Котов Виталий Евгеньевич

**Система учета студентов из числа иностранных граждан**

Дипломная *работа* по специальности

1-31-03-4 «Информатика»

Научный руководитель

Доцент кафедры информатика

Платонов Александр Сергеевич

Дипломная работа допущена к защите

Заведующий кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Н. Батан

\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2015

# **РЕФЕРАТ**

Котов Виталий Евгеньевич. Тема дипломной работы: система учета студентов из числа иностранных граждан.

Java, REST-сервисы, MySql, AngularJS, Intellij Idea, TomCat, Maven, AngularJS.

Разработанное приложение позволяет вести учет студентов из числа иностранных граждан и составление статистики.

# **ОГЛАВЛЕНИЕ**

[РЕФЕРАТ 2](#_Toc417412280)

[ОГЛАВЛЕНИЕ 3](#_Toc417412281)

[ВВЕДЕНИЕ 4](#_Toc417412282)

[ОБЗОР ЛИТЕРАТУРНЫХ ИССТОЧНИКОВ ПО ТЕМЕ 6](#_Toc417412283)

[ГЛАВА 1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ 7](#_Toc417412284)

[1.1 Сведения о MySQL 7](#_Toc417412285)

[1.2 Сведения о REST 7](#_Toc417412286)

[1.3 Сведения о Java 9](#_Toc417412287)

[1.4 Сведения о AngularJS 10](#_Toc417412288)

[ГЛАВА 2 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 11](#_Toc417412289)

[2.1 Анализ предметной области и объекта исследования 11](#_Toc417412290)

[2.2 Постановка задачи 11](#_Toc417412291)

[2.3 Входные и выходные документы 12](#_Toc417412292)

[2.4 Анализ бизнес-процессов 12](#_Toc417412293)

[ГЛАВА 3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 14](#_Toc417412294)

[3.1 Построение инфологической модели 14](#_Toc417412295)

[3.2 Требования к разрабатываемой 15](#_Toc417412296)

[3.3 Логическая модель данных 15](#_Toc417412297)

[3.4 Физическое проектирование 17](#_Toc417412298)

[3.5 Модель интерфейса 18](#_Toc417412299)

[ГЛАВА 4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 20](#_Toc417412300)

[4.1 Обоснование выбора ПО для разработки 20](#_Toc417412301)

[4.2 Структура ИС и функционирование каждого блока 20](#_Toc417412302)

[4.3 Структура базы данных 21](#_Toc417412303)

[4.4 Результаты работы ИС 23](#_Toc417412304)

[ЗАКЛЮЧЕНИЕ 26](#_Toc417412305)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc417412306)

[ПРИЛОЖЕНИЯ 28](#_Toc417412307)

# ВВЕДЕНИЕ

Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Эти базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД).

Одним из ключевых направлений в области автоматизация бизнес-процессов с использованием информационных технологий является разработка баз данных, позволяющих решить проблему хранения и систематизации информации согласно индивидуальным требованиям компании.

Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем привели к широкому распространению наиболее удобных и сравнительно простых для понимания реляционных (табличных) СУБД. Для обеспечения одновременного доступа к данным множества пользователей, нередко расположенных достаточно далеко друг от друга и от места хранения баз данных, созданы сетевые мультипользовательские версии БД основанных на реляционной структуре. В них тем или иным путем решаются специфические проблемы параллельных процессов, целостности (правильности) и безопасности данных, а также санкционирования доступа.

Для проектируемой системы предметной областью является отдел университета, который занимается учетом студентов из числа иностранных граждан. Необходимо спроектировать базу данных, информация которой будет использоваться для учета студентов и составления статистики.

**Предметом исследования** является организация процесса учета студентов из числа иностранных граждан, а также расчет статистики.

**Объектом исследования** является реальный экономический объект: отдел, который отвечает за учет студентов.

**Цель исследования:** разработать автоматизированную систему для оптимизации работы отдела. Разрабатываемый прототип будет предоставлять возможность вести автоматизированный учет студентов и расчет статистических данных.

# ОБЗОР ЛИТЕРАТУРНЫХ ИССТОЧНИКОВ ПО ТЕМЕ

1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ

## Сведения о MySQL

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией.

MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

MySQL портирована на большое количество платформ: Linux, Mac OS, Solaris, SunOS, Windows. MySQL имеет API для языков Delphi, C, C++, Java, PHP, Python, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC. MyODBC работает на всех системах Microsoft Windows и на большинстве платформ Unix.

В реализуемом приложении используется MySQL Server версии 5.6.

## Сведения о REST

REST (сокр. от англ. Representational State Transfer — «передача состояния представления») — метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP и CORBA.

В широком смысле REST означает концепцию построения распределённого приложения, при которой компоненты взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

Хотя данная концепция лежит в самой основе Всемирной паутины, термин REST был введён Роем Филдингом, одним из создателей протокола HTTP, лишь в 2000 году. В своей диссертации в Калифорнийском университете в Ирвайне он подвёл теоретическую основу под метод взаимодействия клиентов и серверов во Всемирной паутине, абстрагировав его и назвав «передачей репрезентативного состояния». Филдинг описал концепцию построения распределённого приложения, при которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе сервера (желаемом репрезентативном состоянии), и сервер не обязан сохранять информацию о состоянии клиента («клиентской сессии»).

В качестве необходимых условий для построения распределенных REST-приложений Филдинг перечислил следующие:

* Клиент-серверная архитектура.
* Сервер не обязан сохранять информацию о состоянии клиента.
* В каждом запросе клиента должно явно содержаться указание о возможности кэширования ответа и получения ответа из существующего кэша.
* Клиент может взаимодействовать не напрямую с сервером, а с произвольным количеством промежуточных узлов. При этом клиент может не знать о существовании промежуточных узлов, за исключением случаев передачи конфиденциальной информации.
* Унифицированный программный интерфейс сервера. Филдинг приводил URI в качестве примера формата запросов к серверу, а в качестве примера ответа сервера форматы HTML, XML и JSON, различаемые с использованием идентификаторов MIME.

Филдинг указывал, что приложения, не соответствующие приведённым условиям, не могут называться REST-приложениями. Если же все условия соблюдены, то, по его мнению, приложение получит следующие преимущества:

* надёжность (за счет отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
* производительность (за счет использования кэша);
* масштабируемость;
* прозрачность системы взаимодействия, особенно необходимая для приложений обслуживания сети;
* простота интерфейсов;
* портативность компонентов;
* легкость внесения изменений;
* способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

## Сведения о Java

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Технология Java протестирована, усовершенствована, расширена и проверена участниками сообщества разработчиков Java, архитекторов и энтузиастов. Java позволяет разрабатывать высокопроизводительные портативные приложения практически на всех компьютерных платформах.

Java стала незаменимым инструментом для разработчиков и открыла для них следующие возможности:

* написание программного обеспечения на одной платформе и его запуск практически на любой другой платформе
* создание программ, работающих в веб-браузере и имеющих доступ к веб-службам;
* разработка приложений на стороне сервера для форумов в Интернете, магазинов, опросов, обработки форм HTML и много другого;
* объединение приложений или служб с использованием языка Java для создания высокоспециализированных приложений или служб
* создание многофункциональных и эффективных приложений для мобильных телефонов, удаленных процессоров, микроконтроллеров, беспроводных модулей, датчиков, шлюзов, потребительских продуктов и практически любых других категорий электронных устройств;

В реализуемом приложении используется Java 8, релиз версии состоялся 19 марта 2014 года.

## Сведения о AngularJS

AngularJS — JavaScript-фреймворк с открытым исходным кодом. Предназначен для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки.

Фреймворк работает с HTML, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные JavaScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных.

Angular придерживается MVC-шаблона проектирования и поощряет слабую связь между представлением, данными и логикой компонентов. Используя внедрение зависимости, Angular переносит на клиентскую сторону такие классические серверные службы, как видозависимые контроллеры. Следовательно, уменьшается нагрузка на сервер и веб-приложение становится легче.

1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

## Анализ предметной области и объекта исследования

Для проектируемой базы данных предметной областью является процесс учета студентов. В учебном отделе собрана информация о студентах, которые обучаются на различных факультетах, а также в определенных группах. При этом, необходим учет их регистрации, места проживания, формы обучения, а также личной информации. Упрощение процесса учета студентов, подсчет их общего количества, а также составление статистики представляет интерес для данного исследования.

База данных студентов создается для экономии времени которое тратится на составление документации (отчетов) об учащихся студентах. При работе с такой базой, добавлять, изменять информацию в отчете гораздо проще и быстрее.

## Постановка задачи

Ставиться задача разработать программный комплекса, который позволит упростить работу отдела университета, отвечающего за учет студентов. Описываемый прототип позволит хранить информацию о факультетах, специальностях, группах, личную информацию, а также сведения регистрации и места проживания. Также функционал прототипа будет включать возможность различных видов подсчетов (общего количества студентов по специальностям, по странам, по общежитиям и т.д.).

## Входные и выходные документы

После проведения необходимых исследований и анализа предметной области, было выявлено, что необходимыми для разработки базы данных являются следующие входные документы:

1. Информация о студентах
2. Информация о факультетах
3. Информация о специальностях
4. Информация о группах
5. Информация о курсах
6. Информация о странах
7. Информация о общежитиях

Выходными документами будут списки студентов по специальностям, по странам, по общежитиям, списки с информацией о регистрации. В которых будет представлена информация о количестве студентов по курсам, а также их общее количество.

## Анализ бизнес-процессов

В отделе университета ведется учет студентов из числа иностранных граждан. Каждый студент имеет личную информацию и информацию об образовании. В университете если несколько факультетов на которых обучают ряду специальностей. Студенты сформированы в группы, каждая группа учится на определенной специальности.

Для составления статистики, отчетов, требуется хранить всю перечисленную выше информацию.

Трикотажная фабрика производит продукцию различного вида (пальто, костюмы и т.д.). Каждый продукт имеет несколько моделей (костюм в полоску, костюм клетчатый и т.д.). При составлении отчетности о поступлениях на склад фиксируются сведения именно о модели товара, которая прибыла на склад, а не о виде продукции. Учитывая это, в отчетную ведомость входят сведения о модели, а не о самом продукте.

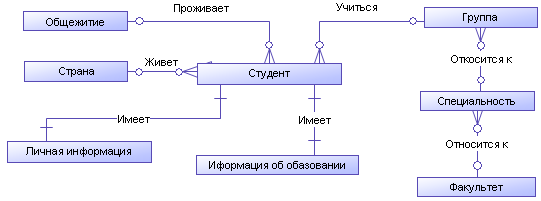
Для составления отчетной ведомости потребуется информация о поступлениях моделей произведенных товаров на склад. Данные о поступлениях на склад должны содержать сведения о поступившей модели, дату поступления и количество поступивших моделей.

Также необходимо хранить информацию непосредственно о самих моделях. А именно название модели, цену модели, а также вид товара к которому относится модель.

1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## Построение инфологической модели

Информация, предоставляемая в базе данных, в первую очередь должна отображать реальные объекты прикладной области и связи между ними.

Результатом инфологического проектирования является концептуальная модель, которая представляет структуру данных не зависимую от любой физической реализации. Инфологическую модель исследуемой области можно представить следующим образом:

* Студент – сущность, в которой храниться информация о студенте.
* Группа – сущность, содержащая информацию о группе в которой учатся студенты.
* Специальность – сущность, которая содержит данные о специальности которой обучаются группы студентов.
* Факультет – сущность.

Связи между сущностями следующие:

* В группе учатся студенты, причем студент связан только с одной группой, а группа хотя бы с одним студентом. Получаем, что сущности связаны между собой связью один ко многим (1:М);
* Специальность связана с несколькими группами, а группа с одной специальностью. Получаем, что сущности связаны между собой связью один ко многим (1:М). Аналогичным образам связаны сущности факультет и специальность, общежитие и студент, а также страна и студент;
* Студент имеет личную информацию и информацию об образовании. Эти сущности связаны между собой связью один к одному(1:1);

Выявим все атрибуты, описывающие сущности ER-модели.

……………

…………

Сущность Товар (Product) имеет атрибуты Код товара (IdProduct) и Наименование товара (NameProduct).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Код товара | + | - | числовой |
| Наименование товара | - | - | строковый |

Сущность Модель (Model) имеет атрибуты Код модели (IdModel), Наименование модели (NameModel), Код товара (IdProduct), Цена модели (PriceModel).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Код модели | + | - | числовой |
| Наименование модели | - | - | строковый |
| Код товара | - | + | числовой |
| Цена модели | - | - | денежный |

Сущность Поступление на склад (Receipt) имеет атрибуты Код поступления (IdReceipt), Код модели (IdModel), Дата поступления (DateReceipt), Количество (Amount, количество моделей товара поступивших на склад), Кто принял товар (Accepted, кто принял поступление товара на склад).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Код поступления | + | - | числовой |
| Код модели | - | + | числовой |
| Дата поступления | - | - | дата |
| Количество | - | - | числовой |
| Кто принял товар | - | - | строковый |

Все атрибуты во всех сущностях не имеют значений по умолчанию и не могут принимать значение Null.

## Требования к разрабатываемой

Учитывая особенности предметной области, разрабатываемый прототип должен поддерживать следующие возможности:

* Добавление, изменение и удаление студентов
* Перевод студента на следующий курс
* Отчисление студента

При этом функционал прототипа должен включать возможность просмотра списка всех студентов. А также в просмотр отчетов по курсам, специальностям, по странам и т. д.

## Логическая модель данных

Была выбрана, как чаще всего и поступают, реляционная модель данных в связи с наглядностью табличного представления данных и удобства работы с ними. Для каждой сущности ER-модели создадим таблицу. Имя сущности – имя таблицы.

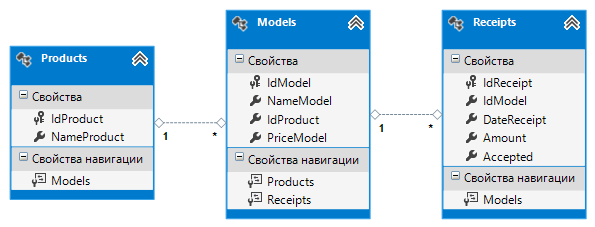
|  |
| --- |
| **Товары** |
| Код товара |
| Наименование товара |

|  |
| --- |
| **Модели** |
| Код модели |
| Наименование модели |
| Код товара |
| Цена модели |

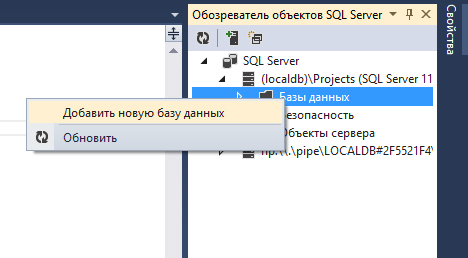
|  |
| --- |
| **Поступления** |
| Код поступления |
| Код модели |
| Дата поступления |
| Количество |
| Кто принял товар |

А так же определим связи между таблицами посредством механизма первичных и внешних ключей.

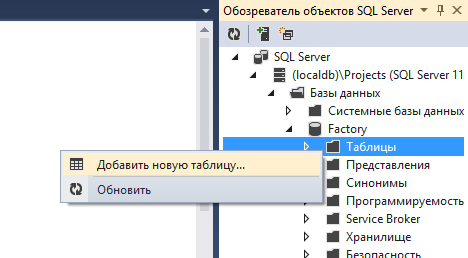
Таблица Товары связана с таблицей Модели с помощью внешнего ключа Код товара из таблицы Модели. Таблица Модели связана с таблицей Поступления посредством внешнего ключа Код модели из таблицы Поступления. На рисунке 2 изображена схема связей между таблицами.



## Физическое проектирование

Для начала создадим новую базу данных на сервере баз данных. Для этого в Visual Studio откройте Обозреватель объектов SQL Server в меню ВИД и кликните по кнопке Добавить новую базу данных (рисунок 3).

В появившемся окне вводим название базы данных. Назовем ее interstudents (Иностранные студенты).

В базе данных нам нужно три таблицы. Для начала добавим таблицу Products. Для этого в окне Обозреватель объектов SQL Server разверните базу данных, которую только что создали, и кликните правой кнопкой мыши по пункту Таблицы. Выберите в появившемся меню Добавить новую таблицу, как показано на рисунке 4.

Появиться конструктор для создания новой таблицы. Будем использовать окно T-SQL, потому что это более краткий и точный способ описать необходимую спецификацию таблицы. Введите SQL-оператор, показанный в листинге 1, и нажмите кнопку Обновить в верхнем левом углу окна конструктора.

CREATE TABLE [dbo].[Products] (

[IdProduct] INT IDENTITY (1, 1) NOT NULL,

[NameProduct] NCHAR (25) NOT NULL,

PRIMARY KEY CLUSTERED ([IdProduct] ASC)

);

Повторим действия для добавления таблиц Models и Receipts. SQL-операторы для этих таблиц приведены в листинге 2.

CREATE TABLE [dbo].[Models] (

[IdModel] INT IDENTITY (1, 1) NOT NULL,

[NameModel] NCHAR (25) NOT NULL,

[IdProduct] INT NOT NULL,

[PriceModel] MONEY NOT NULL,

PRIMARY KEY CLUSTERED ([IdModel] ASC),

CONSTRAINT [FK\_Models\_Products] FOREIGN KEY ([IdProduct]) REFERENCES [dbo].[Products] ([IdProduct])

);

CREATE TABLE [dbo].[Receipts] (

[IdReceipt] INT IDENTITY (1, 1) NOT NULL,

[IdModel] INT NOT NULL,

[DateReceipt] DATE NOT NULL,

[Amount] INT NOT NULL,

[Accepted] NCHAR (25) NOT NULL,

CONSTRAINT [PK\_Receipts] PRIMARY KEY CLUSTERED ([IdReceipt] ASC),

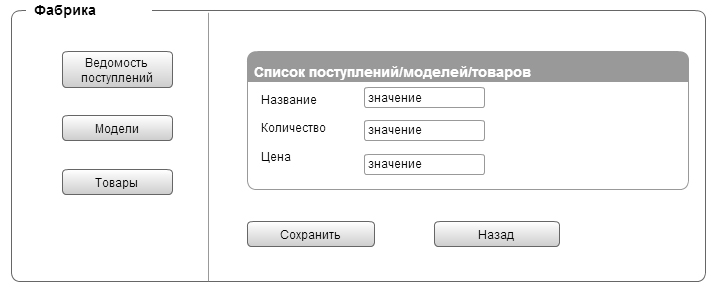
CONSTRAINT [FK\_Receipt\_Models] FOREIGN KEY ([IdModel]) REFERENCES [dbo].[Models] ([IdModel])

);

## Модель интерфейса

Учитывая описанные выше требования к программному комплексу и особенности информационной системы в целом, была разработана модель интерфейса. На рисунке 5 изображен вид при просмотре списков товаров, моделей, а также ведомости поступлений.

При редактировании ведомости поступлений, моделей, товаров интерфейс имеет вид изображенный на рисунке 5.

1. 
2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## Обоснование выбора ПО для разработки

Для создания программного комплекса была выбрана интегрированная среда разработки программного обеспечения Intellij Idea 14, которая содержит все необходимое для начала работы, в том числе инструменты для тестирования, компилятор и отладчик (дебаггер).

Также была использована библиотека Hibernate для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных. Это библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения. Она представляет собой свободное программное обеспечение с открытым исходным кодом

Для визуального проектирования баз данных был выбран MySQL Workbench — инструмент, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое окружение для системы баз данных MySQL.

Для запуска веб-приложения используется Tomcat — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server.

## Структура ИС и функционирование каждого блока

Созданная информационная система является трёхуровневой. Трёхуровневая архитектура программного комплекса предполагает наличие в нём трёх компонентов: клиентского приложения (обычно называемого «тонким клиентом» или интерфейсом), сервера приложения (логики приложения), к которому подключено клиентское приложение, и сервера базы данных (доступ к базе данных), с которым работает сервер приложений.

Сервер базы данных представлен MySQL-сервером; сервер приложения реализован на языке Java и web-сервером TomCat; интерфейс представлен в виде web-приложения.

Созданная информационная система является трёхуровневой. Трёхуровневая архитектура программного комплекса предполагает наличие в нём трёх компонентов: клиентского приложения (обычно называемого «[тонким клиентом](http://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BD%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82)» или интерфейсом), сервера приложения (логики приложения), к которому подключено клиентское приложение, и сервера базы данных (доступ к базе данных), с которым работает сервер приложений.

Сервер базы данных представлен MsSQL-сервером; доступ к данным технологией ADO.NET Entity Framework; сервер приложения технологией ASP.NET и web-сервером IIS; интерфейс представлен в виде web-приложения.

В проект решения входит три части:

* библиотека классов Factory.Domain
* библиотека классов Factory.BusinessLogic
* проект ASP.NET MVC Factory.WebUI

Библиотека классов Factory.Domain включает в себя определение и реализацию интерфейсов доступа к источнику данных. Также и реализацию источника данных (Entity Domain Models).

В состав библиотеки классов Factory.BusinessLogic входят классы, описывающие логику.

Проект ASP.NET MVC Factory.WebUI представляет интерфейс приложения.

## Структура базы данных

С уже имеющейся базы данных Entity Framework может автоматически создать модель данных, состоящую из классов и свойств, соответствующих объектам базы данных (таким, как таблицы и столбцы). Информация о структуре базы, модель данных содержится в XML в файле с расширением .edmx.

Сущность Products (продукты) представлена в листинге 3.

public partial class Products

{

public Products()

{

this.Models = new HashSet<Models>();

}

public int IdProduct { get; set; }

public string NameProduct { get; set; }

public virtual ICollection<Models> Models { get; set; }

}

Сущность Models (модели) и Receipts (поступления на склад) описаны в листингах 4 и 5 соответственно.

public partial class Models

{

public Models()

{

this.Receipts = new HashSet<Receipts>();

}

public int IdModel { get; set; }

public string NameModel { get; set; }

public int IdProduct { get; set; }

public decimal PriceModel { get; set; }

public virtual Products Products { get; set; }

public virtual ICollection<Receipts> Receipts { get; set; }

}

public partial class Receipts

{

public int IdReceipt { get; set; }

public int IdModel { get; set; }

public System.DateTime DateReceipt { get; set; }

public int Amount { get; set; }

public string Accepted { get; set; }

public virtual Models Models { get; set; }

}

Экземпляр контекста DbContext представляет сочетание шаблонов единицы работы и репозитория, которое может быть использовано для запроса от базы данных и группирования изменений, которые можно затем записать обратно в хранилище одним блоком.

Листинг 6 содержит определение DbContext.

public partial class FactoryEntities : DbContext

{

public FactoryEntities()

: base("name=FactoryEntities")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

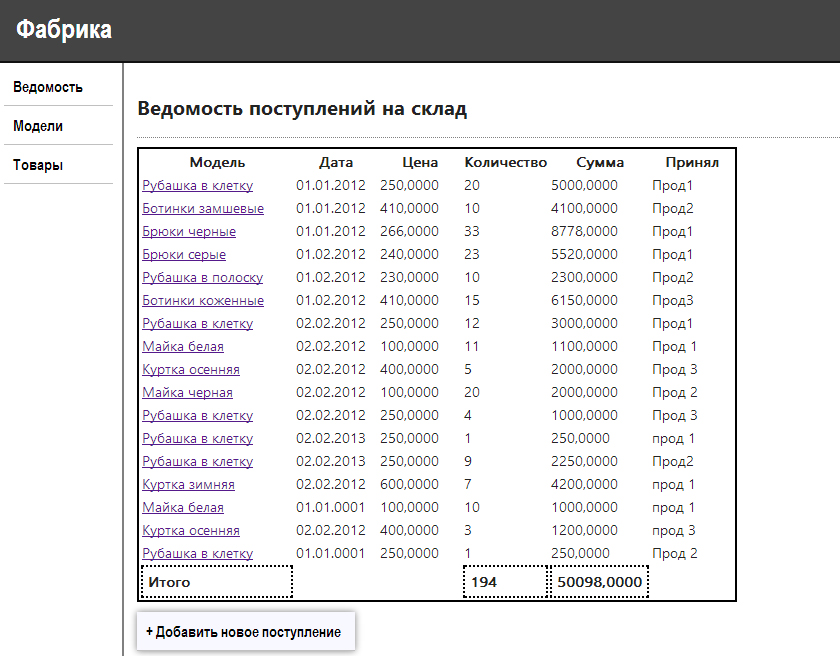
public DbSet<Models> Models { get; set; }

public DbSet<Products> Products { get; set; }

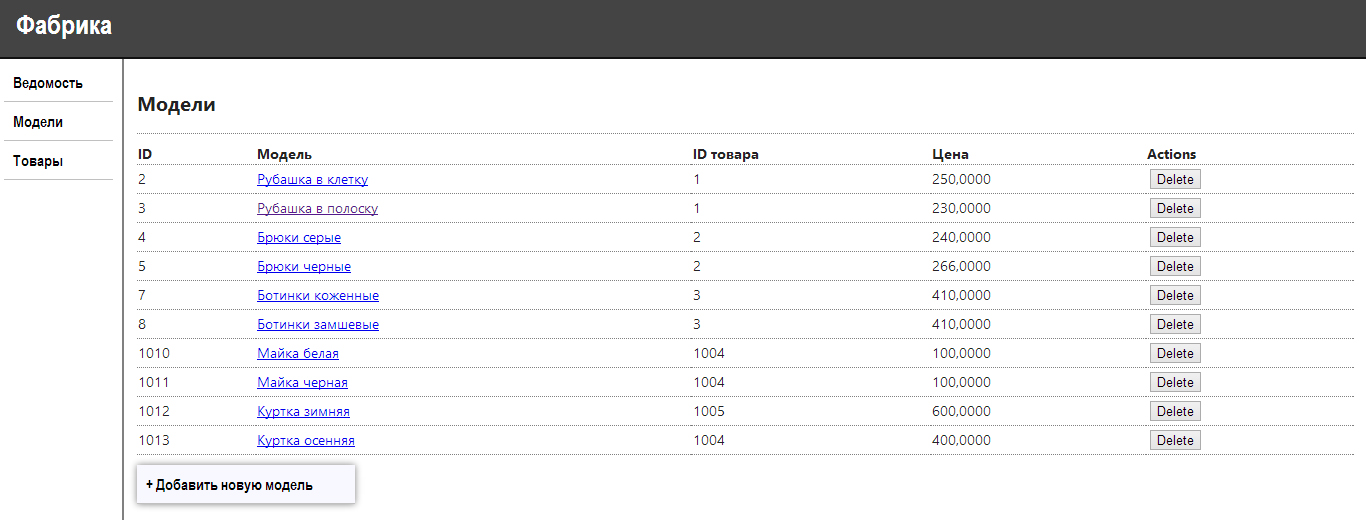
public DbSet<Receipts> Receipts { get; set; }

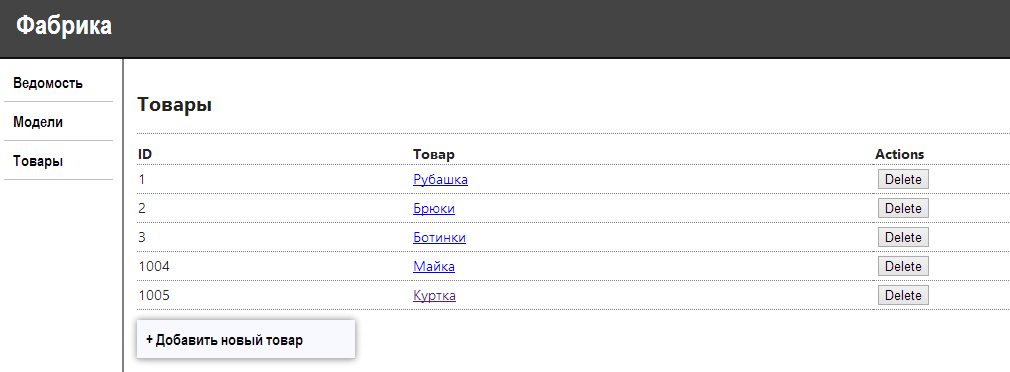
}

## Результаты работы ИС

При запуске приложения появляется ведомость поступлений на склад. В ней присутствует информация о моделях товара, а также их количество и стоимость. Также подсчитывается общее количество все товаров на складе и их стоимость.

Здесь можно редактировать поступления на склад, а также добавить новое поступление.

Для просмотра списка моделей, которые производит фабрика, нужно выбрать в боковом меню вкладку Модели.

Список всех производимых товаров можно просмотреть при переходе на вкладку Товары в боковом меню.

# ЗАКЛЮЧЕНИЕ

В данной дипломной работе была рассмотрена разработка и реализация автоматизированной системы учета студентов. В процессе разработки освоена технология доступа к данным Hibernate и MySQL Server, язык программирования Java и интегрированная среда разработки Intellij Idea.

В ходе дипломной работы были изучены принципы проектирования приложений с Rest архитектурой, а также разработка и реализация систем управления базой данных.

Был разработан компактный пользовательский интерфейс и использования JavaScript фреймворка AngularJS, который предоставляет возможность добавления, изменения и удаления данных. Функционирование приложения можно расширить возможностью выгрузки данных в Excel.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рихтер Дж., CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#, 2012. – 928 с.
2. Троелсен, Э., Язык программирования C# 5.0 и платформа .NET 4.5 6-е издание, 2012. – 1311 с.
3. Шилдт, Г., C# 4.0 Полное руководство, 2011. – 1056 с.
4. Microsoft Developer Network [Электронный ресурс] Режим доступа: http://msdn.microsoft.com/ru-ru/library/system.threading.aspx

Дата доступа: 10.05.2014.

# ПРИЛОЖЕНИЯ