МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И РОБОТОТЕХНИКИ

Кафедра «Система автоматизированного проектирования»

**КУРСОВОЙ ПРОЕКТ**

по курсу “Базы знаний и поддержка принятия решений”

Тема:  **«**Web-приложение администрирования компьютерной сети общежития»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент гр.107518 |
|  |  | Добродей С.Г. |
|  |  |  |
|  |  |  |
| Руководитель: |  | к.т.н., доцент  Кочуров В.А. |
|  |  |  |

Минск 2012

Содержание

[**Введение** 4](#_Toc324357707)

[**1.** **Постановка задачи** 6](#_Toc324357708)

[**2.** **Трехуровневая архитектура** 7](#_Toc324357709)

[**3. Выбор средств и технологий для реализации поставленной задачи** 11](#_Toc324357710)

[3.1 Выбор средств СУБД 11](#_Toc324357711)

[3.2 Выбор языка программирования 14](#_Toc324357712)

[3.3 Выбор технологии для сетевого программирования 16](#_Toc324357713)

[**4. Архитектура базы данных** 19](#_Toc324357714)

[**5.** **Программная реализация** 21](#_Toc324357715)

[5.1 Создание серверного приложения 21](#_Toc324357716)

[5.2 Создание клиентского приложения 24](#_Toc324357717)

[5.3 Взаимодействие клиентского и серверного приложения 26](#_Toc324357718)

[**Заключение** 32](#_Toc324357719)

[**Литература** 33](#_Toc324357720)

[**Приложение** 34](#_Toc324357721)

**Введение**

*Система управления базами данных (СУБД)* – совокупность программных и лингвистических средств общего и специального назначения, обеспечивающих управление созданием и использованием баз данных.

Современные реляционные системы управления базами данных содержат*:*

* набор инструментов для создания таблиц и отношений между связанными таблицами;
* средства администрирования базы данных;
* развитый пользовательский интерфейс, который позволяет получить доступ к информации, хранящейся в базе данных;
* средства разработки приложений, использующих базы данных.

С помощью СУБД *можно:*

* выбрать информацию, представляющую интерес;
* вывести на печать всю таблицу или только выбранные записи и поля в различных форматах;
* отображать информацию базы данных в графическом виде;
* осуществлять необходимые вычисления при формировании отчетов и выборке данных из таблиц.

*Реляционная база данных* – это совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Реляционные базы данных позволяют хранить информацию в нескольких «плоских» (двухмерных) таблицах, связанных между собой посредством совместно используемых полей данных, называемых ключами. Эти таблицы получили название реляций, поэтому модель стала называться реляционной. Записи в таблицах не повторяются. Их уникальность обеспечивается первичным ключом, содержащим набор полей, однозначно определяющих запись. Для быстрого поиска информации в базе данных создаются индексы по одному или нескольким полям таблицы. Значения индексов хранятся в упорядоченном виде и содержат ссылки на записи таблицы. Для автоматической поддержки целостности связанных данных, находящихся в разных таблицах, используются первичные и внешние ключи. Для выборки данных из нескольких таблиц используются значения одного или нескольких совпадающих полей.

Современные системы управления базами данных в основном являются приложениями Windows, так как данная среда позволяет более полно использовать возможности персональной ЭВМ. Среди наиболее ярких представителей СУБД можно отметить: Lotus Approach, Microsoft Access, Borland dBase, Borland Paradox, Microsoft Visual FoxPro, Microsoft Visual Basic, а также баз данных Microsoft SQL Server и Oracle, используемые в приложениях, построенных по технологии «клиент-сервер».

*Технология “Клиент-сервер”* – технология, разделяющая приложение СУБД на две части: клиентскую (интерактивный графический интерфейс, расположенный на компьютере пользователя) и сервер, собственно осуществляющий управление данными, разделение информации, администрирование и безопасность, находящийся на выделенном компьютере. Взаимодействие “клиент-сервер” осуществляется следующим образом: клиентская часть приложения формирует запрос к серверу баз данных, на котором выполняются все команды, а результат исполнения запроса отправляется клиенту для просмотра и использования. Данная технология применяется, когда размеры баз данных велики, когда велики размеры вычислительной сети, и производительность при обработке данных, хранящихся не на компьютере пользователя. Если технология “клиент-сервер” не применяется, то для обработки даже нескольких записей весь файл копируется на компьютер пользователя, а только затем обрабатывается. При этом резко возрастает загрузка сети, и снижается производительность труда многих сотрудников. На сегодняшний день разработчик не связан рамками какого-либо конкретного пакета и в зависимости от поставленной задачи может использовать самые разные приложения.

# **Постановка задачи**

Настройка и обслуживания компьютерной сети общежития – это трудоёмкий процесс. На благо системных администраторов, задачу учета пользователей ЛКС помогают решать современные информационные технологии. Информация находящаяся и распоряжении сисадминов и необходимая им в процессе регистрации клиентов ЛКС состоит из базы соотвествия IP адресов и MAC адресов устройств.

Разрабатываемое приложение должно быть реализовано в виде Web страницы.

# **Трехуровневая архитектура**

В процессе научных исследований, посвященных тому, как именно должна быть устроена СУБД, предлагались различные способы реализации. Самым жизнеспособным из них оказалась предложенная американским комитетом по стандартизации ANSI (American National Institute) трехуровневая система организации БД, изображенная на рис.  2.1:

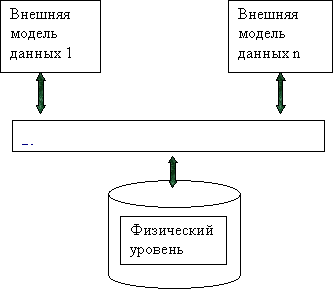


Рис. 2.1 - трехуровневая система организации БД

**Обзор архитектуры**

В литературе встречается несколько вариантов представления трехуровневой архитектуры. Рассмотрим два из них.

Согласно первому архитектура включает три уровня: внутренний, концептуальный и внешний. В общих чертах они представляют собой следующее:

*Внутренний*- это уровень, наиболее близкий к физическому хранению, т.е.  связанный со способами сохранения информации на физических устройствах хранения.

*Внешний*- наиболее близок к пользователям, т.е. он связан со способами представления данных для отдельных пользователей.

*Концептуальный уровень* - это промежуточный уровень между двумя первыми; другими словами, это центральное управляющее звено, где БД представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной БД. Фактически концептуальный уровень отражает обобщенную модель предметной области (объектов реального мира), для которой создавалась БД. Как любая модель, концептуальная модель отражает только существенные, с точки зрения обработки, особенности объектов реального мира.

Внешний уровень - это индивидуальный уровень пользователя. У каждого пользователя есть свой язык общения. Все эти языки включают подъязык данных, т.е. подмножество операторов всего языка, связанное только с объектами и операциями БД.

В принципе любой подъязык данных является на самом деле комбинацией по крайней мере двух  подчиненных языков определения данных (DDL), который поддерживает определения или объявления объектов БД, и языка обработки данных (DML), который поддерживает операции с такими объектами или их обработку.

Внешнее представление - это содержимое БД, каким видит его определенный пользователь (т.е. для этого пользователя внешнее представление и есть БД).

Концептуальное представление - это представление всей информации БД в несколько более абстрактной форме по сравнению с физическим способом хранения данных. Однако концептуальное представление существенно отличается от способа представления данных какому- либо отдельному пользователю. Концептуальное представление - это представление данных такими, какие они есть на самом деле, а не такими, какими вынужден их видеть пользователь. Концептуальная схема - это определение такого представления. В большинстве существующих систем концептуальная схема в действительности представляет собой немного больше, чем простое объединение всех отдельных внешних схем с дополнительными средствами безопасности и правилами обеспечения целостности.

Внутреннее представление - это представление нижнего уровня всей БД. Внутреннее представление так же, как внешнее и концептуальное, не связанно с физическим уровнем. Это представление предполагает бесконечное линейное адресное пространство.

Внутреннее представление описывается с помощью внутренней схемы, которая определяет не только различные типы хранимых записей, но также существующие индексы, способы представления хранимых полей, физическую последовательность хранимых записей и т.д.

Эта архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между 2 и 3) независимость при работе с данными. Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих на этой же БД. Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной БД. Это именно то, чего не хватало при использовании файловых систем.

Выделение концептуального уровня позволило разработать аппарат централизованного управления БД.

Согласно второй теории в компьютерных технологиях трёхуровневая архитектура, синоним трёхзвенная архитектура (англ. three-tier или Multitier architecture) предполагает наличие следующих компонентов приложения: клиентское приложение, подключенное к серверу приложений, который в свою очередь подключен к серверу базы данных.

1. Клиент (терминал) — это интерфейсный компонент, который представляет первый уровень, собственно приложение для конечного пользователя. Первый уровень не должен иметь прямых связей с базой данных, быть нагруженным основной бизнес-логикой и хранить состояние приложения. На первый уровень может быть вынесена и обычно выносятся несложные операции (сортировка, группировка, подсчет значений) с данными, уже загруженными в клиентское приложение.
2. Сервер приложений располагается на втором уровне. На втором уровне сосредоточена бо́льшая часть бизнес-логики. Вне его остаются фрагменты, экспортируемые на терминалы, а также погруженные в третий уровень хранимые процедуры и триггеры.
3. Сервер базы данных обеспечивает хранение данных и выносится на третий уровень. Обычно это стандартная реляционная или объектно-ориентированная СУБД. Если третий уровень представляет собой базу данных вместе с хранимыми процедурами, триггерами и схемой, описывающей приложение в терминах реляционной модели, то второй уровень строится как программный интерфейс, связывающий клиентские компоненты с прикладной логикой базы данных.

В простейшей конфигурации физически сервер приложений может быть совмещён с сервером базы данных на одном компьютере, к которому по сети подключается один или несколько клиентов.

В «правильной» (с точки зрения безопасности, надёжности, масштабирования) конфигурации сервер базы данных находится на выделенном компьютере, к которому по сети подключены один или несколько серверов приложений, к которым, в свою очередь, по сети подключаются клиенты.

**Достоинства**

По сравнению с клиент-серверной или файл-серверной архитектурой можно выделить следующие достоинства трёхуровневой архитектуры:

* масштабируемость
* конфигурируемость — изолированность уровней друг от друга позволяет (при правильном развертывании архитектуры) быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней
* высокая безопасность
* высокая надёжность
* низкие требования к скорости канала (сети) между терминалами и сервером приложений
* низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости. Терминалом может выступать не только компьютер, но и, например, мобильный телефон.

**Недостатки**

Недостатки вытекают из достоинств. По сравнению c клиент-серверной или файл-серверной архитектурой можно выделить следующие недостатки трёхуровневой архитектуры:

* более высокая сложность создания приложений;
* сложнее в разворачивании и администрировании;
* высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;
* высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

# **3.** **Выбор средств и технологий для реализации поставленной задачи**

## Выбор средств СУБД

MySQL представляет собой систему управления базами данных с поддержкой языка SQL. Эта СУБД предоставляется вместе с относящимся к ней открытым исходным кодом и является бесплатной для использования во многих областях. На ранних этапах развития ее программного обеспечения СУБД MySQL время от времени подвергалась критике из-за отсутсвия поддержки некоторых основных конструкций SQL, таких как подзапросы и внешние ключи. Но СУБД MySQL с самого начала обладала такими положительными качествами, как необременительные условия лицензирования, высокая производительности удобство пользования, поэтому в конечном итоге нашла широкий круг горячо поддерживающих ее пользователей. Распространению этой СУБД отчасти способствовала возможность ее широкого применения в сочетании со многими языками программирования и технологиями, такими как PHP, Java, Perl, Python и т.д. Пользователей особенно привлекало то, что для работы с этой СУБД предоставлялись стабильные, хорошо документированные модули и расширения. Разработчики СУБД MySQL, в свою очередь, вознаградили приверженность пользователей, введя поддержку подзапросов и внешних ключей, начиная с ряда версий 4.1.

Поэтому в проекте будет использоваться самая популярная на данный момент система управления базами данных (СУБД) – Microsoft® SQL Server. SQL Server обладает развитыми возможностями по взаимодействию с другими системами, а также по автоматизации регулярных административных задач. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.

*Платформа SQL Server обеспечивает следующие преимущества:*

* повышение эффективности используемых данных: помимо предоставления защищенной и надежной системы управления базами данных для приложений корпоративного класса, SQL Server предоставляет интегрированные службы отчетов, анализа данных и data mining;
* повышение продуктивности сотрудников: за счет развитых возможнстей бизнес-анализа и интеграции со знакомыми инструментами, SQL Server позволяет оперативно обеспечивать сотрудников оперативной информацией, в представлении оптимизированном под них потребности;
* снижение затрат на информационные технологии: SQL Server позволяет упростить разработку, развертывание и сопровождение бизнес-приложений, предоставляя гибкую среду разработки для разработчиков приложений и интегрированного набора инструментов управления для администраторов баз данных.

*Основные характеристики:*

* бесплатная СУБД, что отличает ее от других дорогостоящих продуктов;
* многопоточность: поддержка нескольких одновременных запросов;
* оптимизация связей с присоединением многих данных за один проход;
* записи фиксированной и переменной длины;
* ODBC драйвер в комплекте с исходным кодом;
* гибкая система привилегий и паролей;
* до 16 ключей в таблице, каждый ключ может иметь до 15 полей;
* поддержка ключевых полей и специальных полей в операторе CREATE;
* поддержка чисел длинной от 1 до 4 байт (ints, float, double, fixed), строк переменной длины и меток времени;
* интерфейс с языками C и Perl;
* основанная на потоках, быстрая система памяти;
* утилита проверки и ремонта таблицы (isamchk);
* все данные хранятся в формате ISO8859\_1;
* все операции работы со строками не обращают внимания на регистр символов в обрабатываемых строках;
* псевдонимы применимы как к таблицам, так и к отдельным колонкам в таблице;
* все поля имеют значение по умолчанию. INSERT можно использовать на любом подмножестве полей;
* легкость управления таблицей, включая добавление и удаление ключей и полей.

***Вывод:*** Рассмотрев основные характеристики различных операционных систем и СУБД в дальнейшем в качестве серверной ОС выбираем - Windows, в качестве СУБД - MS SQL.

## Выбор языка программирования

В настоящее время одним из самых распространенных языков программирования является C#.

Язык С# — это очередная ступень бесконечной эволюции языков программирования. Его создание вызвано процессом усовершенствования и адаптации, который определял разработку компьютерных языков в течение последних лет. Подобно всем успешным языкам, которые увидели свет раньше, С# опирается на прошлые достижения постоянно развивающегося искусства программирования.

В языке С# (созданном компанией Microsoft для поддержки среды .NET Framework) проверенные временем средства усовершенствованы с помощью самых современных технологий. С# предоставляет очень удобный и эффективный способ написания программ для современной среды вычислительной обработки данных, которая включает операционную систему Windows, Internet, компоненты и пр. В процессе

становления язык С# переопределил весь "ландшафт" программирования.

Он разработан в 2000 году Андерсом Хейлсбергом, Скоттом Вилтамутом и Питером Гольде под эгидой Microsoft Research. Основным постулатом С# является высказывание: "всякая сущность есть объект". Язык основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

C# был создан специально для технологии ASP.NET.  В то же время на C# полностью написана и сама ASP.NET.

C# — это полнофункциональный объектно-ориентированный язык, который поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Он имеет прекрасную поддержку компонентов, надежен и устойчив благодаря использованию «сборки мусора», обработки исключений, безопасности типов.

Языку С# "досталось" богатое наследство. Он — прямой потомок двух самых успешных языков программирования (С и C++) и тесно связан с не менее успешным языком Java. Понимание природы этих взаимосвязей крайне важно для понимания С#. Поэтому знакомство с С# мы начнем с рассмотрения исторической среды этих трех языков.

Язык C# разрабатывался "с нуля" и вобрал в себя много полезных свойств таких языков, как C++, Java, Visual Basic, а также Pascal, Delphy и др. При этом необходимость обратной совместимости с предыдущими версиями отсутствовала, что позволило языку C# избежать многих отрицательных сторон своих предшественников.

Как и Java, C# разрабатывался для Интернет и примерно 75% его синтаксических возможностей аналогичны языку программирования Java, его также называют «очищенной» версией Java. 10% подобны языку программирования C++, а 5% – заимствованы из языка программирования Visual Basic. Объем новых концептуальных идей в языке C# около 10%.

Выделение и объединение лучших идей современных языков программирования делает язык C# не просто суммой их достоинств, а языком программирования нового поколения.

**Вывод:**

В данном курсовом проекте в качестве языка написания приложения будет использоваться C#.

## Выбор технологии для сетевого программирования

1. **Принципы ASP.NET**

Хотя ASP.NET берёт своё название от старой технологии Microsoft ASP, она значительно от неё отличается. Microsoft полностью перестроила ASP.NET, основываясь на [Common Language Runtime](http://ru.wikipedia.org/wiki/Common_Language_Runtime" \o "Common Language Runtime) (CLR), которая является основой всех приложений Microsoft .NET. Разработчики могут писать код для ASP.NET, используя практически любые языки программирования, входящие в комплект [.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework) ([C#](http://ru.wikipedia.org/wiki/C_Sharp" \o "C Sharp), [Visual Basic.NET](http://ru.wikipedia.org/wiki/VB.NET) и [JScript .NET](http://ru.wikipedia.org/wiki/JScript_.NET" \o "JScript .NET)). ASP.NET имеет преимущество в скорости по сравнению со скриптовыми технологиями, так как при первом обращении код компилируется и помещается в специальный [кэш](http://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88" \o "Кэш), и впоследствии только исполняется, не требуя затрат времени на [парсинг](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D1%81%D0%B8%D0%BD%D0%B3" \o "Парсинг), [оптимизацию](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80%D0%B0), и т. д.

1. **Преимущества ASP.NET перед ASP**

* Компилируемый код выполняется быстрее, большинство ошибок отлавливается ещё на стадии разработки
* Значительно улучшенная обработка ошибок времени выполнения, с использованием блоков try..catch
* Пользовательские элементы управления (controls) позволяют выделять часто используемые шаблоны, такие как меню сайта
* Использование метафор, уже применяющихся в Windows-приложениях, например, таких как элементы управления и события
* Расширяемый набор элементов управления и библиотек классов позволяет быстрее разрабатывать приложения
* ASP.NET опирается на многоязыковые возможности .NET, что позволяет писать код страниц на VB.NET, Delphi.NET, Visual C#, J# и т. д.
* Возможность кэширования всей страницы или её части для увеличения производительности
* Возможность кэширования данных, используемых на странице
* Возможность разделения визуальной части и бизнес-логики по разным файлам («code behind»)
* Расширяемая модель обработки запросов
* Расширенная событийная модель
* Расширяемая модель серверных элементов управления
* Наличие master-страниц для задания шаблонов оформления страниц
* Поддержка [CRUD](http://ru.wikipedia.org/wiki/CRUD)-операций при работе с таблицами через GridView
* Встроенная поддержка [AJAX](http://ru.wikipedia.org/wiki/AJAX)
* ASP.NET имеет преимущество в скорости по сравнению с другими технологиями, основанными на скриптах.

Здесь можно привести определённые сравнения. Так, ASP - производная от Win32, XML и HTML; PHP - от XML, HTML, Java и CDI, тогда ASP.NET - от HTML и .NET(XML и XAML соответственно). При этом, если обычно Rich Media Application создают при помощи Flash, теперь это делается с помощью модуля Silverlight, так же через сам ASP.NET. ASP.NET - богатейшая[[](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8" \o "Википедия:Ссылки на источники)*[источник не указан 455 дней](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8" \o "Википедия:Ссылки на источники)*[]](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8" \o "Википедия:Ссылки на источники) среда для разработки и развёртывания веб-ресурсов. В ASP.NET можно работать с любым .NET языком, вплоть до Managed C++ и Visual Basic.

# **4. Архитектура базы данных**

База данных для данного курсового проекта будет написана на MS SQL. В качестве инструмента создания базы используем Microsoft SQL Server Managment Studio. Назовём создаваемую базу данных «System». Предполагаемая база данных должна состоять трех таблиц:

Таблица mac адресов

Таблица ip aдресов.

Таблица персональной информации.

# 

# **Программная реализация**

Смотрите в приложени 1.

# **Заключение**

В ходе выполнения данного курсового проекта было разработано web приложение администрирования компьютерной сети общежития,

Работа над данным курсовым проектом позволила получить большой объем полезных знаний как по базам данных так и программированию. Так, например, анализ СУБД позволил узнать направление мировых тенденций и сориентироваться в направлении дальнейшего обучения по данному предмету. Так же полезным оказалось практическое знакомство с технологиями сетевого программирования, позволяющим максимально просто и надежно создавать распределенные приложения.

# **Литература**

1. Э. Кровчик и др. «Сетевое программирование для профессионалов». Издательство «Лори», 2005.
2. PHP5 и MySQL. Библия пользователя. : Пер. с англ. – М. : Издательский дом «Вильяс», 2006 – 1216 с. : ил. – Парал. тит. англ.
3. [Network Programming](http://msdn.microsoft.com/ru-ru/library/4as0wz7t.aspx)
4. [Network Programming Samples](http://msdn.microsoft.com/ru-ru/library/ee890485.aspx)

# **Приложение**

Листинг программы

/\* DEFAULTS

----------------------------------------------------------\*/

body

{

background: #b6b7bc;

font-size: .80em;

font-family: "Helvetica Neue", "Lucida Grande", "Segoe UI", Arial, Helvetica, Verdana, sans-serif;

margin: 0px;

padding: 0px;

color: #696969;

}

a:link, a:visited

{

color: #034af3;

}

a:hover

{

color: #1d60ff;

text-decoration: none;

}

a:active

{

color: #034af3;

}

p

{

margin-bottom: 10px;

line-height: 1.6em;

}

/\* HEADINGS

----------------------------------------------------------\*/

h1, h2, h3, h4, h5, h6

{

font-size: 1.5em;

color: #666666;

font-variant: small-caps;

text-transform: none;

font-weight: 200;

margin-bottom: 0px;

}

h1

{

font-size: 1.6em;

padding-bottom: 0px;

margin-bottom: 0px;

}

h2

{

font-size: 1.5em;

font-weight: 600;

}

h3

{

font-size: 1.2em;

}

h4

{

font-size: 1.1em;

}

h5, h6

{

font-size: 1em;

}

/\* this rule styles <h1> and <h2> tags that are the

first child of the left and right table columns \*/

.rightColumn > h1, .rightColumn > h2, .leftColumn > h1, .leftColumn > h2

{

margin-top: 0px;

}

/\* PRIMARY LAYOUT ELEMENTS

----------------------------------------------------------\*/

.page

{

width: 960px;

background-color: #fff;

margin: 20px auto 0px auto;

border: 1px solid #496077;

}

.header

{

position: relative;

margin: 0px;

padding: 0px;

background: #4b6c9e;

width: 100%;

top: 0px;

left: 0px;

}

.header h1

{

font-weight: 700;

margin: 0px;

padding: 0px 0px 0px 20px;

color: #f9f9f9;

border: none;

line-height: 2em;

font-size: 2em;

}

.main

{

padding: 0px 12px;

margin: 12px 8px 8px 8px;

min-height: 420px;

}

.leftCol

{

padding: 6px 0px;

margin: 12px 8px 8px 8px;

width: 200px;

min-height: 200px;

}

.footer

{

color: #4e5766;

padding: 8px 0px 0px 0px;

margin: 0px auto;

text-align: center;

line-height: normal;

}

/\* TAB MENU

----------------------------------------------------------\*/

div.hideSkiplink

{

background-color:#3a4f63;

width:100%;

}

div.menu

{

padding: 4px 0px 4px 8px;

}

div.menu ul

{

list-style: none;

margin: 0px;

padding: 0px;

width: auto;

}

div.menu ul li a, div.menu ul li a:visited

{

background-color: #465c71;

border: 1px #4e667d solid;

color: #dde4ec;

display: block;

line-height: 1.35em;

padding: 4px 20px;

text-decoration: none;

white-space: nowrap;

}

div.menu ul li a:hover

{

background-color: #bfcbd6;

color: #465c71;

text-decoration: none;

}

div.menu ul li a:active

{

background-color: #465c71;

color: #cfdbe6;

text-decoration: none;

}

/\* FORM ELEMENTS

----------------------------------------------------------\*/

fieldset

{

margin: 1em 0px;

padding: 1em;

border: 1px solid #ccc;

}

fieldset p

{

margin: 2px 12px 10px 10px;

}

fieldset.login label, fieldset.register label, fieldset.changePassword label

{

display: block;

}

fieldset label.inline

{

display: inline;

}

legend

{

font-size: 1.1em;

font-weight: 600;

padding: 2px 4px 8px 4px;

}

input.textEntry

{

width: 320px;

border: 1px solid #ccc;

}

input.passwordEntry

{

width: 320px;

border: 1px solid #ccc;

}

div.accountInfo

{

width: 42%;

}

/\* MISC

----------------------------------------------------------\*/

.clear

{

clear: both;

}

.title

{

display: block;

float: left;

text-align: left;

width: auto;

}

.loginDisplay

{

font-size: 1.1em;

display: block;

text-align: right;

padding: 10px;

color: White;

}

.loginDisplay a:link

{

color: white;

}

.loginDisplay a:visited

{

color: white;

}

.loginDisplay a:hover

{

color: white;

}

.failureNotification

{

font-size: 1.2em;

color: Red;

}

.bold

{

font-weight: bold;

}

.submitButton

{

text-align: right;

padding-right: 10px;

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApplication2.Account

{

public partial class Register : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

RegisterUser.ContinueDestinationPageUrl = Request.QueryString["ReturnUrl"];

}

protected void RegisterUser\_CreatedUser(object sender, EventArgs e)

{

FormsAuthentication.SetAuthCookie(RegisterUser.UserName, false /\* createPersistentCookie \*/);

string continueUrl = RegisterUser.ContinueDestinationPageUrl;

if (String.IsNullOrEmpty(continueUrl))

{

continueUrl = "~/";

}

Response.Redirect(continueUrl);

}

}

}

<%@ Page Title="About Us" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"

CodeBehind="About.aspx.cs" Inherits="WebApplication2.About" %>

<%@ Register assembly="System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" namespace="System.Web.UI.DataVisualization.Charting" tagprefix="asp" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">

</asp:Content>

<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">

<h2>

&nbsp;IP&nbsp; MAC table of hostel network users&nbsp;</h2>

<p>

<asp:TextBox ID="TextBox1" runat="server" Width="343px"></asp:TextBox>

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="Search"

Width="89px" />

<asp:GridView ID="GridView1" runat="server" AllowPaging="True"

AutoGenerateColumns="False" BackColor="White" BorderColor="#CCCCCC"

BorderStyle="None" BorderWidth="1px" CellPadding="4" DataKeyNames="Id"

DataSourceID="SqlDataSource1" ForeColor="Black" GridLines="Horizontal"

PageSize="50" Width="537px" Height="1361px">

<Columns>

<asp:BoundField DataField="Id" HeaderText="Id" ReadOnly="True"

SortExpression="Id" />

<asp:BoundField DataField="ip" HeaderText="ip" SortExpression="ip" />

<asp:BoundField DataField="mac" HeaderText="mac" SortExpression="mac" />

</Columns>

<FooterStyle BackColor="#CCCC99" ForeColor="Black" />

<HeaderStyle BackColor="#333333" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="White" ForeColor="Black" HorizontalAlign="Right" />

<SelectedRowStyle BackColor="#CC3333" Font-Bold="True" ForeColor="White" />

<SortedAscendingCellStyle BackColor="#F7F7F7" />

<SortedAscendingHeaderStyle BackColor="#4B4B4B" />

<SortedDescendingCellStyle BackColor="#E5E5E5" />

<SortedDescendingHeaderStyle BackColor="#242121" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"

ConnectionString="<%$ ConnectionStrings:ConnectionString %>"

SelectCommand="SELECT iptable.Id, iptable.ip, mactable.mac FROM iptable INNER JOIN mactable ON iptable.Id = mactable.ip"></asp:SqlDataSource>

</p>

</asp:Content>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

namespace WebApplication2

{

public partial class About : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

bool hack = false;

string query = "SELECT ip FROM iptable WHERE Id = (SELECT ip FROM mactable WHERE mac='"+TextBox1.Text+"')";

if ((query.IndexOf("DELETE") != -1) || (query.IndexOf("INSERT") != -1) || (query.IndexOf("DRPOP") != -1) || (query.IndexOf("CREATE") != -1) || (query.IndexOf("UPDATE") != -1))

{

TextBox1.Text = "SQL Injection Protection";

hack = true;

}

if (!hack)

{

string connectionString = @"Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Database1.mdf;Integrated Security=True";

SqlConnection connection1 = new SqlConnection(connectionString);

connection1.Open();

SqlCommand command1 = new SqlCommand(query, connection1);

SqlDataReader dataReader1 = command1.ExecuteReader();

try

{

while (dataReader1.Read())

{

TextBox1.Text = dataReader1[0].ToString();

}

}

catch

{

TextBox1.Text = "Упс! Что-то пошло не так.";

}

finally

{

dataReader1.Close();

connection1.Close();

}

}

}

}

}