Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Отчет по преддипломной практике

Выполнил:

студент группы 253503

Карабинский А.О.

Руководитель от вуза:

Теслюк В.Н.

Минск 2017

Содержание

[Сведения о программной платформе и проекте](#__RefHeading___Toc756_450552699) 3

[NET Framework](#__RefHeading___Toc758_450552699) 3

[C#](#__RefHeading___Toc760_450552699) 5

[ASP.NET](#__RefHeading___Toc762_450552699) 6

[JavaScript](#__RefHeading___Toc766_450552699) 8

Entity Framework………………………………………………………………..9

SQL……………………………………………………………………………..11

[Заключение](#__RefHeading___Toc778_450552699) 12

[Список использованных материалов](#__RefHeading___Toc780_450552699) 13

# 

# Сведения о программной платформе и проекте

## Программная платформа

**.NET Framework**

В середине 2000 года компания Microsoft объявила о работе над новой платформой для создания программ, которая получила имя платформа .NET (.NET Framework). Платформа .NET образует каркас, включающий среду исполнения, библиотеку классов и набор технологий для построения приложений и служб. Основным инструментом разработки для платформы .NET является интегрированная среда Microsoft Visual Studio.

Основой платформы .NET является общеязыковая среда исполнения (Common Language Runtime, CLR). CLR работает как «прослойка» между операционной системой и программой для платформы .NET. Каждая программа для .NET состоит из одной или нескольких сборок.

Сборка (assembly) является результатом компиляции исходных текстов на некотором языке программирования для платформы .NET и содержит метаданные и код на Common Intermediate Language.

Метаданные – это информационные таблицы с полным описанием всех типов, размещённых в сборке.

Common Intermediate Language (CIL или IL) – внутренний язык платформы .NET, он не зависит от типа процессора. В процессе работы программы CIL компилируется в машинный код специальным JIT-компилятором (Just-in-Time compiler).

Основная задача CLR – это манипулирование сборками: загрузка, JIT-компиляция, создание окружения для выполнения сборок. Важной функцией CLR является управление памятью при работе приложения и выполнение автоматической «сборки мусора», то есть фонового освобождения неиспользуемой памяти. Кроме этого, CLR реализует в приложениях для .NET проверку типов, управление политиками безопасности при доступе к коду и другие функции.

В состав платформы .NET входит обширная библиотека классов Framework Class Library (FCL). Частью этой библиотеки является базовый набор классов Base Class Library (BCL), в который входят классы для работы со строками и коллекциями данных, для поддержки многопоточности и множество других классов. Частью FCL являются компоненты, поддерживающие различные технологии обработки данных и организации взаимодействия с пользователем. Это классы для работы с XML и базами данных, для создания пользовательских интерфейсов.

В стандартную поставку платформы .NET включено несколько компиляторов. Это компиляторы языков C#, F#, Visual Basic .NET, C++/CLI. Благодаря открытым спецификациям компиляторы для .NET предлагаются различными сторонними производителями. Необходимо подчеркнуть, что любой язык для платформы .NET является верхним элементом архитектуры. Имена элементов библиотеки FCL не зависят от языка программирования. Специфичной частью языка остаётся только синтаксис. Этот факт упрощает межъязыковое взаимодействие, перевод текста программы с одного языка на другой.

Для поддержки межъязыкового взаимодействия служат две спецификации платформы .NET. Общая система типов (Common Type System, CTS) описывает набор типов, который должен поддерживаться любым языком программирования для .NET. Общеязыковая спецификация (Common Language Specification, CLS) – это общие правила поведения для всех .NET-языков.

**C#**

Язык C# и связанную с ним среду .NET Framework можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработчиков. Среда .NET является такой средой, которая была создана для того, чтобы в ней можно было разрабатывать практически любое приложение для запуска в Windows, а C# является языком программирования, который был специально создан для использования в .NET Framework. Например, с применением C# и .NET Framework можно создавать динамические веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возможностями для работы в оперативном и автономном режимах.

Не стоит поддаваться заблуждению из-за наличия в названии Framework слова "NET" и думать, что данная среда предназначена только для создания приложений, ориентированных на Интернет. Слово "NET" здесь является лишь показателем того, что, по мнению Microsoft, распределенные приложения, в которых обработка распределяется между клиентом и сервером, являются шагом вперед. Однако важно понимать, что C# представляет собой язык, предназначенный не только для написания приложений, способных работать в Интернете и в сети. Он предоставляет средства для кодирования практически любого типа программного обеспечения или компонентов для платформы Windows. Язык C# и среда .NET привели к революционным изменениям в способе написания разработчиками программ и сделали программирование приложений для Windows гораздо более простым, чем когда-либо.

C# — это относительно новый язык программирования, который характеризуется двумя следующими преимуществами:

* C# спроектирован и разработан специально для применения с Microsoft .NET Framework (развитой платформой разработки, развертывания и выполнения распределенных приложений).
* C# — язык, основанный на современной объектно-ориентированной методологии проектирования, при разработке которого специалисты из Microsoft опирались на опыт создания подобных языков, построенных в соответствии с предложенными около 20 лет назад объектно-ориентированными принципами.

**ASP.NET**

Рассмотрим схему работы в сети Интернет, которую можно назвать классической, так как эта схема является исторически первой. Основными элементами классической схемы являются *браузер* и *веб-сервер*. При взаимодействии браузера и веб-сервера выполняются следующие шаги:

1. Браузер формирует *запрос* к серверу, используя *протокол HTTP*. Как правило, браузер запрашивает *HTML-страницу*, то есть текстовый файл, содержащий HTML-код.
2. Сервер анализирует запрос браузера и извлекает из локального хранилища требуемый файл.
3. Сервер формирует *HTTP-ответ*, включающий требуемую информацию, и отсылает его браузеру по протоколу HTTP.
4. Браузер выполняет отображение страницы.

Классическая схема проста, но обладает существенным недостатком – страницы статичны, и их содержимое не может меняться на сервере в зависимости от запросов клиента. В настоящее время подобный подход не соответствует большинству информационных услуг, предоставляемых с помощью сети Интернет. Все большее распространение получают технологии, при использовании которых страницы (целиком или частично) генерируются на сервере *непосредственно* перед отправкой клиенту. Работают технологии «серверных страниц» по схожим принципам:

* Для представления информации на сайте используются не страницы с HTML-кодом, а серверные страницы специального синтаксиса (который часто является HTML-подобным).
* При запросе серверной страницы веб-сервер запускает отдельный служебный процесс, которому перенаправляется запрос.
* В служебном процессе страница анализируется, по ней генерируется некий объект, соответствующий странице.
* Служебный процесс выполняет методы сгенерированного объекта. Как правило, объект имеет специальный метод, генерирующий выходной поток страницы в виде HTML-кода.
* Выходной HTML-поток перенаправляется веб-серверу, который формирует HTTP-ответ и отсылает его браузеру.
* Браузер выполняет отображение страницы.

Далее выделим особенности, присущие технологии ASP.NET.

* *Работа с управляемым кодом.* Служебный процесс ASP.NET основан на управляемом коде. Запросы к каждому веб-приложению обрабатываются в отдельном домене служебного процесса. Серверной странице ставиться в соответствие пользовательский класс, объект которого непосредственно генерирует страницу. Также доступны стандартные для .NET библиотеки классов и возможности межъязыкового взаимодействия.
* *Разделение кода и представления*. Данная концепция также называется *Code Behind*. Согласно ей желательно, чтобы страница ASP.NET состояла из двух частей: файла с описанием вида страницы (разметка, элементы управления) и файла с кодом методов страницы. Эти два файла могут компилироваться в отдельные классы или представлять собой частичный класс. При изменении любого из файлов на сервере происходит перекомпиляция страницы.
* *Серверные элементы управления.* Для конструирования страницы ASP.NET содержит несколько десятков специальных серверных элементов управления. Каждый такой элемент в конечном итоге транслируется в один или несколько обычных элементов HTML. Серверные элементы управления поддерживают событийную модель, содержат большое количество настраиваемых свойств. Они предоставляют более высокий уровень абстракции в сравнение с классическими элементами управления HTML. Кроме этого, имеется возможность создавать собственные серверные элементы управления.
* *Событийная модель.* Технология ASP.NET пытается перенести на веб-программирование принципы, используемые при написании приложений Windows Forms. Для того чтобы закодировать логику страницы, программист пишет обработчики соответствующих событий. Когда событие происходит, информация о нём пересылается от клиента на сервер, где срабатывает обработчик события. Затем страница вновь пересылается клиенту. Подчеркнем следующие важные детали. Во-первых, основой реализации событийной модели является схема, при которой страница отсылает запросы сама к себе. Во-вторых, чтобы страница сохраняла свое состояние между отдельными циклами приёма-передачи, это состояние фиксируется в специальном скрытом поле страницы. Этот технологический прием называется в ASP.NET *поддержкой состояния представления страницы* (коротко – *поддержка View State*). И, наконец, ASP.NET пытается перенести событийную модель на возможно большее количество классических элементов управления HTML. Для реализации этого используются «вкрапления» в страницу клиентских скриптов.
* *Поддержка пользовательских сессий и кэширование.* В ASP.NET существует богатый набор встроенных возможностей для работы с данными пользовательских сессий, выполнения кэширования данных, идентификации пользователей.

JavaScript

**JavaScript** (часто сокращают до **JS**) — это лёгкий, интерпретируемый, объектно-ориентированный язык с функциями первого класса, самый известный скриптовый язык для веб-страниц, но также исползуется во многих не браузерных окружениях. Прототипно-ориентированный, мультипарадигменный язык сценариев, который поддерживает динамический, объектно-ориентированный, императивный и функциональный стили программирования.

JavaScript запускается на стороне клиента Интернета, который может использоваться для создания/программирования того, как веб-страницы будут вести себя при наступлении каких-либо событий. JavaScript легко изучить, а также это мощный скриптовый язык, широко используемый для контролирования поведения веб-страниц.

Вопреки распространенному мнению, **JavaScript** не **является "интерпретируемым Java"**. В двух словах, JavaScript — это динамический скриптовый язык, поддерживающий прототипное создание объектов. Базовый синтаксис намеренно похож на Java и C++, чтобы уменьшить число новых концепций, необходимых для изучения языка. Такие языковые конструкции, как if, for, while, switch, try ... catch похожи на конструкции этих языков.

JavaScript может функционировать и как процедурный, и как объектно-ориентированный язык. Объекты можно создавать программно во время выполнения, путем присоединения методов и свойств или пустых объектов **во время выполнения**, в отличие от синтаксических определений классов в компилируемых языках, таких как С++ или Java. После того, как объект был создан, он может быть использован в качестве плана (или прототипа) для создания похожих объектов.

Динамические возможности JavaScript включают: создание объектов во время выполнения, переменное число параметров, динамическое создание скриптов (с помощью eval), перебор объектов (с помощью for ... in), восстановление исходного кода (программы на JavaScript могут декомпилировать тела функций обратно в исходный код).

**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.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* 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

## SQL

SQL (обычно произносимый как "СИКВЭЛ" или "ЭСКЮЭЛЬ") символизирует собой *Структурированный Язык Запросов*. Это - язык, который дает Вам возможность создавать и работать в реляционных базах данных, являющихся наборами связанной информации, сохраняемой в таблицах.

Информационное пространство становится более унифицированным. Это привело к необходимости создания стандартного языка, который мог бы использоваться в большом количестве различных видов компьютерных сред. Стандартный язык позволит пользователям, знающим один набор команд, использовать их для создания, нахождения, изменения и передачи информации - независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ.

В нашем все более и более взаимосвязанном компьютерном мире, пользователь снабженый таким языком, имеет огромное преимущество в использовании и обобщении информации из ряда источников с помощью большого количества способов.

Элегантность и независимость от специфики компьютерных технологий, а также его поддержка лидерами промышленности в области технологии реляционных баз данных, сделало SQL (и, вероятно, в течение обозримого будущего оставит его) основным стандартным языком. По этой причине, любой, кто хочет работать с базами данных 90-х годов, должен знать SQL.

Стандарт SQL определяется ANSI (*Американским Национальным Институтом Стандартов*) и в данное время также принимается ISO (*Международной Организацией по Стандартизации*). Однако, большинство коммерческих программ баз данных расширяют SQL без уведомления ANSI, добавляя различные особенности в этот язык, которые, как они считают, будут весьма полезны. Иногда они несколько нарушают стандарт языка, хотя хорошие идеи имеют тенденцию развиваться и вскоре становиться стандартами "*рынка*" сами по себе в силу полезности своих качеств.

На данном уроке мы будем, в основном, следовать стандарту ANSI, но одновременно иногда будет показывать и некоторые наиболее общие отклонения от его стандарта.

Точное описание особенностей языка приводится в документации на СУБД, которую Вы используете. SQL системы InterBase 4.0 соответствует стандарту ANSI-92 и частично стандарту ANSI-III.

# Заключение

В ходе преддипломной практики была изучена структура предприятия, используемые технологии при разработке ПО. Был получен опыт в сфере промышленной разработки программного обеспечения.

В ходе работы над индивидуальным заданием был повышен уровень навыков по вышеизложенным технологиям разработки программного обеспечения. Также хочу отметить руководителей практики, благодаря которым я приобрел новые знания.

# Список использованных материалов

1. Шилдт Г. C# 4.0: полное руководство, М.: ООО «И.Д. Вильямс», 2011

2. Язык программирования С# и платформа .NET 4.0, 5-е изд. : М. : ООО "И.Д. Вильямc", 2011.

3. MSDN [Электронный ресурс]. – Сеть разработчиков Microsoft. – Режим доступа. : http://msdn.ru .

4. MDN [Электронный ресурс]. – Сеть разработчиков Mozilla. – Режим доступа. : https://developer.mozilla.org .