**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.

Базы данных стали основой информационных систем и в корне изменили методы работы многих организаций. В частности, в последние годы развитие технологии баз данных привело к созданию весьма мощных и удобных в эксплуатации систем. Благодаря этому системы баз данных стали доступными широкому кругу пользователей.

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, что компании инвестируют значительные средства в разработку компьютеризированных системы для эффективного решения этих задач. Один из способов повышения эффективности обработки данных — организовать их эффективное хранение и получение. Самый распространенный подход к хранению данных на сегодня — использовать *реляционную базу данных.*

В любом случае БД — это просто средство хранения данных.

Сервер приложений — это мост между программами – клиентами.

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.

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

**Актуальность**

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 разработки проектирования данного приложения является создание приложения, которое может хранить данные и анализировать их, помогая администрации предсказывать какое количество товаров необходимо завезти на склад.

1. **Описание используемых технологий**

**1.1 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.

Общую схему взаимодействия этих компонентов можно представить следующим образом:



В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга.

Благодаря этому реализуется концепция разделение ответственности, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если нам, допустим, важная визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на бэкэнде и тестировать контроллер.

Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

Свою реализацию паттерна представляет платформа ASP.NET MVC. 2013 год ознаменовался выходом новой версии ASP.NET MVC - MVC 5, а также релизом Visual Studio 2013, которая предоставляет инструментарий для работы с MVC5.

Для создания адаптивного и расширяемого интерфейса в MVC 5 используется css-фреймворк Bootstrap

* 1. **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. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с бд - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с бд - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

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.

Способы взаимодействия с БД:

* Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных
* Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.
* Code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.
  1. **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.

1. **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 2016. 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 16

[0] None

Compiler: 2

Please verify your choices:

Compiler: Microsoft Visual C++ 2008 SP1

Location: C:\Program Files\Microsoft Visual Studio 16

Are these correct [y]/n? y

После подобного сообщения — линкер настроен.

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.

После этого мы уже можем получить динамическую библиотеку. Набираем в командном окне MATLAB'a deploytool. Создаем новый .NET Assembly проект MATLABplane, указываем размещение. Далее создаем класс planeClass, добавляем в него plane.m 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.