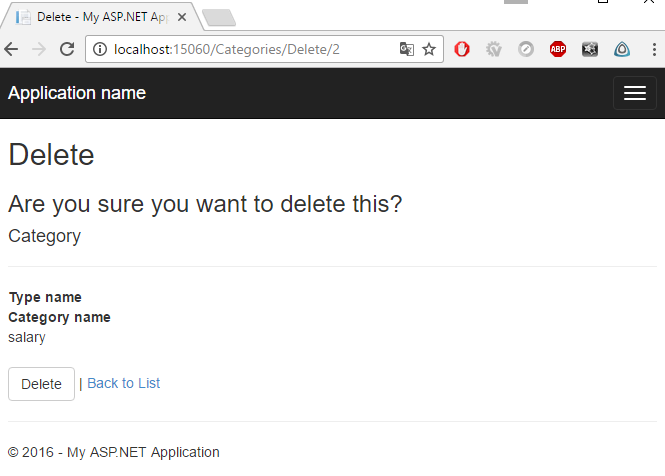
*Рисунок 4 – список категорий*



*Рисунок 5 – создание новой категории*



*Рисунок 6 – редактирование категории*



*Рисунок 7 – подтверждение удаления категории*

# Заключение

В ходе работы над рассчетным заданием получен работоспособный программный продукт с использованием сервисной архитектуры WCF, ORM, а также ASP.NET MVC. Отточены навыки создания программ с использованием трехслойной архитектуры.

В качестве путей развития программного продукта следует указать такую функциональность, как авторизация пользователей, сбор статистики и представление ее в виде графиков и диаграмм, создание отчетов.

# Список использованных источников

1. Общие сведения о платформе Entity Framework – MSDN [Электронный ресурс] – режим доступа: <https://msdn.microsoft.com/ru-ru/library/bb399567(v=vs.110).aspx>
2. Общие сведения о ASP.NET MVC – MSDN [Электронный ресурс] – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/dd381412(v=vs.108).aspx>
3. Windows Communication Foundation – MSDN [Электронный ресурс] – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/dd456779(v=vs.110).aspx>