**Лабораторная работа № 5**

**Аутентификация с помощью форм. Работа с пространством имен System.Web.Security**

**Цель работы**

Освоить механизмы работы с пространством имен System.Web.Security, реализовать в приложении аутентификацию с помощью форм.

**После выполнения ЛР студенты должны уметь**

1. Создавать в БД условия для хранения данных о логине, пароле, ролях пользователей приложения.
2. Использовать алгоритмы хеширования паролей.
3. Добавлять настройки в файле конфигурации и атрибуты у методов контроллеров, необходимые для реализации разграничения функционала.
4. Создавать и использовать билет аутентификации.

**Ход работы**

1. Создать в используемой БД таблицы и атрибуты для хранения логина, хеша пароля, соли и ролей пользователя.
2. Открыть предыдущий проект ASP .NET MVC в среде Visual Studio.
3. Обновить модель БД, убедиться, что изменения в БД отражены в модели.
4. Указать настройки для авторизации форм в файле конфигурации и файле Global.asax.
5. Создать контроллер с методами авторизации и деавторизации и соответствующие представления.
6. Реализовать создание билета аутентификации в методе авторизации.
7. Реализовать разграничение доступности функционала для пользователей с разными ролями.

**Порядок выполнения работы**

* 1. **Базовые операции для работы с данными**

В рамках лабораторной работы вы продолжаете использовать предметную область, которую выбрали во время лабораторной №2.

***ВНИМАНИЕ!*** *В качестве источника данных настоятельно рекомендуется использовать базу данных, созданную в рамках выполнения ИДЗ по дисциплине «Технологии проектирования баз данных»*.

В данных методических материалах рассматривается предметная область о проведении опросов, и используются ранее созданные сущности: [Участник опроса]–[Ответ участника на вопрос]–[Вопрос] (рисунок 1).

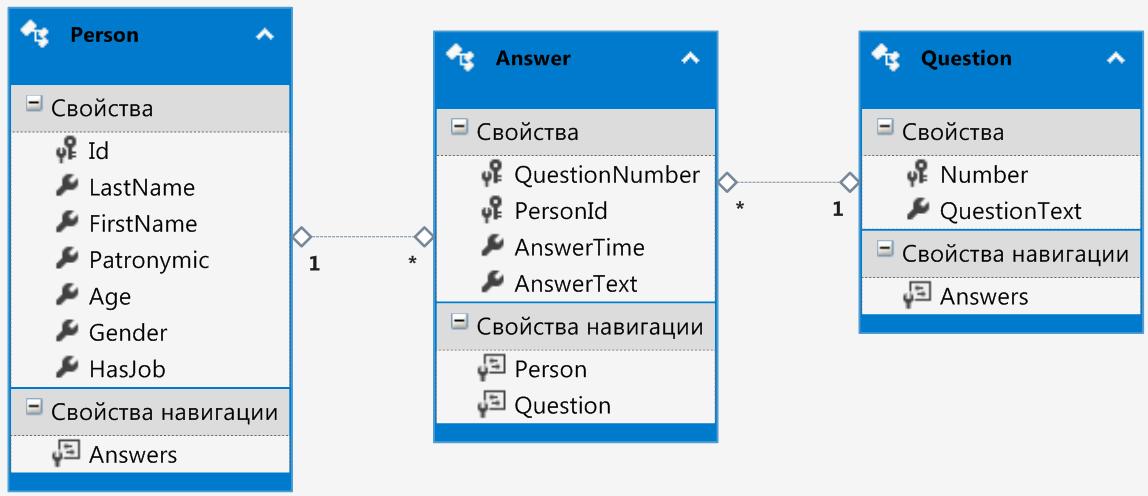


Рисунок 1 – Диаграмма используемой модели

Для возможности организации функционала авторизации пользователей в системе необходимо понять, в какой таблице хранится информация о пользователях и на какую «роль» будет назначен каждый из пользователей (например, администратор, гость, преподаватель, студент и тд).

Если все пользователи системы хранятся в одной таблице (например, в таблице Person), то в эту таблицу можно добавить атрибуты, хранящие логин, соль и захешированный пароль пользователя.

Если пользователи системы разнесены по нескольким таблицам, то можно добавить новую таблицу User, которая будет хранить логин, соль и захешированный пароль каждого пользователя, и в таблицах с пользователями необходимо добавить внешний ключ на данную таблицу (рисунок 2).

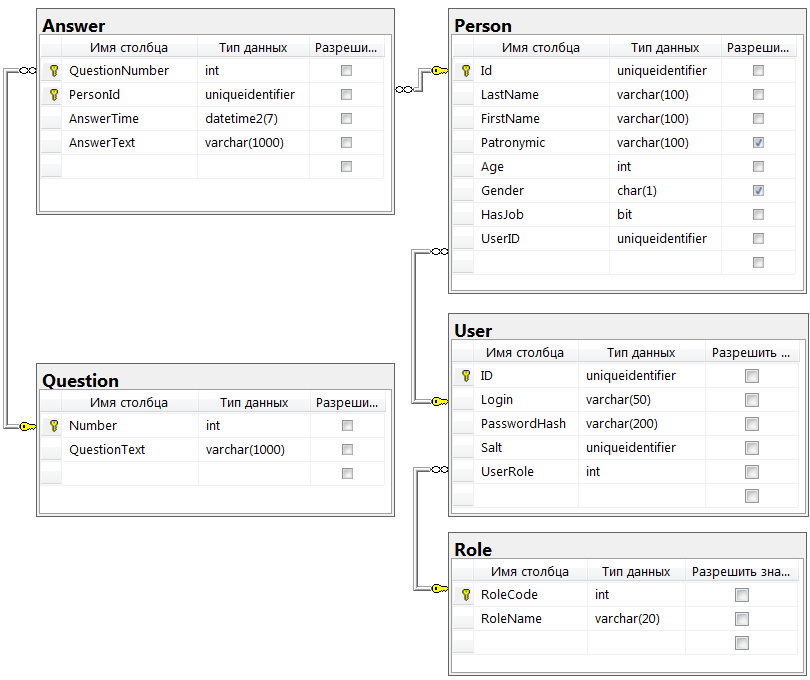


Рисунок 2 – Диаграмма базы данных

Для того, чтобы установить соответствие между пользователем системы и его ролью в системе, необходимо создать таблицу-справочник ролей, как это показано выше на рисунке 2 и сослаться на нее из таблицы пользователей.

Поле Salt («Соль») представляет собой уникальную GUID-последовательность для усложнения хеширования пароля.

PasswordHash («Хэш пароля»), как правило, не хранит сам пароль в целях безопасности, а хранит захешированную последовательность пароля, введенного пользователем при регистрации и значение соли, указанной для данного пользователя. В рамках данной работы в качестве PasswordHash при создании пользователей системы использовалась последовательность пароль + соль, захешированная алгоритмом SHA1.

Помимо алгоритма SHA1 для хеширования часто так же используются такие алгоритмы, как MD5, SHA256, SHA512.

Далее необходимо создать пользователей в БД. Для этого в MS Management Studio нужно написать код, приведённый на трисунке 3, и выполнить его для БД:

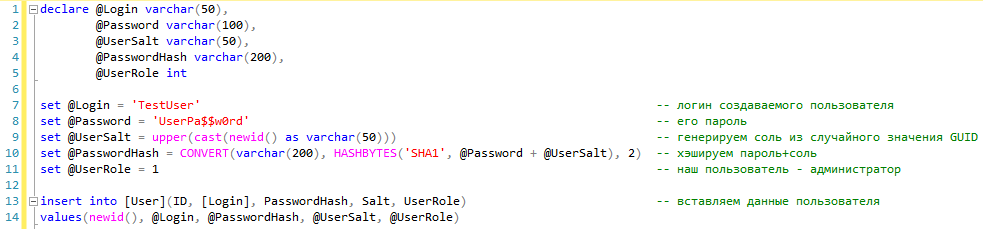


Рисунок 3 – Скрипт T-SQL для создания пользователя

Данные действия необходимо проделать необходимое количество раз (по количеству создаваемых пользователей), указывая каждый раз необходимое логин пользователя (переменная @Login) и его пароль (переменная @Password). Результаты выполнения скрипта необходимо проверить по таблице User.

Все описанные далее действия должны производиться в MS Visual Studio.

Для использования классов, позволяющих производить хешированние одним из вышеперечисленных алгоритмов, нужно подключить директиву using с пространством имён System.Security.Cryptography, в котором находится класс HashAlgorithm.

В рамках разрабатываемой системы ваши пользователи должны быть разделены, как минимум, на две разные роли для того, чтобы разграничить права доступа к некоторым методам веб-приложения. В итоге в приложении должны присутствовать как минимум три следующих типа пользователей:

* неавторизованный пользователь;
* авторизованный с ролью №1 (будет показан на примере пользователя с правами администратора);
* авторизованный с ролью №2 (будет показан на примере пользователя с минимальными правами).

После изменения схемы базы данных, необходимо обновить модель в среде Visual Studio. В итоге получится обновленная модель с необходимыми классами и свойствами (рисунок 4).

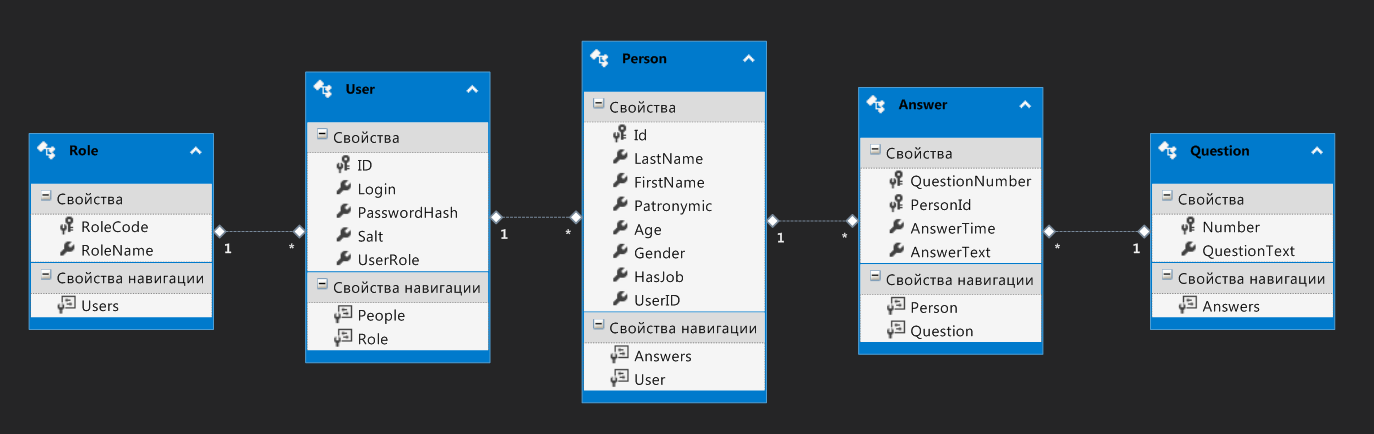


Рисунок 4 – Модель данных

Далее необходимо зайти в web.config (рисунок 5) и прописать все необходимые атрибуты, как указано на рисунке 6. О значении атрибутов вы можете узнать из лекции №6.

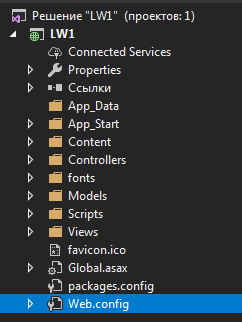


Рисунок 5 – Обозреватель решений

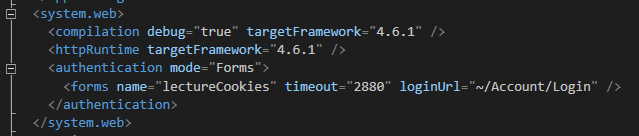


Рисунок 6 – Атрибуты тега <forms>

Далее необходимо открыть файл Global.asax (рисунок 7)

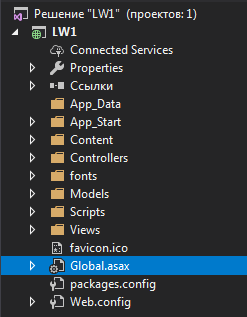


Рисунок 7 – Обозреватель решений

В открывшемся файле необходимо добавить метод парсинга ролей пользователей (рисунок 8), чтобы названия ролей можно было использовать при выставлении атрибутов у методов контроллеров.

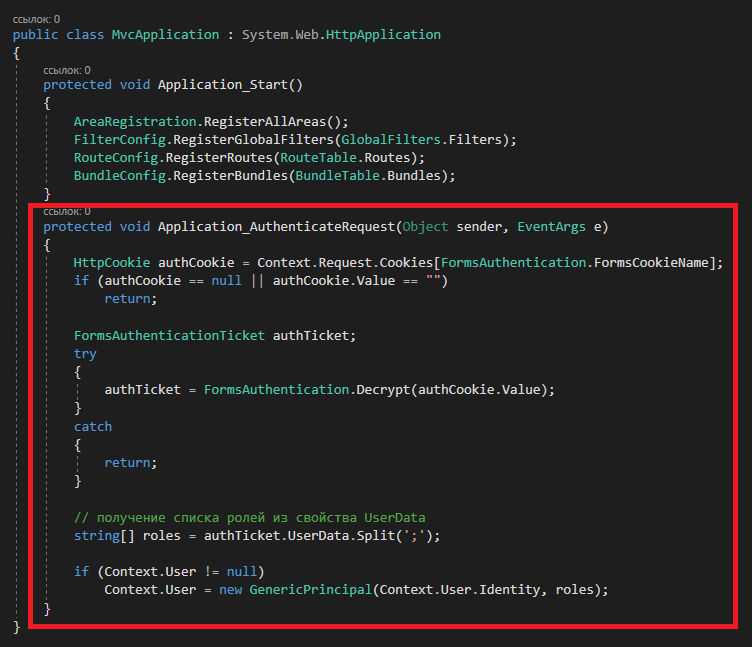


Рисунок 8 – Метод парсинга ролей

Для создания методов авторизации и деавторизации необходимо создать отдельный класс пользователя (модель представления) со свойствами «Логин» и «Пароль», а также контроллер, реализующий методы авторизации и деавторизации. Код класса представлен на рисунке 9, код контроллера представлен на рисунке 10 (внимание: в тексте функции используется метод ReturnHashCode, описание которого приведено далее).

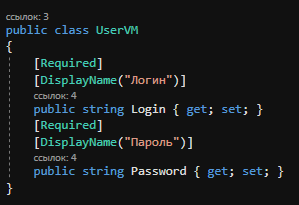


Рисунок 9 – Класс пользователя

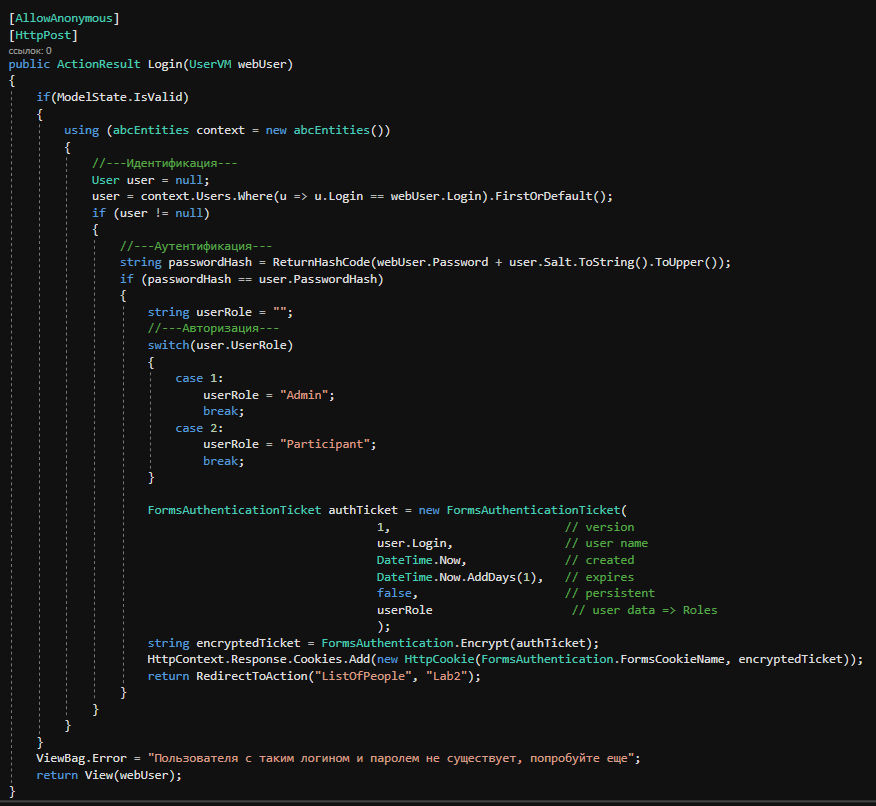


Рисунок 10 – Метод авторизации

Обратите внимание: на рисунке 10 приведён только код метода HttpPost. Создание метода HttpGet оставляется на самостоятельную проработку студенту.

В данном методе происходит поиск пользователя в базе с тем же логином, что был введен пользователем системы. Затем вызывается метод вычисления хеша для логина и соли найденного в базе пользователя. Если вычисленный и хранимый в базе хеши совпали, далее в зависимости от роли пользователя формируется билет аутентификации, содержащий имя пользователя, срок действия, роль, формируются cookies, а затем осуществляется перенаправление на метод, вызывающий нужное представление. Код метода хеширования (в том же контроллере) представлен на рисунке 11. Обратите внимание, что метод имеет уровень доступа private – Вам предлагается самостоятельно подумать, с какой целью.

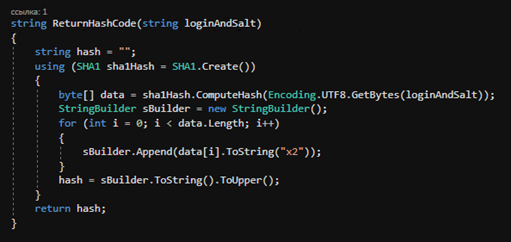


Рисунок 11 – Метод расчета хеша алгоритмом SHA1

Метод деавторизации находится в этом же контроллере и представлен на рисунке 12.

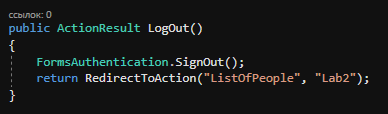


Рисунок 12 – Метод деавторизации

Далее для метода авторизации необходимо создать представление.

Код представления представлен на рисунке 13.

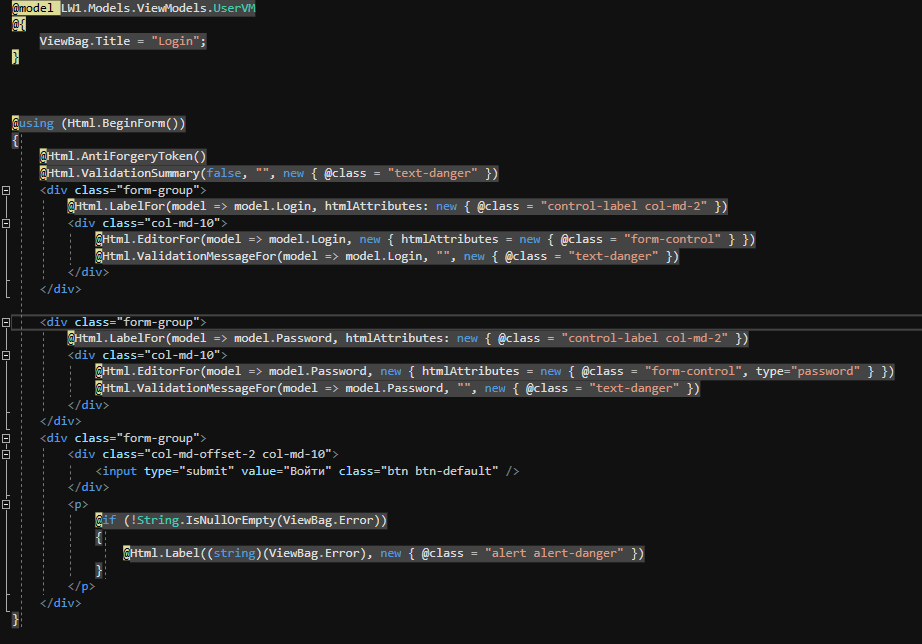


Рисунок 13 – Код представления для авторизации

Выглядит данное представление следующим образом (рисунок 14).

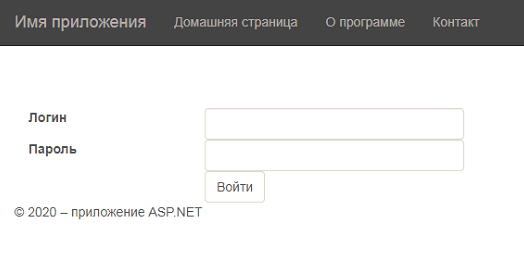


Рисунок 14 – Форма авторизации

Далее в файле \_Layout нужно добавить ссылку «Вход» и «Выход», в зависимости от того, авторизован пользователь или нет. Можно также добавить имя пользователя, если необходимо (рисунок 15).

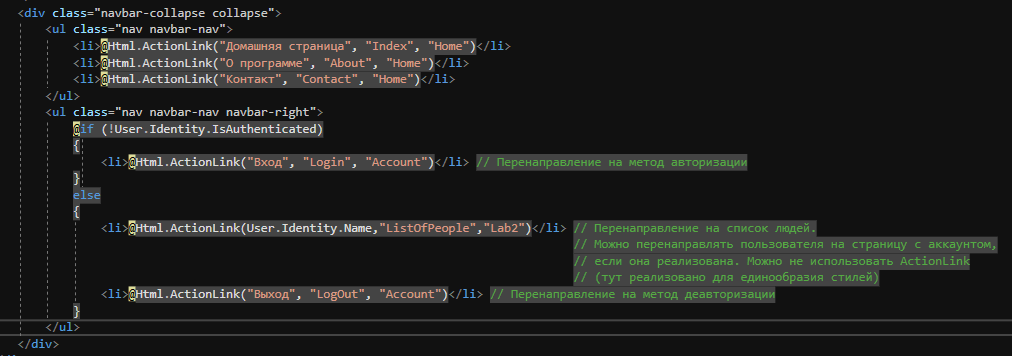


Рисунок 15 – Добавление кнопок для входа и выхода

Для того, чтобы добавить ограничения для разных ролей пользователей, применим следующую логику:

* Неавторизованному пользователю доступен список участников без возможности просмотра детальной информации, редактирования, удаления.
* Авторизованному пользователю с ролью №1 (здесь – сам участник) доступен просмотр списка участников, а также просмотр детальной информации о них.
* Авторизованному пользователю с ролью №2 (здесь – администратор) доступен полный функционал по работе со списком

Для реализации данной логики необходимо настроить ограничения отображения кнопок «подробнее», «редактировать», «удалить», а также добавить ограничения на сами методы контроллера, реализующие данные действия.

Код представлений с ограничениями по отображению представлен на рисунках 16-17.

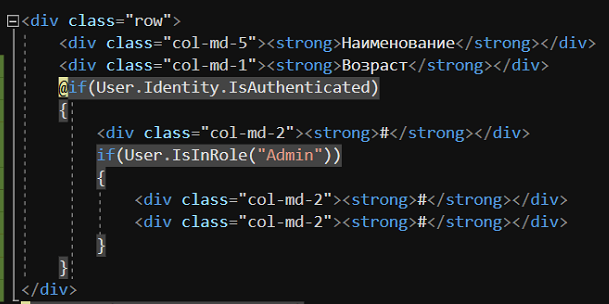


Рисунок 16 – Код представления

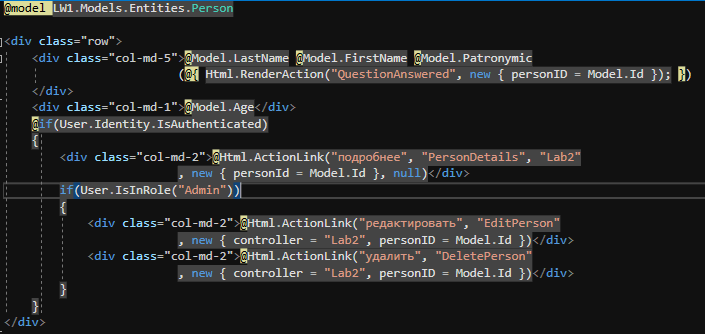


Рисунок 17 – Код частичного представления

Далее логика наложения ограничений на методы контроллера. Для тех методов, которые доступны всем, прописывается атрибут AllowAnonymous, здесь это метод, возвращающий представление со списком участников (рисунок 18).

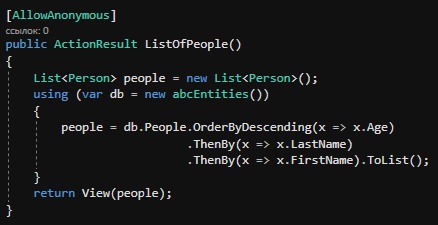


Рисунок 18 – Метод, доступный неавторизованным пользователям

Для тех методов, которые могут быть доступны только авторизованным пользователям (в данном случае метод просмотра детальной информации не требует определенной роли, он доступен всем тем, кто авторизован), прописывается атрибут Authorize (рисунок 19).

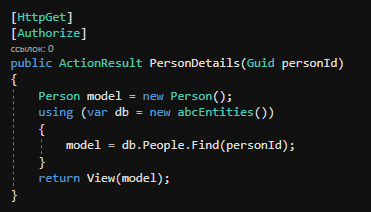


Рисунок 19 – Метод, доступный только авторизованным пользователям

Для тех методов, которые доступны только пользователям определенной роли, прописывается параметр Roles в атрибуте Authorize с названием роли, как это показано на рисунке 20. В данном случае, эту строку нужно добавить перед методами EditPerson и DeletePerson.

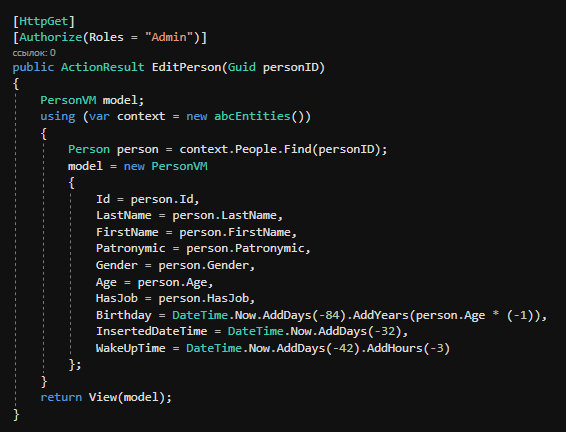


Рисунок 20 – Метод, доступный только авторизованному пользователю с ролью «Admin»

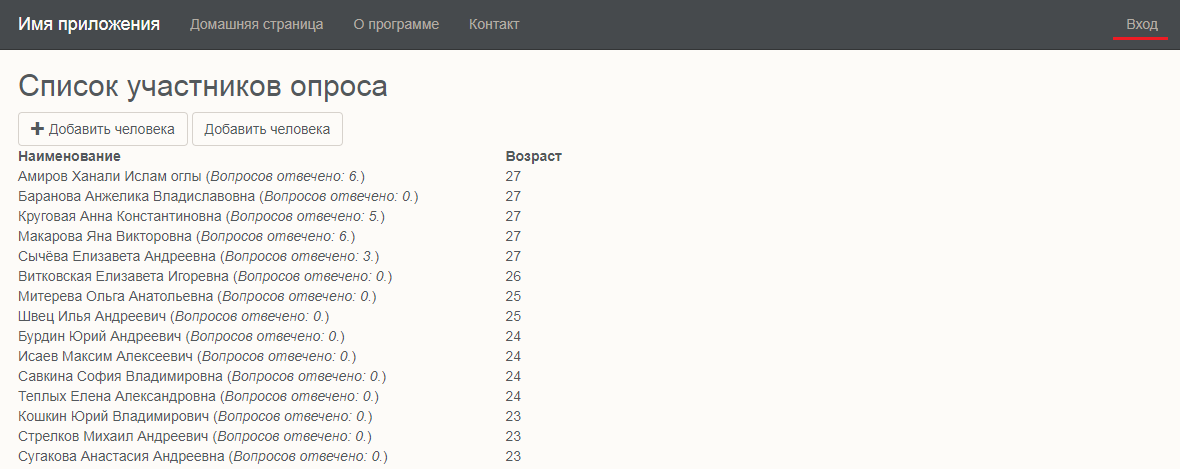
Таким образом, при запуске программы можно наблюдать следующее:

Рисунок 21 – Просмотр страницы неавторизованным пользователем

Обратите внимание на то, что стала доступна кнопка «Вход».

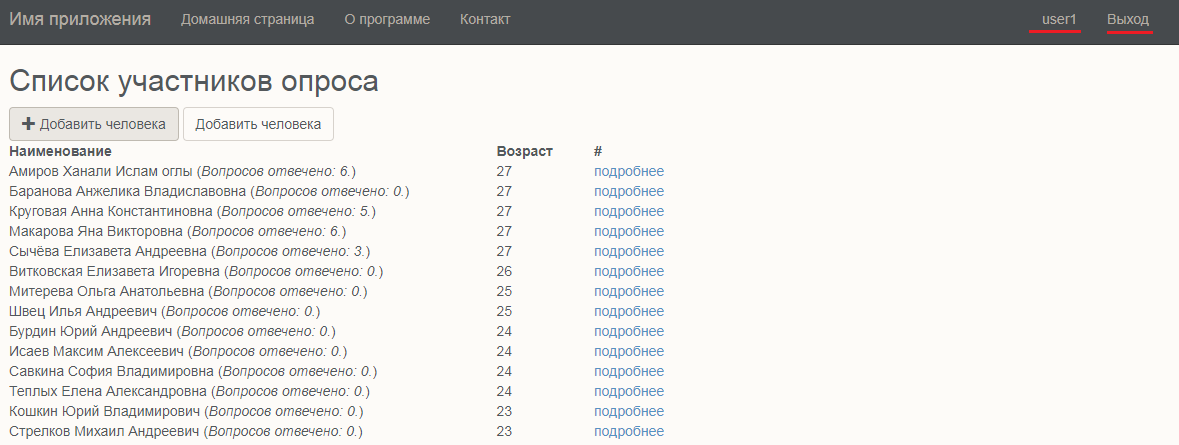


Рисунок 22 – Просмотр страницы участником (user1 – участник)

Обратите внимание, что в шапке теперь отображается имя пользователя и есть кнопка «Выход».

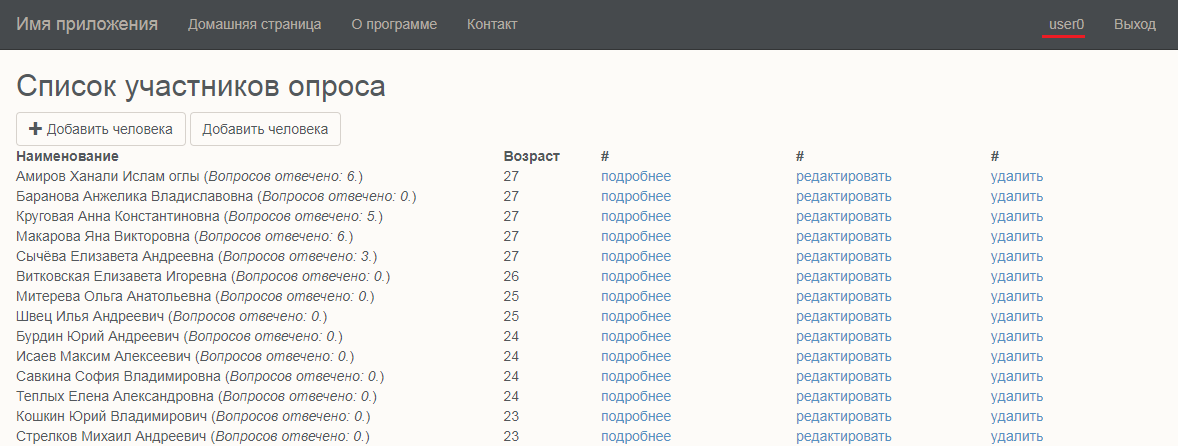


Рисунок 23 – Просмотр страницы администратором (user0 – администратор)

**Оформление отчетов**

Отчет о выполнении лабораторной работы оформляется в печатном виде, должен включать титульный лист и состоять из следующих структурных элементов и разделов:

– цель работы;

– постановка задачи;

– порядок выполняемых действий с описанием;

– выводы по работе.