Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Уральский государственный лесотехнический университет»
(ФГБОУ ВПО «УГЛТУ»)

УДК 002.6:004.89
№ госрегистрации 114070350023
Инв. № 005/2019

УТВЕРЖДАЮ
Проректор по научной работе
д-р сель.-хоз. наук, профессор
_____ С.В.Залесов 2019 г.
«____» _____

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Проект № 2342: Методология и технология проектирования самонастраивающихся нечетких моделей и баз знаний для сайтов 2.0 информационных систем поддержки принятия решений в лесном комплексе

Этап: Разработка и отладка программных модулей создания и управления моделями и базами знаний в среде самонастраивающихся нечетких моделей

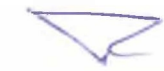
(промежуточный)

Начальник НИЧ,
канд. тех. наук, доцент _____ А.И. Сафронов
подпись, дата

Научный руководитель темы
док. тех. наук, профессор _____ В.П. Часовских
подпись, дата

Екатеринбург 2019

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель темы, д-р технических наук		В.П.Часовских (введение, раздел 3-5, заключение)
Исполнители темы		М.П.Воронов (раздел 1,2)
Нормоконтролер		Н.В. Терещенко
Соисполнители:		
Профессор, Доктор сель. -хоз. наук		В.А.Усольцев (раздел 2)
Аспирант		Д.А. Стаин (раздел 1,2)

Реферат

Отчет 149 с.,72 источника. Модели, базы знаний, сложные системы, управление, условия неопределенности, самонастраивающиеся нечеткие модели, принятие решений, лесной комплекс, динамические сайты, облачные технологии, виртуальные тренажеры, интеллектуальная компьютерная система.

Объектом исследования являются современные информационно-коммуникационные технологии принятия решений в условиях неопределенности.

В результате исследования впервые были созданы программы для ЭВМ и базы данных с динамическим формированием навигации сайта и адаптивной настройкой поиска в базе данных с меньшими временными затратами. Построены адаптивные сайты типа 2.0 в среде ASP.NET MVC для формирования моделей объектов внешней среды, их последующей обработки, виазуализации и принятия решений. Полученные результаты защищены свидетельствами о регистрации программ для ЭВМ и баз данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

При создании современных систем принятия решений в лесном секторе экономики, в условиях постоянного повышения качества принимаемых решений, Интернет, сайт организации, автоматизированные системы управления предприятий базируются на все более совершенных математических и информационных моделях, и все более обширная сфера факторов влияния используется в качестве исходных данных анализа. Современные интеллектуальные системы принятия решений на основе анализа прошлых тенденций и анализа эффекта от уже принятых решений, ставят перед необходимостью хранения и обработки все больших объемов данных.

В предыдущих отчетах мы показали, что в современных информационных системах поддержки принятия решений в лесном секторе экономики основу составляют сайт 2.0, модели и базы знаний. Большой интерес представляет методология создания подобных систем, в которых требуется непосредственное практическое участие лица, принимающего решение, и не приводящих к фатальным последствиям в случае ошибки. В настоящее время наиболее интересными, в прикладном аспекте, являются информационные системы, в которых применяется технология сайта 2.0 и виртуальных тренажеров, облачных вычислений на динамически развивающихся сайтах, что позволяет выполнять имитацию изменения состояний управленческих объектов в зависимости от действий принимающего решения. Интересной является, и сфера обучения управляющих в лесном комплексе (Global Management Challenge). Возникает потребность в системах, настраивающихся на сферы применения и решаемые задачи, с использованием нечетких моделей в среде динамически развивающихся сайтов. Актуальность данного утверждения существенно увеличилась с внедрением в систему высшего образования ФГОС 3+. Лицу, принимающему решение или обучаемому будет предоставлена такая же свобода действий, как и при работе с реальным объектом. Он не ограничен жесткой последовательностью действий. Кроме того, многие ситуации, моделируемые в рамках управленческих дисциплин (менеджмент, маркетинг) зачастую содержат формулировки и требуют решений, выраженных не в четкой количественной форме, а в виде нечеткой информации, или лингвистической конструкции (например, повысить качество продукции, сократить издержки не менее, чем на 10% и т.д.).В данном разрезе исследование системных связей и функциональных закономерностей информационных систем, как средств повышения производительности обработки информации (и как следствие, повышение конкурентоспособности предприятия) представляют широкий практический и научный интерес.

Основу любой автоматизированной системы управления производством (АСУП), а также ее наиболее современной модификации - корпоративной информационной системы (КИС),

составляют ее информационные модели и средства их обработки, представленные в совокупности баз данных (БД) и баз знанмй (БЗ).

Применение СУБД позволяет существенно повысить надежность и эффективность обработки информации в сложных информационных системах, сократить сроки и затраты на их проектирование, внедрение и эксплуатацию. Еще большая эффективность от применения функционально развитых СУБД может быть получена на этапах их внедрения и эксплуатации в распределенных системах обработки данных за счет использования типовых проектных решений для узлов сети и централизации всех этапов жизненного цикла СУБД. При создании корпоративных информационных систем оптимизация процедур обмена данными в распределенной среде приобретает все большую актуальность.

Немаловажным фактором, определяющим эффективность КИС, является выбор СУБД и среды разработки и функционирования программных элементов информационной системы. Характерной чертой современных СУБД является их ориентация на решение прикладных задач, требующих возможность нестандартной обработки данных, а также возможность изменения пользователем требований к прикладным задачам в отношении способа обработки данных, структуры связей между объектами и выходных форм отчета. Построение КИС на основе современных СУБД осуществляется с помощью высокоуровневых, интегрированных с базой данных языков программирования. При этом эффективность сообщения БД со средой разработки информационной системы во многом определяется возможностями этих языков.

Успешность решения сформулированных задач определяется применяемой средой и технологией перехода от предметной области, через модели и программы к эффективной системе принятия решений[1-37]. Анализ среды проектирования [7-35] определил выбор технологию объектно-ориентированного программирования. Наилучшим образом это выбор обеспечивается технологией ASP.NET MVC 5 фирмы Microsoft с использованием комплекса проектирования и сопровождения Visual Studio 2017[28-72].

В данном отчете приводятся результаты исследований разработки и отладки программных модулей создания и управления моделями и базами знаний в среде де ASP.NET MVC и Visual Studio.

1. Обобщенная модель информационной системы для принятия решений

Системы поддержки принятия решений (DSS) - это компьютерные системы, почти всегда интерактивные, разработанные, чтобы помочь менеджеру в принятии решений. DSS включают и данные, и модели, помогающие ЛПР решить проблемы, особенно те, которые плохо формализованы. Данные часто извлекаются из системы диалоговой обработки запросов или базы данных. Модель может быть простой типа "доходы и убытки", чтобы вычислить прибыль при некоторых предположениях, или комплексной типа оптимизационной модели для расчета загрузки для каждой машины в цехе. Системы поддержки принятия решений не всегда оправдываются традиционным подходом «стоимость – прибыль». Для этих систем многие из выгод явным образом не осязаемы, однако они необходимы для более глубокого проникновения в процесс принятия решения и лучшего понимания данных. И эффективность решения будет заключаться в степени достижения целей, отнесенной к затратам на их достижение.

На рис. 1 представлены компоненты системы поддержки принятия решений

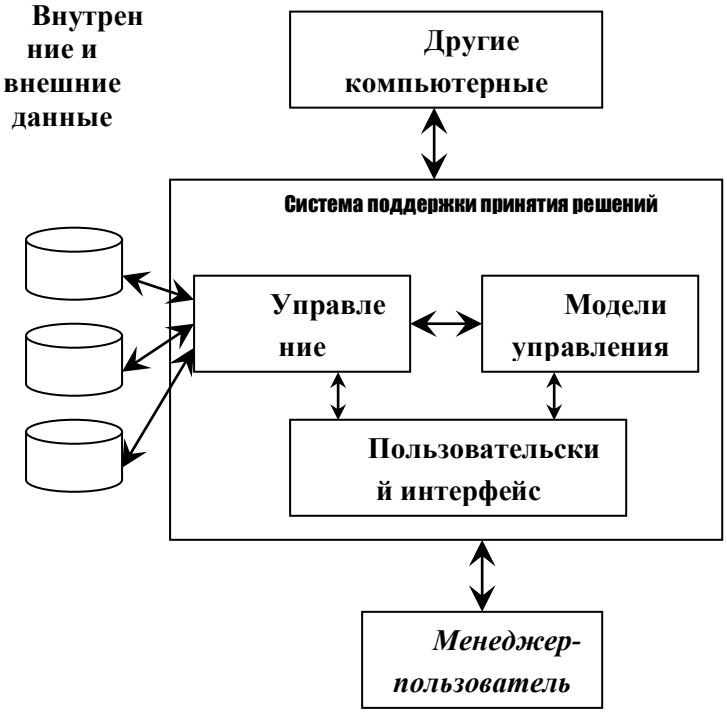


Рис. 1. Компоненты системы поддержки принятия решений

Рис. 1 показывает, что система поддержки принятия решений требует трех первичных компонентов: модели управления, управления данными для сбора и ручной обработки данных и управления диалогом для облегчения доступа пользователя к DSS. Пользователь взаимодействует с DSS через пользовательский интерфейс, выбирая частную модель и набор данных, которые нужно

использовать, а затем DSS представляют результаты пользователю через тот же самый пользовательский интерфейс. Модель управления и управление данными в значительной степени действуют незаметно и варьируются от относительно простой типовой модели в электронной таблице до сложной комплексной модели планирования, основанной на математическом программировании. Исследования показали, что наилучшей средой разработки подобных систем принятия решений является объектно-ориентированный подход, реализованные в ASP.NET MVC 5 Microsoft.

Все DSS. помогают в процессе принятия решения. Напротив, генератор системы поддержки принятия решений – это система, которая обеспечивает набор возможностей быстро и легко строить специфические DSS. Подобный генератор реализован нами как Web компонент ASP.NET MVC 5.

Развитие системы принятия решения реализуется средствами исполнительных информационных систем (Executive Support System – ESS). Ключевая концепция исполнительной информационной системы состоит в том, что такая система поставляет интерактивную совокупность текущей информации относительно конъюнктур рынка, формирует легкий доступ для старших руководителей и других менеджеров без помощи посредников. ESS, реализованная в среде ASP.NET MVC 5 использует современную графику, связь и методы хранения данных, обеспечивая исполнителям легкий интерактивный доступ к текущей информации относительно состояния организации.

Как показали наши исследования так называемые экспертные системы и нейронные сети наиболее пригодны для поддержки управления. Экспертные системы — это системы, которые используют логику принятия решения человеческого эксперта. Самая новая отрасль AI - нейронные сети, которые устроены по аналогии с тем, как работает человеческая нервная система, но фактически используют статистический анализ, чтобы распознать модели из большого количества информации посредством адаптивного изучения.

В настоящее время в области разработки систем искусственного интеллекта сложилась следующая аксиома: никакой, самый сложный и изощренный алгоритм извлечения информации (так называемый механизм логического вывода) из интеллектуальной системы не может компенсировать «информационную бедность» ее базы знаний. Несмотря на широкое распространение и использование понятия «знания» в различных научных дисциплинах и на практике, строгого определения данного термина нет.

Довольно часто используют так называемый прагматический подход: говорят, что *знания* — это формализованная информация, на которую ссылаются и/или которую используют в процессе логического вывода. Однако такое определение ограничено: оно фиксирует сознание на уже существующих методах представления о знаниях и, соответственно, механизмах вывода, не давая возможности представить себе другие («новые»).

2. Модели принятия решений в неопределеной среде средствами современных web-технологий

Для конкретизации и подходящей интерпретации класса предлагаемых моделей среды неопределенности рассмотрим образовательный процесс образовательной организации.

В соответствии с ФГОСами, применяется компетентностный подход к вопросу о возможности присуждения квалификации студенту, который в полной мере справился с образовательной программой по тому или иному направлению подготовки. Таким образом, образовательный процесс можно рассматривать как процесс приобретения компетенций вследствие изучения отдельных дисциплин в составе образовательной программы. Во ФГОС реализована матрица компетенций, в которой определены: какими компетенциями должен владеть выпускник для приобретения специальности, соответствующей данной дисциплине. Данная структура также определяет, какие дисциплины учебного плана определяют приобретение соответствующих компетенций.

Предлагается использовать матрицу компетенций (устранение неопределенности) как базовую структуру математической модели управления образовательным процессом вуза. Абитуриент, недавно поступивший в вуз, имеет пустую матрицу компетенций, т.е. все ее элементы равны 0. Каждый семестр студент изучает дисциплины и в конце семестра в период сессии осуществляется контроль. В случае успешной сдачи дисциплины, соответствующие компетенции получают значения 1.

Пусть K – матрица компетенций некоторого студента.

$$K = \begin{pmatrix} k_{11} & k_{12} & k_{13} & \cdots & k_{1m} \\ k_{21} & k_{22} & k_{23} & \cdots & k_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ k_{n1} & k_{n2} & k_{n3} & \cdots & k_{nm} \end{pmatrix}$$

Столбцы матрицы *K* будут обозначать компетенции, строки – дисциплины. В рамках образовательной программы студент должен приобрести m компетенций, освоив n дисциплин. Таким образом k_{ij} – состояние j-й компетенции в рамках изучения i-й дисциплины.

$$k_{ij} = \begin{cases} 1, \text{если } j \text{ компетенция успешно освоена при изучении } i-\text{й дисциплины} \\ 0, \text{если } j \text{ компетенция не освоена при изучении } i-\text{й дисциплины} \end{cases}, \text{ где}$$

$$\begin{cases} 1 \le i \le n \\ 1 \le j \le m \end{cases}$$

Каждая дисциплина направлена на формирование определенных компетенций. Запишем матрицу компетенций T, определяющую набор компетенций, которые должен приобрести студент в процессе обучения

$$T = \begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1m} \\ t_{21} & t_{22} & t_{23} & \cdots & t_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ t_{n1} & t_{n2} & t_{n3} & \cdots & t_{nm} \end{pmatrix}$$

T определяется стандартом. Назовем ее целевой матрицей компетенций.

Пусть $t_{общ}$ – нормативный срок овладения образовательной программой.

Определим функцию K(t), которая возвращает матрицу компетенций, в которой отражены освоенные компетенции на данный момент времени. Такую матрицу компетенций будем называть фактической матрицей компетенций на момент времени t.

Очевидно, что образовательную программу можно считать успешно освоенной при условии

$$K(t_{\text{общ}}) = T$$

Если такое равенство не выполняется, очевидно, студент не справился с образовательной программой. Т.к. в данном случае $t \to t_{\text{общ}}$, управляющее воздействие может быть только одно – отчислить студента. Такое управление не повышает эффективность образовательного процесса, а как следствие, необходимо ввести временные интервалы управления.

Пусть t_1, $t_2 \ldots t_l$ – временные интервалы управления. В образовательном процессе высшей школы целесообразно ограничиться номером семестра, т.е. t_i – временной интервал окончания i-го семестра. Образовательный процесс ограничен *l* семестрами

Введем функцию $T(t_i)$, которая определяет целевую матрицу компетенций, приобретенных студентом за временной интервал t_i. Таким образом

$T(t_1) \oplus T(t_2) \oplus \ldots \oplus T(t_l) = T(t_{\text{общ}}) = T$, где $\oplus$ - операция дизъюнкции матриц, а T является интегрированной матрицей для $T(t_i)$.

В контексте использованного выше математического аппарата, можем представить систему управления образовательным процессом в вузе следующим образом:

$$K(t) \to T \text{ при } t \to t_{\text{общ}}$$

Крайними точками образовательного процесса являются $K(t_0)$ и T, где

$$K_0 = K(t_0) = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

Все промежуточные состояния растянуты во времени, равном нормативному сроку обучения $t_{общ}$. В этом промежутке времени образовательный процесс нуждается в управляющих действиях.

Для того, чтобы определить, какими компетенциями студент не смог овладеть на данном временном интервале управления, введем понятие матрицы отклонения и обозначим ее ΔK.

$$\Delta K = \begin{pmatrix} \Delta k_{11} & \Delta k_{12} & \Delta k_{13} & \cdots & \Delta k_{1m} \\ \Delta k_{21} & \Delta k_{22} & \Delta k_{23} & \cdots & \Delta k_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \Delta k_{n1} & \Delta k_{n2} & \Delta k_{n3} & \cdots & \Delta k_{nm} \end{pmatrix}$$

Матрица отклонений ΔK используются для генерирования управляющего действия и показывает, какие компетенции, указанные в целевой матрице, отсутствуют в фактической. Если в фактической матрице присутствуют компетенции, отсутствующие в целевой матрице, то управляющего сигнала такая ситуация не требует и, как следствие, соответствующие компетенции не должны выделяться в матрице отклонений.

Предлагается следующая логическая формула для вычисления матрицы отклонений:

$$\Delta K(t) = \overline{(T(t) \to K(t))}$$

В данной формуле применяется логическое «НЕ» от логической импликации целевой и фактической матриц.

$\Delta K(t)$ будет содержать 1 на позициях, где в $K(t)$ отсутствуют компетенции, указанные в целевой матрице. В остальных позициях $\Delta K(t)$ будет содержать 0.

Для демонстрации, рассмотрим четыре возможных ситуации, которые могут иметь место в реальном образовательном процессе в силу человеческого фактора.

Первая ситуация.

Студент не овладел компетенцией, требуемой в рассматриваемом интервале управления. Иначе говоря, фактическая матрица $K(t)$ не содержит компетенции, которую содержит целевая матрица $T(t)$.

$$T(t) = \begin{pmatrix} 1 & \mathbf{1} \\ 0 & 1 \end{pmatrix}$$

$$K(t) = \begin{pmatrix} 1 & \mathbf{0} \\ 0 & 1 \end{pmatrix}$$

$$\Delta K(t) = \overline{(T(t) \to K(t))} = \overline{\begin{pmatrix} 1 & \mathbf{0} \\ 1 & 1 \end{pmatrix}} = \begin{pmatrix} 0 & \mathbf{1} \\ 0 & 0 \end{pmatrix}$$

Имеем 1 на позиции компетенции, которой студент не овладел. Требуется управляющее действие на ликвидацию отклонения.

Вторая ситуация.

Студент овладел всеми компетенциями, которые требуются в рассматриваемом интервале управления. Иначе говоря, фактическая матрица $K(t)$ идентична целевой матрице $T(t)$.

$$T(t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$K(t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Delta K(t) = \overline{(T(t) \to K(t))} = \overline{\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Имеем нулевое отклонение, т.е. управляющее действие не требуется.

Третья ситуация.

Студент овладел всеми компетенциями, которые требуются в рассматриваемом интервале управления, а также компетенциями, которые отсутствуют в целевой матрице. Иначе говоря, фактическая матрица $K(t)$ содержит компетенции, которые не содержит целевая матрица $T(t)$.

$$T(t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$K(t) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\Delta K(t) = \overline{(T(t) \to K(t))} = \overline{\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Имеем нулевое отклонение, т.е. управляющее действие не требуется.

Четвертая ситуация.

Является составной из первой и третьей ситуаций – студент не овладел рядом компетенций, содержащихся в целевой матрице, но овладел компетенциями, которые в целевой матрице отсутствуют.

$$T(t) = \begin{pmatrix} 1 & \mathbf{0} \\ 0 & \mathbf{1} \end{pmatrix}$$

$$K(t) = \begin{pmatrix} 1 & \mathbf{1} \\ 0 & \mathbf{0} \end{pmatrix}$$

$$\Delta K(t) = \overline{(T(t) \to K(t))} = \overline{\begin{pmatrix} 1 & \mathbf{1} \\ 1 & \mathbf{0} \end{pmatrix}} = \begin{pmatrix} 0 & \mathbf{0} \\ 0 & \mathbf{1} \end{pmatrix}$$

Имеем 1 на позиции компетенции, которой студент не овладел. Требуется управляющее действие на ликвидацию отклонения.

Введем функцию суммы элементов матрицы S(M). Чем меньше численно сумма элементов матрицы отклонения, тем меньше отклонение от образовательной траектории. В идеальной ситуации, когда все компетенции осваиваются полностью и в срок, $\mathrm{S}(\Delta K(t)) = 0$ для любого t. Таким образом, предлагается использовать $\mathrm{S}(\Delta K(t))$ в качестве целевой функции системы управления и повышения эффективности образовательного процесса:

$$\mathrm{S}\,(\Delta K(t)) = \sum_{i=1}^{n} \sum_{j=1}^{m} \Delta k_{ij} \to min$$

В случае, когда матрица отклонений ненулевая (первая и четвертая ситуации), требуется сгенерировать управляющий сигнал, распространяющегося на временной интервал, следующий за t (в случае, если это не превышает нормативный срок обучения $t_{общ}$). В данном случае целесообразно применение аппарата анализа и принятия решений. Каждый вуз определяет параметры генерирования такого сигнала. Если $\mathrm{S}\,(\Delta K(t)) \gg 0$, то студента целесообразно отчислить (например, если он имеет больше трех долгов по дисциплинам образовательной программы).

Таким образом, имеем систему управления, которая обладает средствами мониторинга, обратной связи и средствами управления (Рисунок 2.1)

Рисунок 2.1 – Структурно-функциональная модель образовательного процесса.

Рассмотренная выше модель образовательного процесса релевантна компетентностному подходу в высшем образовании. Она также позволяет реализовать пункт 7.1.2 ФГОС 3+, который предписывает организациям фиксировать образовательный процесс на сайте университета. Таким образом, имеем информационную систему с большим количеством данных, их взаимосвязей и математическую модель которые частично визуализируются по неопределенному кругу лиц посредством технических средств. Эффективно решить данную проблему возможно только с

применением современных web-технологий. Необходимо погрузить сгенерированную модель в среду web-технологий. На Рисунке 2.1 наглядно показано, как образовательный процесс, а также модели и методы и методы им встроены в электронную информационно-образовательную среду университета, представляющую из себя web-структуру.

3. Семантический Web-сайт – среда повышения эффективности доступа к данным

Появление Web-сайта все изменило. Базы данных и базы знаний стали доступными в Web-сайте, который является открытой средой: пользователи, прикладные программы, способы использования данных постоянно меняются. В этих условиях модель данных и её семантика данных должна быть доступна непосредственно вместе с самими данными. Доступность семантики для пользователей при визуализации средствами Web-сайта можно обеспечить за счет выбора соответствующего формата представления и визуализации информации. Однако для контроллера и представлений Web-сайта семантика должна быть представлена в формальной, машинно-обрабатываемой форме. Именно этот факт обусловил появление так называемого семантического Web-сайта (англ. Semantic Web).

Семантический Web-сайт, также известный в последние годы как веб данных (англ. Web of Data), основан на различных принципах проектирования, которые можно обобщить следующим образом:

- обеспечение доступности в Web-сайте структурированных и частично структурированных данных, представленных в стандартных форматах;
- обеспечение доступности в Web-сайтае не только наборов данных, но и отдельных элементов данных и отношений между ними;
- описание предполагаемой семантики таких данных с помощью формализма, обеспечивающего возможность машинной обработки этой семантики.

Решение использовать структурированные и частично структурированные данные основано на результатах исследований, а именно на том факте, что на сегодняшний день основой частично структурированного «веба текста и изображений» является на самом деле очень большое количество структурированных и частично структурированных документов.

В основном содержание современных веб-страниц генерируется на основе баз данных и систем управления контентом, содержащих хорошо структурированные наборы данных. Однако структурные связи, существующие в таких наборах данных, полностью теряются при визуализации их в виде веб-страниц на языке разметки гипертекста HTML (Hypertext Markup Language).

Для того чтобы сделать Web-сайт более «семантическим», необходимо

реализовать следующую ключевую идею: визуализировать и связывать друг с другом структурированные наборы данных (вместо того чтобы визуализтровать и связывать между собой HTML-страницы, после того как потеряна большая часть структурных связей данных).

Три вышеупомянутых принципа проектирования реализованы с помощью следующих конкретных технологий:

- В качестве модели данных для описания объектов и отношений между ними используются помеченные графы. Объекты представляются как вершины графа, а отношения между ними - как дуги.
- Для идентификации отдельных элементов данных и отношений между ними, которые включаются в наборы данных, используются веб-идентификаторы URI (Uniform Resource Identifier), возможно теги разметки(семантической).

В качестве модели данных, позволяющей формально представить предполагаемую семантику этих данных, используются онтологии (если говорить кратко: иерархически организованные словари терминов - типов объектов и отношений между ними - свойств). URI используются для идентификации терминов и их свойств.

В разделе 1 данного отчета определена особенность структур данных в информационных системах организации лесного сектора экономики. Эта структурная особенность позволяет предложить новый, более эффективный метод организации данных в СУБД. Предлагаемый метод не зависит от предметной области (организация лесного сектора экономики или государственная организация типа университета) и определяется только структурой дескрипторов поиска в базе данных. На данный момент Web-сайт - это сайт текста и изображений. Такие мультимедийные материалы полезны для людей, но контроллеры и представления играют очень ограниченную роль в сайте: они индексируют сайты по ключе¬вым словам, передают информацию от серверов к клиентам, но этим и ограничиваются их функции. Вся интеллектуальная работа (выбор информации, ее объединение, агрегирование и т. д.) выполняется че¬ловеком. Но если сделать Web-сайт более информативным для контроллеров и представлений, наполнить его машиночитаемыми, то есть «понятными» для машин, данными то становиться возможным реализовать функции, осуществление которых невозможно в настоящее время. Поиск и извлечение информации не бу¬дет больше ограничиваться просто поиском, по ключевым словам, становиться возможным учитывать синонимы, исключать омонимы, а также учитывать контекст и цель поисковых запросов. Web-сайты могут стать более персонализированными, если браузеры смо¬гут понимать содержимое веб-страницы и

адаптировать его к личным интересам пользователя в соответствии с его профилем. Взаимосвязь страниц Web-сайта может стать более осмысленным, если реализовать динами-ческое создание полезных ссылок на основе действий конкретного пользователя, вместо того заранее создать одинаковые ссылки для всех пользователей. Станет возможна интеграция данных (информации) с разных Web-сайтов.

При реализации эффективного и качественного функционирования любого хозяйствующего субъекта, будь то материальное производство или университет, имеет смысл применять методы и средства автоматизированных систем управления (АСУ). Одной из основных функциональных составляющих любой АСУ является база данных (БД), а также набор средств и методов доступа, которые предоставляет система управления базами данных (СУБД).

Структуры данных АСУ вуза в Российской Федерации концептуально следуют из законодательных актов и нормативных документов [38-42]. Анализ законодательства в контексте структур данных АСУ вуза выполнены в [53]. Особенности мебельного производства подробно рассмотрены в [43,49]

Формулирование общих зависимостей в моделях данных позволит сгенерировать особые алгоритмы функционирования СУБД, применение которых повысит эффективность функционирования базы данных. Общие концепции баз данных, реляционных таблиц и языка запросов T-SQL рассмотрены в [61,62,63,64,68].

В таблице (слева) представлено отношение Q, которое обладает всеми свойствами базы данных АСУ вуза или АСУ мебельного производства. Количество значений полей a и b значительно меньше, чем количество записей в базе данных. Поле N отображает те поля таблиц, количество значений которых соизмеримо с количеством записей в БД или уникально.

N	a	b
1	a1	b1
2	a1	b2
3	a1	b1
4	a2	b2
5	a2	b2
6	a2	b1
7	a1	b2
8	a1	b1

N	Np	a	b
1	1	a1	b1
3	2	a1	b1
8	3	a1	b1
2	4	a1	b2
7	5	a1	b2
6	6	a2	b1
4	7	a2	b2
5	8	a2	b2

Таблица. Исходное отношение Q (слева) и отсортированное Qsort (справа).

Отсортируем исходное отношение по не убыванию с сохранением исходной нумерации N, добавим нумерации отсортированного отношения Np (таблица справа) и запишем в новую таблицу (усовершенствованный индекс – отношение I_a, Схема справа) только уникальные значения дескрипторов. В дополнительном поле перечислим адреса участков памяти в исходном отношении, в которых данное значение встречается. В результате, для выборки произвольных значений a и b в исходном отношении, необходимо просмотреть все исходное отношение. Воспользовавшись усовершенствованным индексом, необходимо просмотреть, в среднем, половину усовершенствованного индекса, т.к. он отсортирован по неубыванию. Конечно, в конкретных случаях распределение вероятностей может быть неравномерным. Так, если вероятность выборки любой записи исходного отношения Q равновероятна и равна (100/8)%=12,5%, то сравнение искомого значения с первой строчкой в усовершенствованном индексе снизит неопределенность на (12,5*3)%=37,5% (по количеству указателей в поле ptr), а сравнение искомого значения с 3-й строчкой снизит неопределённость только на 12,5%. Тем не менее, допустим, что распределение вероятностей в I_a равномерно, тогда можно сказать, что для поиска произвольной записи стандартным методом необходимо просмотреть все 8 записей исходного отношения Q, а с использованием усовершенствованного индекса I_a необходимо просмотреть, в среднем, 2 записи (Схема)

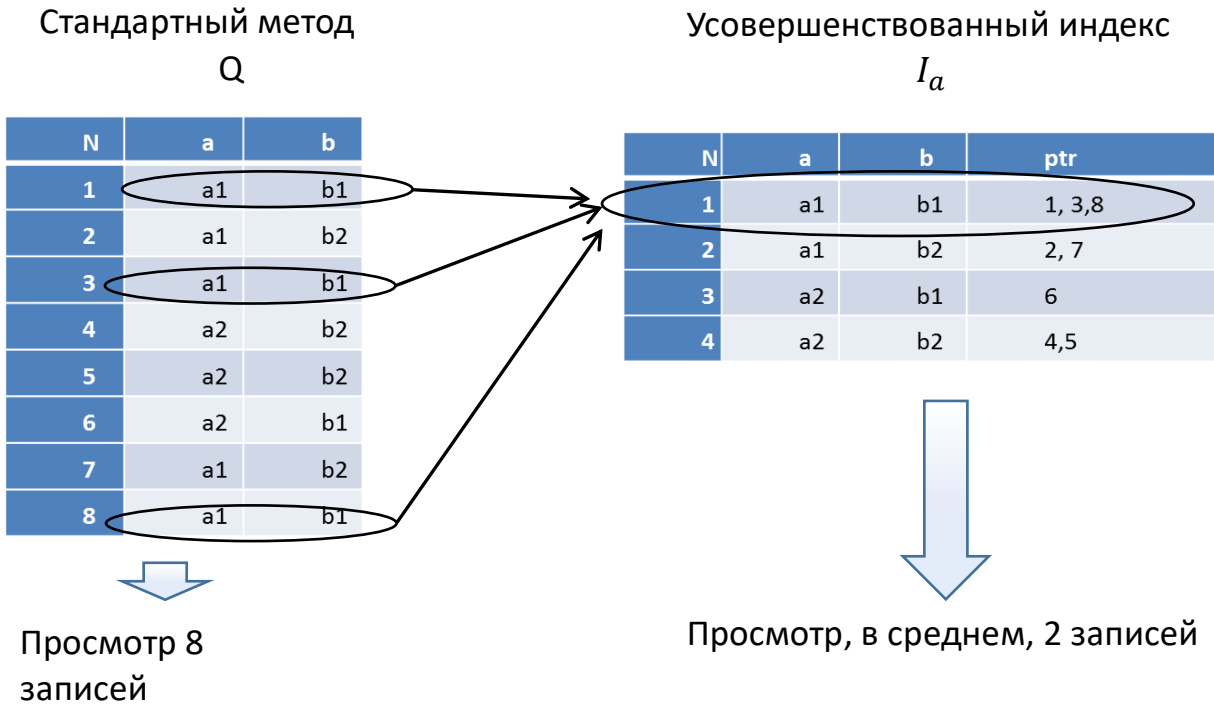


Схема. Стандартный метод выборки и усовершенствованный индекс.

Использование усовершенствованного индекса I_a уже дает значительный эффект. На отношении Q прирост производительности, в среднем, в четыре раза.

На количество записей в индексе I_a влияет не количество записей в исходном отношении, а количество значений дескрипторов. Таким образом, рост количества записей в исходном отношении на время поиска практически не влияет.

Пусть count(I) –количество записей в усовершенствованном индексе I_a.

d_i – количество значений i-го дескриптора

n – Количество дескрипторов.

Тогда по правилам комбинаторики, и с учетом того, что в исходном множестве могут находиться не все возможные комбинации дескрипторов, имеем:

$$count(I) \leq \prod_{i=1}^{n}\left(count(d_i)\right) \qquad (2.1.)$$

Таким образом, при условии, что количество значений дескриптора значительно меньше, чем количество записей в исходном отношении Q, выполняется неравенство count(Q) >> count (I), где count (Q) – количество записей в исходном отношении Q.

Является очевидны, что предложенная модель и алгоритм отображают семантику данных.

Таким образом, проанализированы концептуальные позиции структур данных АСУ вуза и АСУ мебельного производства. Предложены алгоритмы организации и доступа к данным, к их семантике и позволяющие повысить эффективность обработки данных и, как следствие, повысить эффективность функционирования АСУ в целом. Теоретические выводы экспериментально подтверждены.

4. Программное обеспечение Web-сайта

В данном разделе рассмотр им программное обеспечение Web-сайта только в части новых компонент и прежде всего асинхронные коммуникации

Программа основана на архитектуре «Клиент-сервер» с использованием ASP.NET и .NET Framework для динамического формирования исполняемого кода по запросам пользователя. Приводится только один (из 760) законченный код исполняемой программы, поддерживающей семантику данных (пример одной траектории партфолио).

Программа имеет следующий исходный код:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

```
namespace KafedraUGLTU.Domain.Entities
{
  public  class UchebaStudenta
    {
        public int     UchebaStudentaID { get; set; }

         [Display(Name = "Студент")]
        public int StudentID { get; set; }

         [Display(Name = "Семестр")]
         public string Semestr { get; set; }

        [Display(Name = "Дисциплина")]
         public int DisziplinaID { get; set; }

        [UIHint("YesNo")]
        public int _0YesNoID { get; set; }

       [Display(Name = "Курсовая работа оценка")]
        public string KursRabOzenka { get; set; }

        [Display(Name = "Контрольная работа ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo1 { get; set; }

        [Display(Name = "Контрольная работа оценка")]
        public string KontRabOzenka { get; set; }

        [Display(Name = "Лабораторно-практические занятия ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo2 { get; set; }
        [Display(Name = "Число лабораторно-практических занятий")]
        public string ChisloLabPrakt { get; set; }
        [Display(Name = "Лабораторно-практические занятия оценка")]
        public string LabPraktOzenka { get; set; }

        [Display(Name = "Самостоятельная  работа ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo3 { get; set; }
        [Display(Name = "Тема самостоятельной работы")]
        public string TemiSamRab { get; set; }
        [Display(Name = "Самостоятельная работа оценка")]
        public string SamRabOzenka { get; set; }

        [Display(Name = "Зачет ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo4 { get; set; }
        [Display(Name = "Оценка ")]
        public string ZachetOzenka { get; set; }

        [Display(Name = "Экзамен ДА - НЕТ")]
        [UIHint("YesNo")]
        public string YesNo5 { get; set; }
        [Display(Name = "Оценка ")]
        public string EkzamentOzenka { get; set; }
        [Display(Name = "Зачетных единиц ")]
        public string ZachetEdiniz { get; set; }


        [Display(Name = "Ссылка на работу и справку о плагиате")]
        public string UchebaPole01 { get; set; }
        [Display(Name = "Ссылка на протокол защиты")]
        public string UchebaPole02 { get; set; }
```

```
        [Display(Name = "Приобретённые компетенции, записываются через точку с запятой
без пробелов")]
        public string UchebaPole03 { get; set; }
        [Display(Name = "Доп. поле 04")]
        public string UchebaPole04 { get; set; }


        public virtual Student Student { get; set; }

        public virtual Disziplina Disziplina { get; set; }
        public virtual _0YesNo _0YesNo { get; set; }
        public virtual ICollection<UchebaStudenta> UchebaStudentas { get; set; }


    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace KafedraUGLTU.Domain.Entities
{
  public  class VipuskKvalifRabot
    {
      public int VipuskKvalifRabotID { get; set; }

        [DisplayName("Студент")]
        public int StudentID { get; set; }
        [DisplayName("Тема ВК работы")]
        public string TemaVKR { get; set; }
        [DisplayName("Дата - утверждения темы ВКР")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]  //***
отображается только дата, без времени ApplyFormatInEditMode указывает, что данное
форматирование должно применяться также для значений, отображающихся в текстовых строках,
предназначенных для редактирования
        public DateTime NachaloRazVPR { get; set; }
        [DisplayName("Дата - окончание разработки ВКР")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]  //***
отображается только дата, без времени ApplyFormatInEditMode указывает, что данное
форматирование должно применяться также для значений, отображающихся в текстовых строках,
предназначенных для редактирования
        public DateTime OkonchanieRazVKR { get; set; }
        [DisplayName("Дата - предварительная защита на кафедре")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]  //***
отображается только дата, без времени ApplyFormatInEditMode указывает, что данное
форматирование должно применяться также для значений, отображающихся в текстовых строках,
предназначенных для редактирования
        public DateTime PredZasitaVKR { get; set; }
        [DisplayName("Оценка предварительной защиты")]
        public string OzenkaPredZasitaVKR { get; set; }
        [DisplayName("Протокол предварительной защиты")]
        public string ProtokolPredZasitaVKR { get; set; }
        [DisplayName("Заключение о плагиате ВКР")]
        public string SpravkaPlagiat { get; set; }
        [DisplayName("Заключение руководителя ВКР т.е. моё мнение")]
        public string ZaklRukovodVKR { get; set; }
        [DisplayName("Выпускная квалификационная работа")]
        public string FailVKR { get; set; }
        [DisplayName("Дата - защита в ГЭК")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
```

```
        public DateTime ZasitaVKRvGEK { get; set; }
        [DisplayName("Оценка за защиту в ГЭК")]
        public string OzenkaZasitaVKRvGEK { get; set; }
        [DisplayName("Протокол защиты в ГЭК")]
        public string ProtokolZasitiGEK { get; set; }

        [DisplayName ( "Оценка за государственный экзамен" )]
        public string Dop01VKR { get; set; }
        [DisplayName ( "Протокол сдачи ГЭ - имя файла с расширением") ]
        public string Dop02VKR { get; set; }
        [DisplayName ( "Приобретённые компетенции, записываются через точку с запятой без
пробелов") ]
        public string Dop03VKR { get; set; }
        [Display(Name = "Dop04VKR")]
        public string Dop04VKR { get; set; }
        [Display(Name = "Dop05VKR")]
        public string Dop05VKR { get; set; }

        public virtual Student Student { get; set; }
        public virtual ICollection<VipuskKvalifRabot> VipuskKvalifRabots { get; set; }
    }
    }
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using KafedraUGLTU.Domain.Entities;
using KafedraUGLTU.Domain.Concrete;
using PagedList;

namespace KafedraUGLTU.WebUI.Controllers
{
 [Authorize(Users = "admin@Chasovskikh.ru, admin@Chasovskikh1.ru, admin@Chasovskikh2.ru
,admin@Chasovskikh3.ru, admin@Chasovskikh4.ru,admin@Achurina.ru, admin@Achurina1.ru,
admin@Bogoslovskay.ru ,admin@Bogoslovskay1.ru,
admin@Azarenok.ru,admin@Azarenok1.ru,admin@Bessonov.ru, admin@Bessonov1.ru, admin@Spak.ru
,admin@Spak1.ru, admin@Sapegina.ru,admin@Sapegina1.ru, admin@Malutina.ru,
admin@Malutina1.ru ,admin@Sepetkin.ru,
admin@Sepetkin1.ru,admin@Pomitkina.ru,admin@Pomitkina1.ru,admin@Kokosa.ru,
admin@Kokosa1.ru, admin@Komarova.ru ,admin@Komarova1.ru,
admin@Voronov.ru,admin@Voronov1.ru, admin@Usolzev.ru, admin@Usolzev1.ru ,admin@Butko.ru,
admin@Butko1.ru,admin@AdminChas.ru,admin@AdminChas1.ru")]

    public class AdminUchebaStudentasController : Controller
    {
        private EFDbContext db = new EFDbContext();

        // GET: /AdminUchebaStudentas/
//        public ActionResult Index()
//        {
//            var uchebastudentas = db.UchebaStudentas.Include(u => u._0YesNo).Include(u
=> u.Disziplina).Include(u => u.Student);
//            return View(uchebastudentas.ToList());
//        }

        public ViewResult Index(string sortOrder, string currentFilter, string
searchString, int? page)
        {
```

```
            var uchebaStudentas = db.UchebaStudentas.Include(u => u._0YesNo).Include(u =>
u.Disziplina).Include(u => u.Student);

            ViewBag.CurrentSort = sortOrder;


            ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";

            ViewBag.DateSortParm = sortOrder == "Date" ? "date_desc" : "Date";
            ViewBag.Date1SortParm = sortOrder == "Date1" ? "date1_desc" : "Date1";
            ViewBag.Date2SortParm = sortOrder == "Date2" ? "date2_desc" : "Date2";



            if (searchString != null)
            {
                page = 1;
            }
            else { searchString = currentFilter; }
            ViewBag.CurrentFilter = searchString;


            var students = from s in db.UchebaStudentas.Include(u => u._0YesNo).Include(u
=> u.Disziplina).Include(u => u.Student)
                           //** добавил  .OrderBy(p => p.Disziplina.Nazvanie)

                           select s;
            if (!String.IsNullOrEmpty(searchString))
            {
                students = students.Where(s =>
s.Student.Familiy.ToString().Contains(searchString.ToUpper())
                    ||
s.Student._0ObrazovatelnayProgramma.SifrObProg.ToString().Contains(searchString.ToUpper()
)
                    ||
s.Student._0ObrazovatelnayProgramma.NapravlenieI.SifrNapravleniy.ToString().Contains(sear
chString.ToUpper())
                    || s.Student.Gruppa.ToString().Contains(searchString.ToUpper()));
            }
            switch (sortOrder)
            {
                case "name_desc":
                    students = students.OrderByDescending(s => s.Student.Familiy);
                    break;
                case "Date":
                    students = students.OrderBy(s =>
s.Student._0ObrazovatelnayProgramma.SifrObProg);
                    break;
                case "date_desc":
                    students = students.OrderByDescending(s =>
s.Student._0ObrazovatelnayProgramma.SifrObProg);
                    break;

                case "Date1":
                    students = students.OrderBy(s =>
s.Student._0ObrazovatelnayProgramma.NapravlenieI.SifrNapravleniy);
                    break;
                case "date1_desc":
                    students = students.OrderByDescending(s =>
s.Student._0ObrazovatelnayProgramma.NapravlenieI.SifrNapravleniy);
                    break;

                case "Date2":
                    students = students.OrderBy(s => s.Student.Gruppa);
```

```
                    break;
                case "date2_desc":
                    students = students.OrderByDescending(s => s.Student.Gruppa);
                    break;

                default:
                    students = students.OrderBy(s => s.Student.Familiy);
                    break;
            }


            int pageSize = 10000;
            int pageNumber = (page ?? 1);
            return View(students.ToPagedList(pageNumber, pageSize));
        }


        //**************************************************************************   INDEX1

        public ActionResult Index1(string sortOrder, string currentFilter, string
searchString, int? page, int? id, int? id1)
        {


            ViewBag.Id = id;
            ViewBag.Id1 = id1;
            if (id1 == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            UchebaStudenta uchebaStudenta = db.UchebaStudentas.Find(id1);
            if (uchebaStudenta == null)
            {
                return HttpNotFound();
            }

            var uchebaStudentas = db.UchebaStudentas.Include(u => u._0YesNo).Include(u =>
u.Disziplina).Include(u => u.Student).Include(u => u.EkzamentOzenka).Include(u =>
                u.Student._0ObrazovatelnayProgramma.SifrObProg).Include(u =>
u.Student._0ObrazovatelnayProgramma.NapravlenieI.SifrNapravleniy);

            ViewBag.CurrentSort = sortOrder;


            ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";

            ViewBag.DateSortParm = sortOrder == "Date" ? "date_desc" : "Date";
            ViewBag.Date1SortParm = sortOrder == "Date1" ? "date1_desc" : "Date1";
            ViewBag.Date2SortParm = sortOrder == "Date2" ? "date2_desc" : "Date2";



            if (searchString != null)
            {
                page = 1;
            }
            else { searchString = currentFilter; }
            ViewBag.CurrentFilter = searchString;


            var students = from s in db.UchebaStudentas.Include(u => u._0YesNo).Include(u
=> u.Disziplina).Include(u => u.Student)
                           //*                          .Where(s => s.UchebaStudentaID ==
id)
```

```
                            select s;


            if (!String.IsNullOrEmpty(searchString))
            {
                students = students.Where(s =>
s.Disziplina.Nazvanie.ToUpper().Contains(searchString.ToUpper())

                       || s.EkzamentOzenka.ToUpper().Contains(searchString.ToUpper()));

            }
            switch (sortOrder)
            {
                case "name_desc":
                    students = students.OrderByDescending(s => s.Disziplina.Nazvanie);
                    break;
                case "Date":
                    students = students.OrderBy(s => s.EkzamentOzenka);
                    break;
                case "date_desc":
                    students = students.OrderByDescending(s => s.EkzamentOzenka);
                    break;


                default:
                    students = students.OrderBy(s => s.Disziplina.Nazvanie);
                    break;
            }


            int pageSize = 10000;
            int pageNumber = (page ?? 1);
            return View(students.ToPagedList(pageNumber, pageSize));
        }


        //*************************************************************



        // GET: /AdminUchebaStudentas/Create
        public ActionResult Create()
        {
            ViewBag._0YesNoID = new SelectList(db._0YesNo, "_0YesNoID", "YesNo");
            var nazvanie = db.Disziplinas
                .OrderBy(p => p.Nazvanie);
            ViewBag.DisziplinaID = new SelectList(nazvanie, "DisziplinaID", "Nazvanie");
            var familiya = db.Students
                 .OrderBy(p => p.Familiy);
            ViewBag.StudentID = new SelectList(familiya, "StudentID", "Familiy");
            return View();
        }

        // POST: /AdminUchebaStudentas/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в статье http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult
Create([Bind(Include="UchebaStudentaID,StudentID,Semestr,DisziplinaID,_0YesNoID,KursRabOz
enka,YesNo1,KontRabOzenka,YesNo2,ChisloLabPrakt,LabPraktOzenka,YesNo3,TemiSamRab,SamRabOz
enka,YesNo4,ZachetOzenka,YesNo5,EkzamentOzenka,ZachetEdiniz,UchebaPole01,UchebaPole02,Uch
ebaPole03,UchebaPole04")] UchebaStudenta uchebastudenta)
```

```
        {
            if (ModelState.IsValid)
            {
                db.UchebaStudentas.Add(uchebastudenta);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag._0YesNoID = new SelectList(db._0YesNo, "_0YesNoID", "YesNo",
uchebastudenta._0YesNoID);
            var nazvanie = db.Disziplinas
                .OrderBy(p => p.Nazvanie);
            ViewBag.DisziplinaID = new SelectList(nazvanie, "DisziplinaID", "Nazvanie",
uchebastudenta.DisziplinaID);
            var familiya = db.Students
                 .OrderBy(p => p.Familiy);
            ViewBag.StudentID = new SelectList(familiya, "StudentID", "Familiy",
uchebastudenta.StudentID);
            return View(uchebastudenta);
        }

        // GET: /AdminUchebaStudentas/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            UchebaStudenta uchebastudenta = db.UchebaStudentas.Find(id);
            if (uchebastudenta == null)
            {
                return HttpNotFound();
            }
            ViewBag._0YesNoID = new SelectList(db._0YesNo, "_0YesNoID", "YesNo",
uchebastudenta._0YesNoID);
            var nazvanie = db.Disziplinas
                .OrderBy(n => n.Nazvanie);
            ViewBag.DisziplinaID = new SelectList(nazvanie, "DisziplinaID", "Nazvanie",
uchebastudenta.DisziplinaID);
            var familiay = db.Students
               .OrderBy(n => n.Familiy);
            ViewBag.StudentID = new SelectList(familiay, "StudentID", "Familiy",
uchebastudenta.StudentID);
            return View(uchebastudenta);
        }

        // POST: /AdminUchebaStudentas/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в статье http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult
Edit([Bind(Include="UchebaStudentaID,StudentID,Semestr,DisziplinaID,_0YesNoID,KursRabOzen
ka,YesNo1,KontRabOzenka,YesNo2,ChisloLabPrakt,LabPraktOzenka,YesNo3,TemiSamRab,SamRabOzen
ka,YesNo4,ZachetOzenka,YesNo5,EkzamentOzenka,ZachetEdiniz,UchebaPole01,UchebaPole02,Ucheb
aPole03,UchebaPole04")] UchebaStudenta uchebastudenta)
        {
            if (ModelState.IsValid)
            {
                db.Entry(uchebastudenta).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
```

```
            ViewBag._0YesNoID = new SelectList(db._0YesNo, "_0YesNoID", "YesNo",
uchebastudenta._0YesNoID);

            var nazvanie = db.Disziplinas
               .OrderBy(n => n.Nazvanie);
            ViewBag.DisziplinaID = new SelectList(nazvanie, "DisziplinaID", "Nazvanie",
uchebastudenta.DisziplinaID);
            var familiay = db.Students
               .OrderBy(n => n.Familiy);
            ViewBag.StudentID = new SelectList(familiay, "StudentID", "Familiy",
uchebastudenta.StudentID);
            return View(uchebastudenta);
        }

        // GET: /AdminUchebaStudentas/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            UchebaStudenta uchebastudenta = db.UchebaStudentas.Find(id);
            if (uchebastudenta == null)
            {
                return HttpNotFound();
            }
            return View(uchebastudenta);
        }

        // POST: /AdminUchebaStudentas/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            UchebaStudenta uchebastudenta = db.UchebaStudentas.Find(id);
            db.UchebaStudentas.Remove(uchebastudenta);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
    }
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using KafedraUGLTU.Domain.Concrete;
using KafedraUGLTU.Domain.Entities;
using PagedList;

namespace KafedraUGLTU.WebUI.Controllers
{
```

```
    [Authorize(Users = "admin@Chasovskikh.ru, admin@Chasovskikh1.ru,
admin@Chasovskikh2.ru ,admin@Chasovskikh3.ru, admin@Chasovskikh4.ru,admin@Achurina.ru,
admin@Achurina1.ru, admin@Bogoslovskay.ru ,admin@Bogoslovskay1.ru,
admin@Azarenok.ru,admin@Azarenok1.ru,admin@Bessonov.ru, admin@Bessonov1.ru, admin@Spak.ru
,admin@Spak1.ru, admin@Sapegina.ru,admin@Sapegina1.ru, admin@Malutina.ru,
admin@Malutina1.ru ,admin@Sepetkin.ru,
admin@Sepetkin1.ru,admin@Pomitkina.ru,admin@Pomitkina1.ru,admin@Kokosa.ru,
admin@Kokosa1.ru, admin@Komarova.ru ,admin@Komarova1.ru,
admin@Voronov.ru,admin@Voronov1.ru, admin@Usolzev.ru, admin@Usolzev1.ru ,admin@Butko.ru,
admin@Butko1.ru,admin@AdminChas.ru,admin@AdminChas1.ru")]

    public class AdminVipuskKvalifRabotsController : Controller
    {
        private EFDbContext db = new EFDbContext();

        // GET: AdminVipuskKvalifRabots
//        public ActionResult Index()
//        {
//            var vipuskKvalifRabots = db.VipuskKvalifRabots.Include(v => v.Student);
//            return View(vipuskKvalifRabots.ToList());
//        }

        public ViewResult Index(string sortOrder, string currentFilter, string
searchString, int? page)
        {

            var vipuskKvalifRabots = db.VipuskKvalifRabots.Include(v => v.Student);

            ViewBag.CurrentSort = sortOrder;
            ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
            ViewBag.DateSortParm = sortOrder == "Date" ? "date_desc" : "Date";
            if (searchString != null)
            {
                page = 1;
            }
            else { searchString = currentFilter; }
            ViewBag.CurrentFilter = searchString;


            var students = from s in db.VipuskKvalifRabots.Include(v => v.Student)
                           select s;
            if (!String.IsNullOrEmpty(searchString))
            {
                students = students.Where(s =>
s.Student.Familiy.ToString().Contains(searchString.ToUpper())
                    || s.TemaVKR.ToString().Contains(searchString.ToUpper()));
            }
            switch (sortOrder)
            {
                case "name_desc":
                    students = students.OrderByDescending(s => s.Student.Familiy);
                    break;
                case "Date":
                    students = students.OrderBy(s => s.TemaVKR);
                    break;
                case "date_desc":
                    students = students.OrderByDescending(s => s.TemaVKR);
                    break;
                default:
                    students = students.OrderBy(s => s.Student.Familiy);
                    break;
            }
```

```
            int pageSize = 10;
            int pageNumber = (page ?? 1);
            return View(students.ToPagedList(pageNumber, pageSize));


        }


        // GET: AdminVipuskKvalifRabots/Create
        public ActionResult Create()
        {
            var familiya = db.Students
                .OrderBy(p => p.Familiy);
            ViewBag.StudentID = new SelectList(familiya, "StudentID", "Familiy");
            return View();
        }

        // POST: AdminVipuskKvalifRabots/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
        // сведения см. в статье http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"VipuskKvalifRabotID,StudentID,TemaVKR,NachaloRazVPR,OkonchanieRazVKR,PredZasitaVKR,Ozenk
aPredZasitaVKR,ProtokolPredZasitaVKR,SpravkaPlagiat,ZaklRukovodVKR,FailVKR,ZasitaVKRvGEK,
OzenkaZasitaVKRvGEK,ProtokolZasitiGEK,Dop01VKR,Dop02VKR,Dop03VKR,Dop04VKR,Dop05VKR")]
VipuskKvalifRabot vipuskKvalifRabot)
        {
            if (ModelState.IsValid)
            {
                db.VipuskKvalifRabots.Add(vipuskKvalifRabot);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            var familiya = db.Students
                .OrderBy(p => p.Familiy);
            ViewBag.StudentID = new SelectList(familiya, "StudentID", "Familiy",
vipuskKvalifRabot.StudentID);
            return View(vipuskKvalifRabot);
        }

        // GET: AdminVipuskKvalifRabots/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            VipuskKvalifRabot vipuskKvalifRabot = db.VipuskKvalifRabots.Find(id);
            if (vipuskKvalifRabot == null)
            {
                return HttpNotFound();
            }
            var familiya = db.Students
                .OrderBy(p => p.Familiy);
            ViewBag.StudentID = new SelectList(familiya, "StudentID", "Familiy",
vipuskKvalifRabot.StudentID);
            return View(vipuskKvalifRabot);
        }

        // POST: AdminVipuskKvalifRabots/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные
свойства, для которых следует установить привязку. Дополнительные
```

```
        // сведения см. в статье http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"VipuskKvalifRabotID,StudentID,TemaVKR,NachaloRazVPR,OkonchanieRazVKR,PredZasitaVKR,Ozenk
aPredZasitaVKR,ProtokolPredZasitaVKR,SpravkaPlagiat,ZaklRukovodVKR,FailVKR,ZasitaVKRvGEK,
OzenkaZasitaVKRvGEK,ProtokolZasitiGEK,Dop01VKR,Dop02VKR,Dop03VKR,Dop04VKR,Dop05VKR")]
VipuskKvalifRabot vipuskKvalifRabot)
        {
            if (ModelState.IsValid)
            {
                db.Entry(vipuskKvalifRabot).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            var familiya = db.Students
                            .OrderBy(p => p.Familiy);
            ViewBag.StudentID = new SelectList(familiya, "StudentID", "Familiy",
vipuskKvalifRabot.StudentID);
            return View(vipuskKvalifRabot);
        }

        // GET: AdminVipuskKvalifRabots/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            VipuskKvalifRabot vipuskKvalifRabot = db.VipuskKvalifRabots.Find(id);
            if (vipuskKvalifRabot == null)
            {
                return HttpNotFound();
            }
            return View(vipuskKvalifRabot);
        }

        // POST: AdminVipuskKvalifRabots/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            VipuskKvalifRabot vipuskKvalifRabot = db.VipuskKvalifRabots.Find(id);
            db.VipuskKvalifRabots.Remove(vipuskKvalifRabot);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
@model PagedList.IPagedList<KafedraUGLTU.Domain.Entities.UchebaStudenta>
@using PagedList.Mvc;
<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />


@{
```

```
    ViewBag.Title = "Учеба студента";
    Layout = "~/Views/Shared/_LayoutAdminOsnov.cshtml";
}

<title>Разработка профессора УГЛТУ Часовских В. П. Используется плагин FooTable - jQuery</title>
<meta name="description" content="Сайт   полностью отвечающий требованиям нового закона об образование" />
<meta name="keywords" content="Менеджмент, УГЛТУ , наука, МиВЭДП , институт экономики и управления, студенту, преподаватели, рабочие программы, патенты,УГЛТУ" />
<meta name="author" content="Часовских В.П.  Chasovskikh V." />
<meta name="viewport" content="width = device-width, initial-scale = 1.0, minimum-scale = 1.0, maximum-scale = 1.0, user-scalable = no" /><!--
**********************************************************-->
<br /><br />
<div style="margin-top:0em;">
    <h2 style="font-size:1.4em; color:#a1845a;   text-align:center;">

        АДМИНИСТРИРОВАНИЕ - учеба обучающегося <br /><br />


    </h2>
    В таблице   если "кликнуть" по названию  столбца "Обучающийся"  (выделено синим цветом), то будет выполняться
    сортировка по убыванию или возрастанию значений (чередуются). <br />
    Можно выполнять поиск записей по фамилии обучающегося, номеру студенческого билета,названию дисциплины, группе студента,
    оценке за экзамен (разрешается указывать от <a style="color:red">ДВУХ</a> символов.<br />
    "Кликнув" по + перед фамилией увидите полную информацию об учебе обучающегося.
</div>




<link href="~/FooTable-master/css/footable-0.1.css" rel="stylesheet" />


<script src="~/Scripts/jquery-1.10.2.js"></script>
<script src="js/footable.js" type="text/javascript"></script>



<script type="text/javascript">
    $(function () {
        $('table').footable();
    });
</script>

<p class="buttonGorMenu2014 greenGorMenu2014 " style="font-size:0.95em; margin-right:auto; margin-left:auto; display:block; width:37%;  ">
    @Html.ActionLink("Ввести новую запись", "Create", new { area = "" }, new { style = "color:white; font-size:1em;" })
</p><br /><br />

<div style="float:right;">
    <p class="buttonGorMenu2014 greenGorMenu2014">
        <a href="~/RabotiStudentov-master/index.html" style="color:white; font-size:0.8em;" target="_blank">Файлы на сервере</a>
    <p>Работы и протоколы размещаются в папке <a class="text-danger" style="font-size:1.5em;">
        RabotiStudentov</a> в соответствующей папке. Например: /Prepod01/OchayPrepod01/Disziplina1/Kurs/Test.docx
```

```
</div>



@{
    int p1 = 0; int p2 = 0; string t1 = "123456789";




    using (Html.BeginForm("Index", "AdminUchebaStudentas", FormMethod.Get))
    { <p>
        Поиск по   фамилии обучающегося или дисциплине: @Html.TextBox("SearchString")
        <input type="submit" value="Найти" style="border-radius:5px; background-
color:green;color:white;text-align:center" />
    </p> }

    foreach (var item in Model)
    {
        int nom = Convert.ToInt32(item.StudentID);
        if ((nom == 00000) & (item.Student.Familiy == "Учебный год"))
        { <h4 style="font-size:1.4em; color:#a1845a;   text-
align:center;">@item.Student.Imy </h4>
        }
    }

    <table class="footable" style="width:100%; margin: 0 auto;">
        <thead>
            <tr>
                <th data-hide="expand" style="text-align:center;">
                    № пп
                </th>

                <th data-hide="phone" style="text-align:center;">
                    Дисциплины образовательной программы
                </th>
                <th data-class="expand" style="text-align:center;">
                    @Html.ActionLink("Обучающейся", "Index", new { sortOrder =
ViewBag.NameSortParm })
                </th>
                <th data-hide="phone" style="text-align:center;">
                     Группа
                </th>
                <th data-hide="phone" style="text-align:center;">
                     Шифр образовательной программы
                </th>

                <th data-hide="all" style="text-align:center;">
                     Название образовательной программы
                </th>
                <th data-hide="phone" style="text-align:center;">
                   Действие
                </th>

            </tr>
        </thead>
        @foreach (var item in Model)
        {
            if (@item.Student.Familiy != "Учебный год")
            {
```

```
                if (t1 != item.Student.Familiy)
                {
                    t1 = @item.Student.Familiy.ToString();
                    p1 = p1 + 1;
                    <tr>
                        <td style="text-align:center">@p1.</td>

                        <td>

                            @Html.ActionLink("Посмотреть", "Index1", new { id =
item.StudentID, id1 = item.UchebaStudentaID })


                        </td>

                        <td>
                            @Html.ActionLink(@item.Student.Familiy, "Edit", new { id =
item.UchebaStudentaID })

                            @Html.DisplayFor(modelItem => item.Student.Imy)
                            @Html.DisplayFor(modelItem => item.Student.Otchestvo)

                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.Student.Gruppa)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem =>
item.Student._0ObrazovatelnayProgramma.SifrObProg)
                        </td>

                        <td>
                            @Html.DisplayFor(modelItem =>
item.Student._0ObrazovatelnayProgramma.NapravlenieI.SifrNapravleniy)
                        </td>
                        <td>

                            @Html.ActionLink("Удалить", "Delete", new { id =
item.UchebaStudentaID, @class = "btn btn-primary" })
                        </td>
                    </tr>
                }
            }
        }
    </table>
}
<br /> Страница @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) из
@Model.PageCount
@Html.PagedListPager(Model, page => Url.Action("Index", "AdminUchebaStudentas", new {
page, sortOrder = ViewBag.CurrentSort, currentFilter = ViewBag.CurrentFilter }))
@model KafedraUGLTU.Domain.Entities.UchebaStudenta

@{
    ViewBag.Title = "Ввод";
    Layout = "~/Views/Shared/_LayoutAdminOsnov.cshtml";
}

<title>Разработка профессора Часовских В.П. Используется плагин FooTable - jQuery
</title>
<meta name="viewport" content="width = device-width, initial-scale = 1.0, minimum-scale =
1.0, maximum-scale = 1.0, user-scalable = no" />
<!-- ************************************************   -->
<!--**********************************************************-->
```

```
<br /><br />
<div style="margin-top:0em;">
    <h2 style="font-size:1.4em; color:#a1845a;   text-align:center;">

        АДМИНИСТРИРОВАНИЕ - ввод записи учеба обучающегося <br /><br />

    </h2>

</div>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.StudentID, "Обучающейся", htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("StudentID", null, htmlAttributes: new { @class =
"form-control" })

                @Html.ValidationMessageFor(model => model.StudentID, "", new { @class =
"text-danger" })
            </div>

        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Semestr, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Semestr, new { htmlAttributes = new {
@class = "form-control" } })

                @Html.ValidationMessageFor(model => model.Semestr, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DisziplinaID, "Дисциплина", htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("DisziplinaID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.DisziplinaID, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model._0YesNoID, "Курсовая работа ДА - НЕТ",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("_0YesNoID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model._0YesNoID, "", new { @class =
"text-danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.UchebaPole01, "Ссылка на работу и справку о
плагиате", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UchebaPole01, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UchebaPole01, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UchebaPole02, "Ссылка на протокол защиты",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UchebaPole02, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UchebaPole02, "", new { @class
= "text-danger" })
            </div>
        </div>


        <div class="form-group">
            @Html.LabelFor(model => model.KursRabOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.KursRabOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.KursRabOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo1, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo1, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo1, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.KontRabOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.KontRabOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.KontRabOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo2, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo2, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo2, "", new { @class =
"text-danger" })
```

```
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ChisloLabPrakt, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ChisloLabPrakt, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ChisloLabPrakt, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.LabPraktOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.LabPraktOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.LabPraktOzenka, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo3, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo3, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo3, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TemiSamRab, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TemiSamRab, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TemiSamRab, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SamRabOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.SamRabOzenka, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.SamRabOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo4, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo4, new { htmlAttributes = new {
@class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.YesNo4, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZachetOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZachetOzenka, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZachetOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo5, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo5, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo5, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.EkzamentOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EkzamentOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.EkzamentOzenka, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZachetEdiniz, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZachetEdiniz, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZachetEdiniz, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UchebaPole03, "Приобретённые компетенции,
записываются через точку с запятой без пробелов", htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UchebaPole03, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UchebaPole03, "", new { @class
= "text-danger" })
            </div>
        </div>



        <div class="panel-footer">
```

```
            <input type="submit" value="Сохранить в БД" class="btn btn-primary" />
            @Html.ActionLink("Отказаться и вернуться обратно", "Index", null, new
{
    @class = "btn btn-default"
})
        </div>
    </div>
}

@model KafedraUGLTU.Domain.Entities.UchebaStudenta

@{
    ViewBag.Title = "Редактировать";
    Layout = "~/Views/Shared/_LayoutAdminOsnov.cshtml";
}

<title>Разработка профессора Часовских В.П. Используется плагин FooTable - jQuery
</title>
<meta name="viewport" content="width = device-width, initial-scale = 1.0, minimum-scale =
1.0, maximum-scale = 1.0, user-scalable = no" />
<!-- ******************************************************   -->
<!--**********************************************************-->
<br /><br />
<div style="margin-top:0em;">
    <h2 style="font-size:1.4em; color:#a1845a;   text-align:center;">

        АДМИНИСТРИРОВАНИЕ - редактирование записи учеба обучающегося <br /><br />

    </h2>

</div>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.UchebaStudentaID)

        <div class="form-group">
            @Html.LabelFor(model => model.StudentID, "Фамилия обучающегося",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("StudentID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.StudentID, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Semestr, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Semestr, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Semestr, "", new { @class =
"text-danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.DisziplinaID, "Дисциплина", htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("DisziplinaID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.DisziplinaID, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model._0YesNoID, "Курсовая работа", htmlAttributes:
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("_0YesNoID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model._0YesNoID, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UchebaPole01, "Ссылка на работу и справку о
плагиате", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UchebaPole01, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UchebaPole01, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.UchebaPole02, "Ссылка на протокол защиты",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UchebaPole02, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UchebaPole02, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.KursRabOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.KursRabOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.KursRabOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo1, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo1, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo1, "", new { @class =
"text-danger" })
            </div>
```

```
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.KontRabOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.KontRabOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.KontRabOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo2, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo2, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo2, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ChisloLabPrakt, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ChisloLabPrakt, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ChisloLabPrakt, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.LabPraktOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.LabPraktOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.LabPraktOzenka, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo3, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo3, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo3, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TemiSamRab, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TemiSamRab, new { htmlAttributes = new {
@class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.TemiSamRab, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SamRabOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.SamRabOzenka, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.SamRabOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo4, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo4, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo4, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZachetOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZachetOzenka, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZachetOzenka, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.YesNo5, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.YesNo5, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.YesNo5, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.EkzamentOzenka, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EkzamentOzenka, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.EkzamentOzenka, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZachetEdiniz, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.ZachetEdiniz, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZachetEdiniz, "", new { @class
= "text-danger" })
            </div>
        </div>



        <div class="form-group">
            @Html.LabelFor(model => model.UchebaPole03, "Приобретённые компетенции,
записываются через точку с запятой без пробелов", htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UchebaPole03, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.UchebaPole03, "", new { @class
= "text-danger" })
            </div>
        </div>




        <div class="panel-footer">
            <input type="submit" value="Сохранить в БД" class="btn btn-primary" />
            @Html.ActionLink("Отказаться и вернуться обратно", "Index", null, new
{
    @class = "btn btn-default"
})
        </div>
    </div>
}

 @model PagedList.IPagedList<KafedraUGLTU.Domain.Entities.VipuskKvalifRabot>
@using PagedList.Mvc;
<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />

@{
    ViewBag.Title = "ВКР";
    Layout = "~/Views/Shared/_LayoutAdminOsnov.cshtml";
}


<br /><br />

<title>Разработка профессора УГЛТУ Часовских В. П. Используется плагин FooTable -
jQuery</title>
<meta name="description" content="Сайт   полностью отвечающий требованиям нового закона
об образование" />
<meta name="keywords" content="Менеджмент, УГЛТУ , наука, МиВЭДП , институт экономики и
управления, студенту, преподаватели, рабочие программы, патенты,УГЛТУ" />
<meta name="author" content="Часовских В.П.  Chasovskikh V." />
<meta name="viewport" content="width = device-width, initial-scale = 1.0, minimum-scale =
1.0, maximum-scale = 1.0, user-scalable = no" />
<!--*****************************************************-->
<br /><br />
<div style="margin-top:-4em;">
    <h2 style="font-size:1.4em; color:#a1845a;   text-align:center;">

        АДМИНИСТРИРОВАНИЕ - выпускная квалификационная работа <br /><br />
```

```
    </h2>
    <h6 style="text-align:center;color:#a1845a;">
        В таблице   если "кликнуть" по названию в столбце "Фамилия Студента" (выделено
синим цветом), то будет выполняться
        сортировка по убыванию или возрастанию значений (чередуются). <br />
        Можно выполнять поиск записей по фамилии студента и названию ВКР (разрешается
указывать от <a style="color:red">ДВУХ</a>  символов.
    </h6>
</div>




<link href="~/FooTable-master/css/footable-0.1.css" rel="stylesheet" />


<script src="~/Scripts/jquery-1.10.2.js"></script>
<script src="js/footable.js" type="text/javascript"></script>



<script type="text/javascript">
    $(function () {
        $('table').footable();
    });
</script>

<p class="buttonGorMenu2014 greenGorMenu2014 " style="font-size:0.95em; margin-
right:auto; margin-left:auto; display:block; width:37%;  ">
    @Html.ActionLink("Ввести новую запись", "Create", new { area = "" }, new { style =
"color:white; font-size:1em;" })
</p><br /><br />


<div style="float:right;">
    <p class="buttonGorMenu2014 greenGorMenu2014">
        <a href="~/Filemanager-master/index.html" style="color:white; font-size:0.8em;"
target="_blank">Файлы на сервере</a>
    <p>ВКР размещаются в папке <a class="text-danger" style="font-size:1.5em;">BKR</a>

</div>



@using (Html.BeginForm("Index", "AdminVipuskKvalifRabots", FormMethod.Get))
{ <p>
        Поиск по теме ВКР и фамилии студента: @Html.TextBox("SearchString",
ViewBag.CurrentFilter as string)
        <input type="submit" value="Найти" style="border-radius:5px; background-
color:green;color:white" />
    </p> }
<br />




    <table class="footable" style="width:100%; margin: 0 auto;">
        <thead>
            <tr>
                <th data-class="expand" style="text-align:center; ">
```

```
                    @Html.ActionLink("Студент", "Index", new { sortOrder =
ViewBag.NameSortParm })
                </th>
                <th   style="text-align:center;">
                   Тема ВКР
                </th>
                <th data-hide="all" style="text-align:center;">
                    Дата получения задания на ВКР
                </th>
                <th data-hide="all" style="text-align:center;">
                    Дата окончания задания на ВКР
                </th>
                <th data-hide="all" style="text-align:center;">
                    Дата предзащиты на кафедре
                </th>
                <th data-hide="all" style="text-align:center;">
                    Оценка предзащиты
                </th>
                <th data-hide="all" style="text-align:center;">
                    Протокол предзащиты
                </th>
                <th data-hide="phone" style="text-align:center;">
                    Справка о плагиате в ВКР
                </th>
                <th data-hide="all" style="text-align:center;">
                    Заключение руководителя ВКР
                </th>
                <th data-hide="all" style="text-align:center;">
                    ВКР
                </th>
                <th data-hide="phone" style="text-align:center;">
                    Дата защиты ВКР в ГЭК
                </th>
                <th data-hide="all" style="text-align:center;">
                    Оценка за защиту ВКР в ГЭК
                </th>
                <th data-hide="all" style="text-align:center;">
                    Протокол защиты ВКР в ГЭК
                </th>
                <th data-hide="all" style="text-align:center;">
                    Оценка за государственный экзамен
                </th>
                <th data-hide="all" style="text-align:center;">
                    Протокол государственного экзамена
                </th>
                <th data-hide="all" style="text-align:center;">
                    Приобретенные компетенции
                </th>

                <th data-hide="all" style="text-align:center;">
                    Внешняя рецензия на ВКР
                </th>

                <th data-hide="all" style="text-align:center;">Действие</th>
            </tr>
        </thead>
            @foreach (var item in Model)
            {
                <tr>
                    <td>
                        @Html.ActionLink(item.Student.Familiy, "Edit", new { id =
item.VipuskKvalifRabotID })

                        @Html.DisplayFor(modelItem => item.Student.Imy)
```

```
                    @Html.DisplayFor(modelItem => item.Student.Otchestvo)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.TemaVKR)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.NachaloRazVPR)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.OkonchanieRazVKR)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.PredZasitaVKR)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.OzenkaPredZasitaVKR)
                </td>
                <td>
                    @if (@item.ProtokolPredZasitaVKR != null)
                    {
                        <a href="~/Uploads/BKR/@item.ProtokolPredZasitaVKR "
target="_blank">  Посмотреть  </a>
                    }
                    else
                    { <h5 style="color:red; ">Копия протокола  о предзащите не
загружена</h5>}
                </td>
                <td>
                    @if (@item.SpravkaPlagiat != null)
                    {
                        <a href="~/Uploads/BKR/@item.SpravkaPlagiat "
target="_blank">
                            Посмотреть
                        </a>
                    }
                    else
                    { <h5 style="color:red; ">Копия справки о плагиате не
загружена</h5>}


                </td>
                <td>
                    @if (@item.ZaklRukovodVKR != null)
                    {
                        <a href="~/Uploads/BKR/@item.ZaklRukovodVKR "
target="_blank">
                            Посмотреть
                        </a>
                    }
                    else
                    { <h5 style="color:red; ">Копия заключения руководителя ВКР не
загружена</h5>}


                </td>
                <td>
                    @if (@item.FailVKR != null)
                    {
                        <a href="~/Uploads/BKR/@item.FailVKR " target="_blank">
                            Посмотреть

                        </a>
                    }
                    else
```

```
                    { <h5 style="color:red; ">Копия ВКР не загружена</h5>}


                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ZasitaVKRvGEK)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.OzenkaZasitaVKRvGEK)
                </td>
                <td>
                    @if (@item.ProtokolZasitiGEK != null)
                    {
                        <a href="~/Uploads/BKR/@item.ProtokolZasitiGEK"
target="_blank">
                            Посмотреть
                        </a>
                    }
                    else
                    { <h5 style="color:red; ">Копия протокола о защите не
загружена</h5>}


                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Dop01VKR)
                </td>
                <td>
                    @if (@item.Dop02VKR != null)
                    {
                        <a href="~/Uploads/BKR/@item.Dop02VKR" target="_blank">
                            Посмотреть

                        </a>
                    }
                    else
                    { <h5 style="color:red; ">Копия протокола о госэкзамене не
загружена</h5>}


                </td>

                <td>
                    @Html.DisplayFor(modelItem => item.Dop03VKR)
                </td>
                <td>
                    @if (@item.Dop04VKR != null)
                    {
                        <a href="~/Uploads/BKR/VnRezenziy/@item.Dop04VKR"
target="_blank">
                            Посмотреть
                        </a>
                    }
                    else
                    { <h5 style="color:red; ">Копия внешней рецензии  не
загружена</h5>}


                </td>
  @if(@item.FailVKR == null)
  {
                <td>
```

```
                        @Html.ActionLink("Удалить", "Delete",    new { id =
item.VipuskKvalifRabotID}, new { @class = "btn btn-primary" } )
                    </td>
  }
  else
  {<td > Удалять запрещено </td>      }
                </tr>
            }

    </table>

    <br /> Страница @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) из
@Model.PageCount
    @Html.PagedListPager(Model, page => Url.Action("Index", "AdminVipuskKvalifRabots",
new { page, sortOrder = ViewBag.CurrentSort, currentFilter = ViewBag.CurrentFilter }))

@model KafedraUGLTU.Domain.Entities.VipuskKvalifRabot

@{
    ViewBag.Title = "Ввод";
    Layout = "~/Views/Shared/_LayoutAdminOsnov.cshtml";
}
<br /><br />
<h2 style="font-size:1.4em; color:#a1845a;   text-align:center;">

    АДМИНИСТРИРОВАНИЕ - выпускная квалификационная работа ВВОД НОВОЙ ЗАПИСИ



</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.StudentID, "Студент", htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("StudentID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.StudentID, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TemaVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TemaVKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TemaVKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.NachaloRazVPR, htmlAttributes: new { @class =
"control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.EditorFor(model => model.NachaloRazVPR, new { htmlAttributes = new
{ @class = "form-control", @id = "datep" } })
                @Html.ValidationMessageFor(model => model.NachaloRazVPR, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OkonchanieRazVKR, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.OkonchanieRazVKR, new { htmlAttributes =
new { @class = "form-control", @id = "datep1" } })
                @Html.ValidationMessageFor(model => model.OkonchanieRazVKR, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PredZasitaVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PredZasitaVKR, new { htmlAttributes = new
{ @class = "form-control", @id = "datep2" } })
                @Html.ValidationMessageFor(model => model.PredZasitaVKR, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OzenkaPredZasitaVKR, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.OzenkaPredZasitaVKR, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.OzenkaPredZasitaVKR, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ProtokolPredZasitaVKR, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ProtokolPredZasitaVKR, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ProtokolPredZasitaVKR, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SpravkaPlagiat, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.SpravkaPlagiat, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.SpravkaPlagiat, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.ZaklRukovodVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZaklRukovodVKR, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZaklRukovodVKR, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.FailVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FailVKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FailVKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZasitaVKRvGEK, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZasitaVKRvGEK, new { htmlAttributes = new
{ @class = "form-control", @id = "datep3" } })
                @Html.ValidationMessageFor(model => model.ZasitaVKRvGEK, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OzenkaZasitaVKRvGEK, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.OzenkaZasitaVKRvGEK, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.OzenkaZasitaVKRvGEK, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ProtokolZasitiGEK, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ProtokolZasitiGEK, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ProtokolZasitiGEK, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Dop01VKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Dop01VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop01VKR, "", new { @class =
"text-danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.Dop02VKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Dop02VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop02VKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Dop03VKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Dop03VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop03VKR, "", new { @class =
"text-danger" })
            </div>
        </div>
        <div class="form-group"><a style="float:left; color:black;">      
      Внешняя рецензия</a>
            <div class="col-md-10"  style="float:right">
                @Html.EditorFor(model => model.Dop04VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop04VKR, "", new { @class =
"text-danger" })
            </div>
        </div>



        <div class="panel-footer">
            <input type="submit" value="Сохранить в БД" class="btn btn-primary" />
            @Html.ActionLink("Отказаться и вернуться обратно", "Index", null, new
{
    @class = "btn btn-default"
})
        </div>
    </div>
}

@model KafedraUGLTU.Domain.Entities.VipuskKvalifRabot

@{
    ViewBag.Title = "Редактировать";
    Layout = "~/Views/Shared/_LayoutAdminOsnov.cshtml";
}
<br /><br />
<h2 style="font-size:1.4em; color:#a1845a;   text-align:center;">

    АДМИНИСТРИРОВАНИЕ - выпускная квалификационная работа РЕДАКТИРОВАТЬ<br /><br />


</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
```

```
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.VipuskKvalifRabotID)

        <div class="form-group">
            @Html.LabelFor(model => model.StudentID, "Студент", htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("StudentID", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.StudentID, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TemaVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TemaVKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TemaVKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.NachaloRazVPR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NachaloRazVPR, new { htmlAttributes = new
{ @class = "form-control", @id = "datep" } })
                @Html.ValidationMessageFor(model => model.NachaloRazVPR, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OkonchanieRazVKR, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.OkonchanieRazVKR, new { htmlAttributes =
new { @class = "form-control", @id = "datep1" } })
                @Html.ValidationMessageFor(model => model.OkonchanieRazVKR, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PredZasitaVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PredZasitaVKR, new { htmlAttributes = new
{ @class = "form-control", @id = "datep2" } })
                @Html.ValidationMessageFor(model => model.PredZasitaVKR, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.OzenkaPredZasitaVKR, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.OzenkaPredZasitaVKR, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.OzenkaPredZasitaVKR, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ProtokolPredZasitaVKR, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ProtokolPredZasitaVKR, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ProtokolPredZasitaVKR, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SpravkaPlagiat, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.SpravkaPlagiat, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.SpravkaPlagiat, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZaklRukovodVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZaklRukovodVKR, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZaklRukovodVKR, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.FailVKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FailVKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FailVKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZasitaVKRvGEK, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZasitaVKRvGEK, new { htmlAttributes = new
{ @class = "form-control", @id = "datep3" } })
                @Html.ValidationMessageFor(model => model.ZasitaVKRvGEK, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.OzenkaZasitaVKRvGEK, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.OzenkaZasitaVKRvGEK, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.OzenkaZasitaVKRvGEK, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ProtokolZasitiGEK, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ProtokolZasitiGEK, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ProtokolZasitiGEK, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Dop01VKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Dop01VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop01VKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Dop02VKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Dop02VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop02VKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Dop03VKR, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Dop03VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop03VKR, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <a style="float:left; color:black;">            
Внешняя рецензия</a>
            <div class="col-md-10" style="float:right">
                @Html.EditorFor(model => model.Dop04VKR, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Dop04VKR, "", new { @class =
"text-danger" })
            </div>
        </div>
```

```
        <div class="panel-footer">
            <input type="submit" value="Сохранить в БД" class="btn btn-primary" />
            @Html.ActionLink("Отказаться и вернуться обратно", "Index", null, new
{
    @class = "btn btn-default"
})
        </div>
    </div>
}
<!DOCTYPE html>
<html>
<head>

    <link href="~/Content/1GorMenu2014/VipodMenuSoySeti.css" rel="stylesheet" />

    <link href="~/Content/2UsfeuMiVEDP/UsfeuMiVEDPcss.css" rel="stylesheet" />
    @*   <link href="~/Content/1GorMenu2014/Menu2014style.css" rel="stylesheet" />  *@


    <link href="~/Content/1GorMenu2014Knopki/GorMenu2014Knopki.css" rel="stylesheet" />
    <link href="~/Content/StyleModOkno.css" rel="stylesheet" />
    <link href="~/Content/KafedraStyle.css" rel="stylesheet" />

    <link href="~/Content/MENU.css" rel="stylesheet" />
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" />
    <!-- **********************   Kalendar    Начало   ********************-->
    <script src="~/jQueryKalendar/jquery-1.10.2.js"></script>
    <script src="~/jQueryKalendar/jquery-ui-1.10.4.custom.min.js"></script>
    <script src="~/jQueryKalendar/jquery-ui-i18n.js"></script>
    <link href="~/jQueryKalendar/jquery-ui-1.10.4.custom.min.css" rel="stylesheet" />

    <link href="~/Content/StyleOblako.css" rel="stylesheet" />

    <!--    для высоты шапки меню-->
    <link href="~/Content/bootstrapMENUSAPKA.css" rel="stylesheet" />

    <style type="text/css">
        input {
            width: 200px;
            text-align: left;
            margin-right: 10px;
        }

        #wrapper > * {
            float: left;
        }
    </style>

    <script type="text/javascript">
        $(document).ready(function () {
            $("#datep").datepicker({
                changeMonth: true,
                changeYear: true,
                yearRange: "2010:2017"
            });
            $("#datep1").datepicker({
                changeMonth: true,
                changeYear: true,
                yearRange: "2010:2017"
            });
            $("#datep2").datepicker({
                changeMonth: true,
                changeYear: true,
```

```
            yearRange: "2010:2017"
        });
        $("#datep3").datepicker({
            changeMonth: true,
            changeYear: true,
            yearRange: "2010:2017"
        });
    });

</script>
<!-- **********************   Kalendar    Конец  ********************-->
<style>
    .ssilka {
        float: left;
        margin-top: 0.15em;
    }

    .ssilka1 {
        float: right;
        margin-top: 0.15em;
    }

    .img {
        max-width: 100%;
    }

    .srift1 {
        background-color: #fff;
        color: #444;
        font-family: Adamina;
        font-feature-settings: "'liga' 1";
        font-kerning: auto;
        font-style: normal;
        font-size: 1em;
        font-weight: 400;
        letter-spacing: 0;
        line-height: 1.5em;
        text-align: left;
        text-decoration: none;
        text-transform: none;
        word-spacing: 0;
    }

    .srift2 {
        background-color: #fff;
        color: #444;
        font-family: Adamina;
        font-feature-settings: "'liga' 1";
        font-kerning: auto;
        font-style: normal;
        font-size: 1em;
        font-weight: 400;
        letter-spacing: 0;
        line-height: 1.5em;
        text-align: left;
        text-decoration: none;
        text-transform: none;
        word-spacing: 0;
    }

    .srift3 {
        background-color: #fff;
        color: #444;
        font-family: Adamina;
```

```
            font-feature-settings: "'liga' 1";
            font-kerning: auto;
            font-style: normal;
            font-size: 1em;
            font-weight: 400;
            letter-spacing: 0;
            line-height: 1.5em;
            text-align: left;
            text-decoration: none;
            text-transform: none;
            word-spacing: 0;
        }
    </style>




</head>
<body style=" background:#E8E8E8;     width: 100%; " class="container ">

    <div class="navbar navbar-fixed-top navbar-inverse" role="navigation"
style="background:#7e0a0a ">

         <div class="ssilka">
            <p>
                <a style="margin-top:2em; color:#fff" href="http://www.usfeu.ru/rus/"
target="_blank">   УГЛТУ</a>

            </p>
        </div>
        <div class="ssilka1">
            <p>
                <a style="margin-top:2em;" href="http://www.feu-usfeu.ru/ "
target="_blank" class="button2"> ИЭУ   </a>

            </p>
        </div>
        <div class="ssilka1">
            <p>
                @Html.ActionLink("МенУК", "Index", "Slaider1Stranizi", new { area = "" },
new { @style = "color:#fff", @id = "registerLink" })
            </p>
        </div>


        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target="#b-menu-1" style="background:#b9a11e">
                    <span class="sr-only">Переключатель навигации</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>

            </div>
            <div class="collapse navbar-collapse" id="b-menu-1">
                <ul class="nav navbar-nav navbar-right">
                    @*
================================================================================  *@
                    <li class="dropdown  ">
```

```
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">ЭИОСО<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu" style="color:white; border-
radius:8px;">

                            <li>@Html.ActionLink("ЭИОСО", "Index", "AdminEIOS", new {
area = "" }, new { @style = "color:#000; background-color:white;  ", @class = "srift1 "
})</li>

                        </ul>
                    </li>

                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Справочники<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu">
                            <li>@Html.ActionLink("Слайдер", "Index",
"AdminSlaider1Stranizi", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Должности", "Index", "AdminDolgnosts",
new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-
primary" })</li>

                            <li>@Html.ActionLink("Название института", "Index",
"AdminInstituts", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Абриавиатура института", "Index",
"AdminInstitutAbriviaturas", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Название кафедры", "Index",
"AdminKafedras", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Абриавиатура кафедры", "Index",
"AdminKafedraAbriviaturas", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Ученая степень", "Index",
"AdminStepens", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Ученое звание", "Index",
"AdminZvanies", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Гос. звание", "Index",
"AdminZvanieGos", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Уровень публикации", "Index",
"AdminUrovenPublikaziis", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Тип учебно - научной публикации",
"Index", "AdminTipUchNauchLiteraturis", new { area = "" }, new { @style =
"color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Справочник документов 1-го уровня",
"Index", "AdminSpravDokumentis", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>

                            <li>@Html.ActionLink("Фамилии преподавателе кафедры",
"Index", "Admin_0PrepodavateliKafedri", new { area = "" }, new { @style =
"color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Форма обучения", "Index",
"Admin_0FormaObucheniy", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Форма оплаты", "Index",
"Admin_0FormaOplati", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
```

```
                            <li>@Html.ActionLink("Название практик", "Index",
"Admin_0NazvaniePraktik", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Номер контрольной работы", "Index",
"Admin_0NomerKontRaboti", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Образовательная программа", "Index",
"Admin_0ObrazovatelnayProgramma", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Уровень образование", "Index",
"Admin_0UrovenObrazovaniy", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Тип работы студента", "Index",
"Admin_0TipRaboti", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Да-Нет", "Index", "Admin_0YesNo", new {
area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary"
})</li>

                        </ul>
                    </li>
                    <li class="dropdown    ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style=" color:white;">Сотрудники<b class=" caret"></b></a>
                        <ul role="menu" class="dropdown-menu" style="color:white">

                            <li>@Html.ActionLink("НПР кафедры и ИТП", "Index",
"AdminPrepods", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>



                        </ul>
                    </li>

                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Наука<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu">
                            <li>@Html.ActionLink("Научные школы", "Index",
"AdminNauchSkolas", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("НИР", "Index", "AdminNirs", new { area
= "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary"
})</li>
                            <li>@Html.ActionLink("Малые инновационные предприятия",
"Index", "AdminMIP", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>

                            <li>@Html.ActionLink("Патенты", "Index",
"AdminIntelProducts", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Публикации", "Index",
"AdminPublikaziis", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Журнал", "Index",
"AdminGurnalEkoPotenzials", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>


                        </ul>
                    </li>

                    <li class="dropdown  ">
```

```
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Образование<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu">

                            <li>@Html.ActionLink("Библиотека", "Index",
"AdminBiblioteka", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Дисциплины", "Index",
"AdminDisziplinas", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Расписание преподавателей кафедры",
"Index", "AdminRaspisanies", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Консультация(вопрос-ответ) ", "Index",
"AdminVoprosOtvets", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>

                            <li>@Html.ActionLink("Результаты независимой оценки качества
образования кафедры", "Index", "AdminPriznanieObrazDocumentis", new { area = "" }, new {
@style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>

                        </ul>
                    </li>



                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Кафедра<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu">

                            <li>@Html.ActionLink("История кафедры", "Index",
"AdminKafedraIstoriys", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Выпускники кафедры", "Index",
"AdminKafedraVipusknikis", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Признание кафедры", "Index",
"AdminKafedraPriznanies", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Достижения студентов кафедры", "Index",
"AdminKafedraPriznanieStudentovs", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Документы кафедры", "Index",
"AdminDocumentis", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Документы2 кафедры", "Index",
"AdminDocumenti2", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;",
@class = "label-primary" })</li>
                            <li>@Html.ActionLink("Документы рейтинга", "Index",
"AdminDocumReitings", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Документы2 рейтинга", "Index",
"AdminDocumReiting2", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Показатели текущего рейтинга кафедры",
"Index", "AdminReitings", new { area = "" }, new { @style = "color:#f1dc5d;font-
weight:bolder;", @class = "label-primary" })</li>
                        </ul>
                    </li>

                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Студенту<b class="caret"></b></a>
```

```
                        <ul role="menu" class="dropdown-menu">

                            <li>@Html.ActionLink("Загрузка работ студентов - подготовка библиотек преподавателя ", "Index", "AdminZagruskaRabotiStudentovs", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            @*    <li>@Html.ActionLink("Проверка работ студентов ", "IndexUchStud", "AdminZagruskaRabotiStudentovs2", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
*@
                            <li>@Html.ActionLink("Практика", "Index", "AdminPraktikas", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Рецензии на ВКР", "Index", "AdminRezenziyNaVKRs", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Студенты, выпускаемые кафедрой", "Index", "AdminStudents", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Темы контрольных работ", "Index", "AdminTemiKontRabs", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Темы курсовых работ", "Index", "AdminTemiKursRabots", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Учеба студентов, выпускаемых кафедрой", "Index", "AdminUchebaStudentas", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Выбор контрольной работы", "Index", "AdminViborKontRabotis", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Итоговая аттестация", "Index", "AdminVipuskKvalifRabots", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Самостоятельная работа", "Index", "AdminSamRabotas", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Проверка студенческих отчетов и работ ", "IndexUchStud", "ZagruskaRabotiStudentovs2", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("График защит выпускных квалификационных работ", "Index", "AdminGrafikZasitVKRs", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>


                        </ul>
                    </li>

                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle" style="color:white;">Абитуриенту<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu">

                            <li>@Html.ActionLink("Высшее образование", "Index", "AdminAbiturientuObrazovanies", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Формы обучения", "Index", "AdminAbiturientuFormiObucheniys", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Стоимость обучения", "Index", "AdminAbiturientuStoimostObucheniys", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
                            <li>@Html.ActionLink("Трудоустройство выпускников", "Index", "AdminAbiturientuTrudoustroistvoes", new { area = "" }, new { @style = "color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>
```

```
                            <li>@Html.ActionLink("Направления и специальности кафедры",
"Index", "AdminAbiturientuNapravleniys", new { area = "" }, new { @style =
"color:#f1dc5d;font-weight:bolder;", @class = "label-primary" })</li>


                        </ul>
                    </li>
                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Контакты<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu" style="color:white; border-
radius:8px;">

                            <li>@Html.ActionLink("Контакты", "Index",
"AdminKafedraKontaktis", new { area = "" }, new { @style = "color:#000; background-
color:white;  ", @class = "srift1 " })</li>

                        </ul>
                    </li>

                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:#ffd800;">Админ<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu">

                            <li style="height:2em;">@Html.ActionLink("Администратор",
"Index", "Admin", new { area = "" }, new { @style = "max-height:4em; font-
weight:bolder;float:right;   ", })</li>

                        </ul>

                    </li>
                    <li class="dropdown  ">
                        <a href="#" data-toggle="dropdown" class="dropdown-toggle"
style="color:white;">Войти<b class="caret"></b></a>
                        <ul role="menu" class="dropdown-menu" style="color:white; border-
radius:8px;">

                            <li>
                                <div style="margin-top: 2em;">
                                    @Html.Partial("_LoginPartial")
                                </div>
            </div>
        </div>

    </div>
    <br />



    <div style="background-color:#fff">

        @RenderBody()
        <br /><br />
    </div>

    <br /><br />

    <div class="row" style="background: #7e0a0a; border-radius: 10px;">


        <div class="col-md-12" style="margin-top:1em; ">
```

```
            <div style="float:left;   color: white; width: 70%; margin-left:1%;">

                <b>  Информационные образовательные ресурсы:</b><br />

                <a style="color:white;" href="http://lib.usfeu.ru/"
target="_blank">Научная библиотека УГЛТУ</a>
                <br />
                <a style="color:white;" href="http://www.mon.gov.ru"
target="_blank">Минобрнауки РФ</a>;
                <br />
                <a style="color:white;" href="http://www.edu.ru"
target="_blank">Российское образование</a>;
                <br />
                <a style="color:white;" href="http://window.edu.ru"
target="_blank">Единое окно к образованию</a>;
                <br />
                <a style="color:white;" href="http://school-collection.edu.ru"
target="_blank">Цифровые образовательные ресурсы</a>;
                <br />
                <a style="color:white;" href="http://fcior.edu.ru"
target="_blank">Федеральный центр образовательных ресурсов</a>;<br />

            </div>


            <a href="https://twitter.com/share" class="twitter-share-button" data-
url="http://часовских.net/Account/Login?ReturnUrl=%2fhttp://" data-text="сайт профессора"
data-lang="ru" data-count="none">Твитнуть</a>
            <script>
                !function (d, s, id) {
                    var js, fjs = d.getElementsByTagName(s)[0], p =
/^http:/.test(d.location) ? 'http' : 'https';
                    if (!d.getElementById(id)) { js = d.createElement(s); js.id = id;
js.src = p + '://platform.twitter.com/widgets.js'; fjs.parentNode.insertBefore(js, fjs);
}
                }(document, 'script', 'twitter-wjs');</script>      <!--  Кнопка ТВИНУТЬ
-->
            <!-- Place this tag where you want the +1 button to render. -->
            <div class="g-plusone" data-size="medium" data-annotation="inline" data-
width="300"></div>    <!--  КНОПКА g - +  -->
            <!-- Place this tag after the last +1 button tag. -->



            <div class="share42init"></div>                                          <!--
Соц кнопки  -->

            <script src="~/Scripts/share42/share42.js"></script>
            <div class="g-ytsubscribe" data-channel="GoogleDevelopers" data-
layout="default" data-count="hidden" data-onytevent="onYtEvent"></div>  <!--  Ютуб  -->


        </div>


        <div id="footerimy" style="
 text-align:center;
 color:white;
 margin-top:7em;
">&copy;@DateTime.Now.Year- V. Chasovskikh</div>


</div>
```

```
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)

<script type="text/javascript"
src="~/SlaiderDly1Stranizi/js/responsiveCarousel.min.js"></script>
<script src="~/FooTable-master/js/footable.js"></script>
<script src="~/jQueryKalendar/jquery-ui-1.10.4.custom.min.js"></script>
<script src="~/jQueryKalendar/jquery-ui-i18n.js"></script>

<link href="~/Scripts/fancybox/jquery.fancybox.css?v=2.1.5" rel="stylesheet" />
<script src="~/Scripts/fancybox/jquery.fancybox.js"></script>
<!-- Слайдер   -->
<!-- Слайдер конец    -->
</body>
</html>
```

Тип ЭВМ реализующей программу: IBM PC совместимый компьютер

Язык программирования: Microsoft Visual C# 2017, MVC5, T-SQL, JavaScript, jQuery

Вид и версии операционной системы: Microsoft Windows Server 2012 R2 x64 VL

5. Авторизация и аутентификация

Информационную безопасность сайта и базы данных целесообразно обеспечивать одной из самых эффективных систем авторизация и аутентификация ASP.NET Identity, встроенную в ASP.NET **(**Pro ASP.NET Core).

Данная система позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями или использовать для входа на сайт учетные записи внешних провайдеров, таких как Facebook, Google, Microsoft, Twitter и других.

Функционально обеспечивается авторизация на основе ролей, Claim-Based авторизацию (относительно новое понятие и представляет пару строк "ключ-значение"), аутентификацию через соцсети, двухфакторную аутентификацию с подтверждением по смс или по электронной почте и др.

Авторизация — процесс предоставления доступа. Когда вы входите в любимую соцсеть, в личный кабинет или любой другой защищенный ресурс, вам нужен логин и пароль, чтобы вас распознали как «своего» и дали доступ к

информации.

Аутентификация — часть процесса авторизации, суть которого — проверить личность пользователя.

ASP.NET Identity представляет встроенную в ASP.NET систему аутентификации и авторизации.

Данная система позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями или использовать для входа на сайт учетные записи внешних провайдеров, таких как Facebook, Google, Microsoft, Twitter и других.

Пользователи могут создать учетную запись с информацией для входа, хранящейся в Identity, или они могут использовать внешнего поставщика входа. Поддерживаемые внешние поставщики услуг входа включают Facebook, Google, учетную запись Майкрософт и Twitter.

Исходный код Identity доступен на GitHub. Scaffold Identity и просмотр созданных файлов для просмотра взаимодействия шаблона с Identity.

Удостоверение обычно настраивается с помощью базы данных SQL Server для хранения имен пользователей, паролей и данных профиля. Кроме того, можно использовать другое постоянное хранилище, например хранилище таблиц Azure.

ASP.NET Core Identity не связан с платформой удостоверений Майкрософт.

ASP.NET Core Identity добавляет функции входа в пользовательский интерфейс (UI) для веб-приложений ASP.NET Core. Чтобы защитить веб-API и SPA, используйте один из следующих компонентов:

- Azure Active Directory
- Azure Active Directory B2C (Azure AD B2C)
- Duende IdentityServer. Duende IdentityServer является продуктом 3-й стороны.

Duende IdentityServer — это платформа OpenID Connect и OAuth 2.0 для ASP.NET Core. Duende IdentityServer включает следующие функции безопасности:

- Аутентификация как услуга (AaaS)
- Единый вход и выключение (SSO) для нескольких типов приложений
- Управление доступом для API
- Шлюз федерации

5.1. Создание веб-приложения с проверкой подлинности

Создайте проект основного веб-приложения ASP.NET с индивидуальными учетными записями пользователей.

- Visual Studio
- Интерфейс командной строки .NET Core
- Выберите шаблон ASP.NET основных веб-приложений. Присвойте проекту имя WebApp1, чтобы он имел то же пространство имен, что и загружаемый проект. Нажмите кнопку ОК.
- В поле Тип проверки подлинности выберите Учетные записи отдельных пользователей.

Сгенерированный проект предоставляет ASP.NET Core Identity как библиотеку классов Razor. Библиотека классов Identity Razor предоставляет конечные точки с областью. Например:Identity

- /Личность/Учетная запись/Логин
- /Удостоверение/Учетная запись/Выход
- /Удостоверение/Учетная запись/Управление

Применение миграций. Примените миграции для инициализации базы данных.

- Visual Studio
- Интерфейс командной строки .NET Core

Выполните следующую команду в консоли диспетчера пакетов (PMC):

Update-Database

Тест Регистрация и Вход. Запустите приложение и зарегистрируйте пользователя. В зависимости от размера экрана может потребоваться нажать кнопку переключения навигации, чтобы увидеть ссылки Регистрация и Вход.

Просмотр базы данных удостоверений

- Visual Studio
- Интерфейс командной строки .NET Core
- В меню Вид выберите Обозреватель объектов SQL Server (SSOX).
- Перейдите к (localdb)MSSQLLocalDB(SQL Server 13). Щелкните правой кнопкой мыши на dbo. AspNetUsers > просмотр данных:

Настройка служб удостоверений. Услуги добавлены в базу даннх. Типичный шаблон заключается в вызове методов в следующем порядке:

Program.cs

1. Add{Service}
2. builder.Services.Configure{Service}

C #Копировать

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using WebApp1.Data;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddRazorPages();

builder.Services.Configure<IdentityOptions>(options =>
{
    // Password settings.
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 6;
    options.Password.RequiredUniqueChars = 1;

    // Lockout settings.
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
    options.Lockout.MaxFailedAccessAttempts = 5;
    options.Lockout.AllowedForNewUsers = true;
```

```
        // User settings.
        options.User.AllowedUserNameCharacters =

"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789-._@+";
        options.User.RequireUniqueEmail = false;
    });

    builder.Services.ConfigureApplicationCookie(options =>
    {
        // Cookie settings
        options.Cookie.HttpOnly = true;
        options.ExpireTimeSpan = TimeSpan.FromMinutes(5);

        options.LoginPath = "/Identity/Account/Login";
        options.AccessDeniedPath = "/Identity/Account/AccessDenied";
        options.SlidingExpiration = true;
    });

    var app = builder.Build();

    if (app.Environment.IsDevelopment())
    {
        app.UseMigrationsEndPoint();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
```

```
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapRazorPages();

app.Run();
```

Приведенный код программы настраивает идентификатор со значениями параметров по умолчанию. Службы становятся доступными для приложения с помощью внедрения зависимостей.

Удостоверение включается путем вызова метода UseAuthentication. добавляет промежуточное ПО проверки подлинности в конвейер запросов. UseAuthentication. Приложение, созданное шаблоном, не использует авторизацию. включено, чтобы убедиться, что он добавлен в правильном порядке, если приложение добавляет авторизацию. , и должен вызываться в порядке, указанном в предыдущем коде.app.UseAuthorizationUseRoutingUseAuthenticationUseAuthorization

Регистрация, вход, вход в систему и регистрация Подтверждение

- Visual Studio
- Интерфейс командной строки .NET Core

Добавьте файлы и следуйте идентификатору Scaffold в проекте Razor с инструкциями авторизации, чтобы создать код.

Проверить реестр. Когда пользователь нажимает кнопку Регистрация на странице, вызывается действие. Пользователь создается методом CreateAsync(TUser) на объекте: RegisterRegisterModel.OnPostAsync_userManager

C #Копировать

```
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl = returnUrl ?? Url.Content("~/");
    ExternalLogins = (await
_signInManager.GetExternalAuthenticationSchemesAsync())
                        .ToList();
    if (ModelState.IsValid)
```

```
        {
            var user = new IdentityUser { UserName = Input.Email, Email = Input.Email };
            var result = await _userManager.CreateAsync(user, Input.Password);
            if (result.Succeeded)
            {
                _logger.LogInformation("User created a new account with password.");

                var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
                code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
                var callbackUrl = Url.Page(
                    "/Account/ConfirmEmail",
                    pageHandler: null,
                    values: new { area = "Identity", userId = user.Id, code = code },
                    protocol: Request.Scheme);

                await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
                    $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

                if (_userManager.Options.SignIn.RequireConfirmedAccount)
                {
                    return RedirectToPage("RegisterConfirmation",
                                new { email = Input.Email });
                }
                else
                {
                    await _signInManager.SignInAsync(user, isPersistent: false);
                    return LocalRedirect(returnUrl);
                }
            }
            foreach (var error in result.Errors)
```

```
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }

        // If we got this far, something failed, redisplay form
        return Page();
    }
```

Отключение проверки учетной записи по умолчанию. С помощью шаблонов по умолчанию пользователь перенаправляется туда, где он может выбрать ссылку для подтверждения учетной записи. Значение по умолчанию используется только для тестирования, автоматическая проверка учетной записи должна быть отключена в рабочем приложении.Account.RegisterConfirmationAccount.RegisterConfirmation

Чтобы потребовать подтвержденную учетную запись и предотвратить немедленный вход при регистрации, установите :DisplayConfirmAccountLink = false/Areas/Identity/Pages/Account/RegisterConfirmation.cshtml.cs

```
C #Копировать
[AllowAnonymous]
public class RegisterConfirmationModel : PageModel
{
    private readonly UserManager<IdentityUser> _userManager;
    private readonly IEmailSender _sender;

    public RegisterConfirmationModel(UserManager<IdentityUser> userManager, IEmailSender sender)
    {
        _userManager = userManager;
        _sender = sender;
    }

    public string Email { get; set; }

    public bool DisplayConfirmAccountLink { get; set; }
```

```
        public string EmailConfirmationUrl { get; set; }

        public async Task<IActionResult> OnGetAsync(string email, string returnUrl
= null)
        {
            if (email == null)
            {
                return RedirectToPage("/Index");
            }

            var user = await _userManager.FindByEmailAsync(email);
            if (user == null)
            {
                return NotFound($"Unable to load user with email '{email}'.");
            }

            Email = email;
            // Once you add a real email sender, you should remove this code that lets
you confirm the account
            DisplayConfirmAccountLink = false;
            if (DisplayConfirmAccountLink)
            {
                var userId = await _userManager.GetUserIdAsync(user);
                var                code                =                await
_userManager.GenerateEmailConfirmationTokenAsync(user);
                code                                                        =
WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
                EmailConfirmationUrl = Url.Page(
                    "/Account/ConfirmEmail",
                    pageHandler: null,
                    values: new { area = "Identity", userId = userId, code = code, returnUrl
= returnUrl },
                    protocol: Request.Scheme);
            }
```

```
            return Page();
        }
    }
```

Войти. Форма Входа отображается, когда:

- Выбрана ссылка Войти.
- Пользователь пытается получить доступ к странице с ограниченным доступом, к которой он не авторизован, или если он не прошел проверку подлинности в системе.

При отправке формы на странице входа вызывается действие. вызывается на объекте.OnPostAsyncPasswordSignInAsync_signInManager

```
    public async Task<IActionResult> OnPostAsync(string returnUrl = null)
    {
        returnUrl = returnUrl ?? Url.Content("~/");

        if (ModelState.IsValid)
        {
            // This doesn't count login failures towards account lockout
            // To enable password failures to trigger account lockout,
            // set lockoutOnFailure: true
            var result = await _signInManager.PasswordSignInAsync(Input.Email,
                        Input.Password, Input.RememberMe, lockoutOnFailure: true);
            if (result.Succeeded)
            {
                _logger.LogInformation("User logged in.");
                return LocalRedirect(returnUrl);
            }
            if (result.RequiresTwoFactor)
            {
                return RedirectToPage("./LoginWith2fa", new
                {
                    ReturnUrl = returnUrl,
                    RememberMe = Input.RememberMe
                });
            }
```

```
            if (result.IsLockedOut)
            {
                _logger.LogWarning("User account locked out.");
                return RedirectToPage("./Lockout");
            }
            else
            {
                ModelState.AddModelError(string.Empty, "Invalid login attempt.");
                return Page();
            }
        }

        // If we got this far, something failed, redisplay form
        return Page();
    }
```

Выход. Ссылка Выход вызывает действие.LogoutModel.OnPost

```
    using Microsoft.AspNetCore.Authorization;
    using Microsoft.AspNetCore.Identity;
    using Microsoft.AspNetCore.Mvc;
    using Microsoft.AspNetCore.Mvc.RazorPages;
    using Microsoft.Extensions.Logging;
    using System.Threading.Tasks;

    namespace WebApp1.Areas.Identity.Pages.Account
    {
        [AllowAnonymous]
        public class LogoutModel : PageModel
        {
            private readonly SignInManager<IdentityUser> _signInManager;
            private readonly ILogger<LogoutModel> _logger;

            public LogoutModel(SignInManager<IdentityUser> signInManager,
ILogger<LogoutModel> logger)
            {
```

```
            _signInManager = signInManager;
            _logger = logger;
        }

        public void OnGet()
        {
        }

        public async Task<IActionResult> OnPost(string returnUrl = null)
        {
            await _signInManager.SignOutAsync();
            _logger.LogInformation("User logged out.");
            if (returnUrl != null)
            {
                return LocalRedirect(returnUrl);
            }
            else
            {
                return RedirectToPage();
            }
        }
    }
}
```

В предыдущем коде код должен быть перенаправлением, чтобы браузер выполнил новый запрос и получил идентификатор пользователя updated.return RedirectToPage();

SignOutAsync clears the user's claims stored in a cookie.

Post is specified in the :Pages/Shared/_LoginPartial.cshtml

CSHTMLCopy

```
@using Microsoft.AspNetCore.Identity
@inject SignInManager<IdentityUser> SignInManager
@inject UserManager<IdentityUser> UserManager

<ul class="navbar-nav">
@if (SignInManager.IsSignedIn(User))
```

```
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index"
                                    title="Manage">Hello @User.Identity.Name!</a>
        </li>
        <li class="nav-item">
            <form class="form-inline" asp-area="Identity" asp-page="/Account/Logout"
                                asp-route-returnUrl="@Url.Page("/", new { area = "" })"
                                method="post" >
                <button type="submit" class="nav-link btn btn-link text-dark">Logout</button>
            </form>
        </li>
    }
    else
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Register</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Login">Login</a>
        </li>
    }
    </ul>
```

Тестовое удостоверение. Шаблоны веб-проектов по умолчанию разрешают анонимный доступ к домашним страницам. Чтобы проверить удостоверение, добавьте [Авторизовать]:

C #Копировать

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Logging;
```

```
namespace WebApp1.Pages
{
    [Уполномочивать]
    public class PrivacyModel : PageModel
    {
        private readonly ILogger<PrivacyModel> _logger;

        public PrivacyModel(ILogger<PrivacyModel> logger)
        {
            _logger = logger;
        }

        public void OnGet()
        {
        }
    }
}
```

Если вы вошли в систему, выйдите из системы. Запустите приложение и выберите ссылку Конфиденциальность. Вы будете перенаправлены на страницу входа.

Исследуйте идентичность. Чтобы изучить Идентификацию более подробно:

• Создание источника пользовательского интерфейса с полным удостоверением

• Изучите источник каждой страницы и выполните пошаговое выполнение отладчика.

Компоненты удостоверений. Все зависящие от удостоверений пакеты NuGet включены в общую платформу ASP.NET Core.

Основным пакетом для удостоверения является Microsoft.AspNetCore.Identity. Этот пакет содержит основной набор интерфейсов для ASP.NET Core Identity и входит в состав .Microsoft.AspNetCore.Identity.EntityFrameworkCore

Миграция на ASP.NET core identity. Дополнительные сведения и рекомендации по переносу существующего хранилища удостоверений см. в

разделе Перенос проверки подлинности и удостоверения.

Установка надежности пароля. В разделе Конфигурация приведен пример, устанавливающий минимальные требования к паролям.

AddDefaultIdentity и AddIdentity

AddDefaultIdentity был представлен в ASP.NET Core 2.1. Вызов аналогичен вызову следующего:AddDefaultIdentity

- AddIdentity
- AddDefaultUI
- AddDefaultTokenProviders

.

Запретить публикацию статических ресурсов удостоверений. Чтобы запретить публикацию статических ресурсов удостоверений (таблиц стилей и файлов JavaScript для пользовательского интерфейса удостоверений) в корневом веб-каталоге, добавьте в файл проекта приложения следующее свойство и целевой объект: ResolveStaticWebAssetsInputsDependsOnRemoveIdentityAssets

.XMLКопировать

```
<PropertyGroup>

<ResolveStaticWebAssetsInputsDependsOn>RemoveIdentityAssets</ResolveStaticWebAssetsInputsDependsOn>
</PropertyGroup>

<Target Name="RemoveIdentityAssets">
  <ItemGroup>
    <StaticWebAsset Remove="@(StaticWebAsset)" Condition="%(SourceId) == 'Microsoft.AspNetCore.Identity.UI'" />
  </ItemGroup>
</Target>
```

5.2. Добавление, загрузка и удаление пользовательских данных в Identity в проекте ASP.NET Core

Необходимые условия

Пакет SDK для .NET 6.0

Создание веб-приложения Razor

- Visual Studio
- Интерфейс командной строки .NET Core
- В меню Файл Visual Studio выберите Команду Создать проект >. Присвойте проекту имя WebApp1, если вы хотите, чтобы он соответствовал пространству имен загружаемого примера кода.
- Выберите ASP.NET основное веб-приложение > ОК
- Выберите Веб-приложение > ОК
- Выполните сборку и запустите проект.

Запуск скаффапола удостоверений

- Visual Studio
- Интерфейс командной строки .NET Core
- В обозревателе решений щелкните правой кнопкой мыши проект > Добавить > новый элемент формирования шаблонов.
- В левой области диалогового окна Добавление шаблона выберите Удостоверение > Добавить.
- В диалоговом окне Добавление удостоверения используются следующие параметры:
  - Выберите существующий файл макета ~/Pages/Shared/_Layout.cshtml
  - Выберите следующие файлы для переопределения:
    - Учетная запись/Регистрация
    - Аккаунт/Управление/Индекс
  - Нажмите кнопку +, чтобы создать новый класс контекста данных. Примите тип (WebApp1.Models.WebApp1Context, если проект называется WebApp1).
  - Нажмите кнопку +, чтобы создать новый класс User. Примите тип (WebApp1User, если проект называется WebApp1) > Add.
- Выберите Добавить.

Следуйте инструкциям в разделах Миграция, ИспользованиеАутентификация и макет, чтобы выполнить следующие действия.

- Создайте миграцию и обновите базу данных.
- Добавить в программу.csUseAuthentication
- Добавьте в файл макета.<partial name="_LoginPartial" />
- Протестируйте приложение:
  - Регистрация пользователя

o Выберите новое имя пользователя (рядом со ссылкой Выход). Возможно, потребуется развернуть окно или выбрать значок панели навигации, чтобы отобразить имя пользователя и другие ссылки.

o Выберите вкладку Личные данные.

o Нажмите кнопку Загрузить и изучите файл.PersonalData.json

o Проверьте кнопку Удалить, которая удаляет вошедшего в систему пользователя.

Добавление пользовательских данных в базу данных удостоверений. Обновите производный класс пользовательскими свойствами. Если проект назван WebApp1, файл будет называться . Обновите файл следующим кодом:IdentityUserAreas/Identity/Data/WebApp1User.cs

```
using Microsoft.AspNetCore.Identity;

namespace WebApp1.Areas.Identity.Data;

public class WebApp1User : IdentityUser
{
   [PersonalData]
   public string ? Name { get; set; }
   [PersonalData]
   public DateTime DOB { get; set; }
}
```

Свойства с атрибутом PersonalData:

• Удаляется при вызове Razor Page .Areas/Identity/Pages/Account/Manage/DeletePersonalData.cshtmlUserManager.Delete

• Включено в данные, загруженные страницей Razor.Areas/Identity/Pages/Account/Manage/DownloadPersonalData.cshtml

Обновление страницыAccount/Manage/Index.cshtml. Обновите вход следующим выделенным кодом: InputModelAreas/Identity/Pages/Account/Manage/Index.cshtml.cs

```
public class IndexModel : PageModel
{
```

```
        private readonly UserManager<WebApp1User> _userManager;
        private readonly SignInManager<WebApp1User> _signInManager;

        public IndexModel(
            UserManager<WebApp1User> userManager,
            SignInManager<WebApp1User> signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }

        /// <summary>
        ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
        ///     directly from your code. This API may change or be removed in future releases.
        /// </summary>
        public string Username { get; set; }

        // Remaining API warnings ommited.

        [TempData]
        public string StatusMessage { get; set; }

        [BindProperty]
        public InputModel Input { get; set; }

        public class InputModel
        {
            [Обязательно]
      [Тип данных(Тип данных.Текст)]
      [Отображение(Имя = "Полное имя")] публичная строка Name { get; комплект; }

     [Обязательно]
```

```
    [Отображение(Имя = "Дата рождения")]
    [Тип данных(Тип данных.Дата)] общественный DateTime DOB { get; набор; }

        [Phone]
        [Display(Name = "Phone number")]
        public string PhoneNumber { get; set; }
    }

    private async Task LoadAsync(WebApp1User user)
    {
        var userName = await _userManager.GetUserNameAsync(user);
        var phoneNumber = await _userManager.GetPhoneNumberAsync(user);

        Username = userName;

        Input = new InputModel
        {
  Имя = пользователь. Имя
  DOB = пользователь. Дата рождения,
            PhoneNumber = phoneNumber
        };
    }

    public async Task<IActionResult> OnGetAsync()
    {
        var user = await _userManager.GetUserAsync(User);
        if (user == null)
        {
            return NotFound($"Unable to load user with ID '{_userManager.GetUserId(User)}'.");
        }

        await LoadAsync(user);
        return Page();
```

```
        }

        public async Task<IActionResult> OnPostAsync()
        {
            var user = await _userManager.GetUserAsync(User);
            if (user == null)
            {
                return NotFound($"Unable to load user with ID
'{_userManager.GetUserId(User)}'.");
            }

            if (!ModelState.IsValid)
            {
                await LoadAsync(user);
                return Page();
            }

            var phoneNumber = await _userManager.GetPhoneNumberAsync(user);
            if (Input.PhoneNumber != phoneNumber)
            {
                var setPhoneResult = await _userManager.SetPhoneNumberAsync(user,
Input.PhoneNumber);
                if (!setPhoneResult.Succeeded)
                {
                    StatusMessage = "Unexpected error when trying to set phone
number.";
                    return RedirectToPage();
                }
            }

            if (Input.Name != пользователь. Имя)
        {
        пользователь. Имя = Input.Name;
        } if (Input.DOB != user. Дата рождения)
        {
```

```
пользователь. DOB = Вход.DOB;
} await _userManager.UpdateAsync(user);
    await _signInManager.RefreshSignInAsync(user);
    StatusMessage = "Your profile has been updated";
    return RedirectToPage();
  }
}
```

Обновите следующую выделенную разметку: Areas/Identity/Pages/Account/Manage/Index.cshtml

CSHTMLКопировать

```
@page
@model IndexModel
@{
  ViewData["Title"] = "Profile";
  ViewData["ActivePage"] = ManageNavPages.Index;
}

<h3>@ViewData["Title"]</h3>
<partial name="_StatusMessage" for="StatusMessage" />
<div class="row">
  <div class="col-md-6">
    <form id="profile-form" method="post">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-floating">
        <input asp-for="Username" class="form-control" disabled />
        <label asp-for="Username" class="form-label"></label>
      </div>
 <div class="form-group"> <label asp-for="Input.Name" class="form-control"></label> <input asp-for="Input.Name" class="form-label" /> </div> <div class="form-group"> <label asp-for="Input.DOB" class="form-control"></label> <input asp-for="Input.DOB" class="form-label" /> </div>
      <div class="form-floating">
        <input asp-for="Input.PhoneNumber" class="form-control" />
        <label asp-for="Input.PhoneNumber" class="form-label"></label>
```

```
                <span asp-validation-for="Input.PhoneNumber" class="text-danger"></span>
            </div>
            <button id="update-profile-button" type="submit" class="w-100 btn btn-lg btn-primary">Save</button>
        </form>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Обновление страницыAccount/Register.cshtml. Обновите вход следующим выделенным кодом:InputModelAreas/Identity/Pages/Account/Register.cshtml.cs

C #Копировать

```
public class RegisterModel : PageModel
{
    private readonly SignInManager<WebApp1User> _signInManager;
    private readonly UserManager<WebApp1User> _userManager;
    private readonly IUserStore<WebApp1User> _userStore;
    private readonly IUserEmailStore<WebApp1User> _emailStore;
    private readonly ILogger<RegisterModel> _logger;
    private readonly IEmailSender _emailSender;

    public RegisterModel(
        UserManager<WebApp1User> userManager,
        IUserStore<WebApp1User> userStore,
        SignInManager<WebApp1User> signInManager,
        ILogger<RegisterModel> logger,
        IEmailSender emailSender)
    {
        _userManager = userManager;
        _userStore = userStore;
        _emailStore = GetEmailStore();
        _signInManager = signInManager;
```

```
            _logger = logger;
            _emailSender = emailSender;
        }

        /// <summary>
        ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
        ///     directly from your code. This API may change or be removed in future releases.
        /// </summary>
        [BindProperty]
        public InputModel Input { get; set; }

        // Remaining API warnings ommited.
        public string ReturnUrl { get; set; }

        public IList<AuthenticationScheme> ExternalLogins { get; set; }

        public class InputModel
        {
            [Обязательно]
    [Тип данных(Тип данных.Текст)]
    [Отображение(Имя = "Полное имя")] публичная строка Name { get; комплект; }

    [Обязательно]
    [Отображение(Имя = "Дата рождения")]
    [Тип данных(Тип данных.Дата)] общественный DateTime DOB { get; набор; }

            [Required]
            [EmailAddress]
            [Display(Name = "Email")]
            public string Email { get; set; }
```

```
            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at
max {1} characters long.", MinimumLength = 6)]
            [DataType(DataType.Password)]
            [Display(Name = "Password")]
            public string Password { get; set; }

            [DataType(DataType.Password)]
            [Display(Name = "Confirm password")]
            [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
            public string ConfirmPassword { get; set; }
        }


        public async Task OnGetAsync(string returnUrl = null)
        {
            ReturnUrl = returnUrl;
            ExternalLogins                    =                    (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
        }

        public async Task<IActionResult> OnPostAsync(string returnUrl = null)
        {
            returnUrl ??= Url.Content("~/");
            ExternalLogins                    =                    (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
            if (ModelState.IsValid)
            {
                var user = CreateUser();

                user.Name = Input.Name;
                user.DOB = Input.DOB;

                await          _userStore.SetUserNameAsync(user,          Input.Email,
```

```
CancellationToken.None);
                await _emailStore.SetEmailAsync(user, Input.Email,
CancellationToken.None);
                var result = await _userManager.CreateAsync(user, Input.Password);

                if (result.Succeeded)
                {
                    _logger.LogInformation("User created a new account with
password.");

                    var userId = await _userManager.GetUserIdAsync(user);
                    var code = await
_userManager.GenerateEmailConfirmationTokenAsync(user);
                    code =
WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
                    var callbackUrl = Url.Page(
                        "/Account/ConfirmEmail",
                        pageHandler: null,
                        values: new { area = "Identity", userId = userId, code = code,
returnUrl = returnUrl },
                        protocol: Request.Scheme);

                    await _emailSender.SendEmailAsync(Input.Email, "Confirm your
email",
                        $"Please confirm your account by <a
href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

                    if (_userManager.Options.SignIn.RequireConfirmedAccount)
                    {
                        return RedirectToPage("RegisterConfirmation", new { email =
Input.Email, returnUrl = returnUrl });
                    }
                    else
                    {
                        await _signInManager.SignInAsync(user, isPersistent: false);
```

```
                    return LocalRedirect(returnUrl);
                }
            }
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }

        // If we got this far, something failed, redisplay form
        return Page();
    }

    private WebApp1User CreateUser()
    {
        try
        {
            return Activator.CreateInstance<WebApp1User>();
        }
        catch
        {
            throw new InvalidOperationException($"Can't create an instance of '{nameof(WebApp1User)}'. " +
                $"Ensure that '{nameof(WebApp1User)}' is not an abstract class and has a parameterless constructor, or alternatively " +
                $"override the register page in /Areas/Identity/Pages/Account/Register.cshtml");
        }
    }

    private IUserEmailStore<WebApp1User> GetEmailStore()
    {
        if (!_userManager.SupportsUserEmail)
        {
            throw new NotSupportedException("The default UI requires a user
```

store with email support.");
            }
            return (IUserEmailStore<WebApp1User>)_userStore;
        }
    }
}

Обновите следующую выделенную разметку: 6.3. Areas/Identity/Pages/Account/Register.cshtml

CSHTMLКопировать

```
@page
@model RegisterModel
@{
    ViewData["Title"] = "Register";
}

<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-4">
        <form id="registerForm" asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <h2>Create a new account.</h2>
            <hr />
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

     <div class="form-floating"> <label asp-for="Input.Name"></label> <input asp-for="Input.Name" class="form-control" /> <span asp-validation-for="Input.Name" class="text-danger"></ span> </div> <div class="form-floating"> <label asp-for="Input.DOB"></label> <input asp-for="Input.DOB" class="form-control" /> <span asp-validation-for="Input.DOB" class="text-danger"></span> </div>

            <div class="form-floating">
                <input asp-for="Input.Email" class="form-control"
```

```
autocomplete="username" aria-required="true" />
                <label asp-for="Input.Email"></label>
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
            <div class="form-floating">
                <input asp-for="Input.Password" class="form-control"
autocomplete="new-password" aria-required="true" />
                <label asp-for="Input.Password"></label>
                <span asp-validation-for="Input.Password" class="text-
danger"></span>
            </div>
            <div class="form-floating">
                <input asp-for="Input.ConfirmPassword" class="form-control"
autocomplete="new-password" aria-required="true" />
                <label asp-for="Input.ConfirmPassword"></label>
                <span asp-validation-for="Input.ConfirmPassword" class="text-
danger"></span>
            </div>
            <button id="registerSubmit" type="submit" class="w-100 btn btn-lg btn-
primary">Register</button>
        </form>
    </div>
    <div class="col-md-6 col-md-offset-2">
        <section>
            <h3>Use another service to register.</h3>
            <hr />
            @{
                if ((Model.ExternalLogins?.Count ?? 0) == 0)
                {
                    <div>
                        <p>
                            There are no external authentication services configured. See
this <a href="https://go.microsoft.com/fwlink/?LinkID=532715">article
                            about setting up this ASP.NET application to support logging
in via external services</a>.
```

```
                    </p>
                </div>
            }
            else
            {
                <form id="external-account" asp-page="./ExternalLogin" asp-route-returnUrl="@Model.ReturnUrl" method="post" class="form-horizontal">
                    <div>
                        <p>
                            @foreach (var provider in Model.ExternalLogins!)
                            {
                                <button type="submit" class="btn btn-primary" name="provider" value="@provider.Name" title="Log in using your @provider.DisplayName account">@provider.DisplayName</button>
                            }
                        </p>
                    </div>
                </form>
            }
        }
    </section>
  </div>
</div>

@section Scripts {
  <partial name="_ValidationScriptsPartial" />
}
```

Постройте проект.

5.3. Настройка модели идентификации в ASP.NET Core

Identity ASP.NET Core предоставляет платформу для управления учетными записями пользователей и их хранения в приложениях ASP.NET Core. Identity добавляется в проект при выборе отдельных учетных записей пользователей в качестве механизма проверки подлинности. По умолчанию Identity

используется модель данных Entity Framework (EF) Core.

Identity и EF Core миграции. Прежде чем изучать модель, полезно понять, как Identity работает с EF Core миграциями для создания и обновления базы данных. На верхнем уровне процесс выполняется следующим образом:

1. Определение или обновление модели данных в коде.
2. Добавьте миграцию для преобразования этой модели в изменения, которые можно применить к базе данных.
3. Убедитесь, что миграция правильно представляет ваши намерения.
4. Примените миграцию, чтобы обновить базу данных для синхронизации с моделью.
5. Повторите шаги с 1 по 4, чтобы дополнительно уточнить модель и сохранить базу данных в синхронизации.

Используйте один из следующих подходов для добавления и применения миграций.

- Окно консоли диспетчера пакетов (PMC) при использовании Visual Studio.
- Интерфейс командной строки .NET Core, если используется командная строка.
- При запуске приложения нажмите кнопку "Применить миграции " на странице ошибки.

ASP.NET Core имеет обработчик страницы ошибок во время разработки. Обработчик может применять миграции при запуске приложения. Рабочие приложения обычно создают скрипты SQL из миграций и развертывают изменения базы данных в рамках управляемого развертывания приложения и базы данных.

После создания нового приложения Identity шаги 1 и 2 выше уже завершены. То есть исходная модель данных уже существует, и первоначальная миграция была добавлена в проект. Первоначальная миграция по-прежнему должна применяться к базе данных. Начальная миграция может применяться одним из следующих подходов:

- Запустите Update-Database в PMC.
- Выполните dotnet ef database update в командной оболочке.
- Нажмите кнопку "Применить миграцию " на странице ошибки при запуске приложения.

Повторите предыдущие шаги по мере внесения изменений в модель.

Модель Identity и Типы сущностей. Модель Identity состоит из следующих типов сущностей.

Тип сущности Описание

User Представляет пользователя.

Role Представляет роль.

UserClaim Представляет утверждение о том, что пользователь обладает.

UserToken Представляет маркер проверки подлинности для пользователя.

UserLogin Связывает пользователя с именем входа.

RoleClaim Представляет утверждение, которое предоставляется всем пользователям в роли.

UserRole Сущность соединения, которая связывает пользователей и роли.

Связи типов сущностей. Типы сущностей связаны друг с другом следующими способами:

- Каждый из них User может иметь много UserClaims.
- Каждый из них User может иметь много UserLogins.
- Каждый из них User может иметь много UserTokens.
- Каждый из них Role может иметь много связанных RoleClaims.
- Каждый User может иметь много связанных Roles, и каждый из них Role может быть связан со многими Users. Это связь "многие ко многим", которая требует таблицы соединения в базе данных. Таблица соединения представлена сущностью UserRole .

Конфигурация модели по умолчанию. Identity определяет множество классов контекста , наследующих от DbContext настройки и использования модели. Эта конфигурация выполняется с помощью EF Core API Code First Fluent в OnModelCreating методе класса контекста. Конфигурация по умолчанию:

C#Копировать

```
builder.Entity<TUser>(b =>
{
    // Primary key
    b.HasKey(u => u.Id);
```

```
        // Indexes for "normalized" username and email, to allow efficient lookups
        b.HasIndex(u => u.NormalizedUserName).HasName("UserNameIndex").IsUnique();
        b.HasIndex(u => u.NormalizedEmail).HasName("EmailIndex");

        // Maps to the AspNetUsers table
        b.ToTable("AspNetUsers");

        // A concurrency token for use with the optimistic concurrency checking
        b.Property(u => u.ConcurrencyStamp).IsConcurrencyToken();

        // Limit the size of columns to use efficient database types
        b.Property(u => u.UserName).HasMaxLength(256);
        b.Property(u => u.NormalizedUserName).HasMaxLength(256);
        b.Property(u => u.Email).HasMaxLength(256);
        b.Property(u => u.NormalizedEmail).HasMaxLength(256);

        // The relationships between User and other entity types
        // Note that these relationships are configured with no navigation properties

        // Each User can have many UserClaims
        b.HasMany<TUserClaim>().WithOne().HasForeignKey(uc => uc.UserId).IsRequired();

        // Each User can have many UserLogins
        b.HasMany<TUserLogin>().WithOne().HasForeignKey(ul => ul.UserId).IsRequired();

        // Each User can have many UserTokens
        b.HasMany<TUserToken>().WithOne().HasForeignKey(ut => ut.UserId).IsRequired();

        // Each User can have many entries in the UserRole join table
        b.HasMany<TUserRole>().WithOne().HasForeignKey(ur => ur.UserId).IsRequired();
```

```
});

builder.Entity<TUserClaim>(b =>
{
    // Primary key
    b.HasKey(uc => uc.Id);

    // Maps to the AspNetUserClaims table
    b.ToTable("AspNetUserClaims");
});

builder.Entity<TUserLogin>(b =>
{
    // Composite primary key consisting of the LoginProvider and the key to use
    // with that provider
    b.HasKey(l => new { l.LoginProvider, l.ProviderKey });

    // Limit the size of the composite key columns due to common DB restrictions
    b.Property(l => l.LoginProvider).HasMaxLength(128);
    b.Property(l => l.ProviderKey).HasMaxLength(128);

    // Maps to the AspNetUserLogins table
    b.ToTable("AspNetUserLogins");
});

builder.Entity<TUserToken>(b =>
{
    // Composite primary key consisting of the UserId, LoginProvider and Name
    b.HasKey(t => new { t.UserId, t.LoginProvider, t.Name });

    // Limit the size of the composite key columns due to common DB restrictions
    b.Property(t => t.LoginProvider).HasMaxLength(maxKeyLength);
    b.Property(t => t.Name).HasMaxLength(maxKeyLength);

    // Maps to the AspNetUserTokens table
```

```
        b.ToTable("AspNetUserTokens");
    });

    builder.Entity<TRole>(b =>
    {
        // Primary key
        b.HasKey(r => r.Id);

        // Index for "normalized" role name to allow efficient lookups
        b.HasIndex(r => r.NormalizedName).HasName("RoleNameIndex").IsUnique();

        // Maps to the AspNetRoles table
        b.ToTable("AspNetRoles");

        // A concurrency token for use with the optimistic concurrency checking
        b.Property(r => r.ConcurrencyStamp).IsConcurrencyToken();

        // Limit the size of columns to use efficient database types
        b.Property(u => u.Name).HasMaxLength(256);
        b.Property(u => u.NormalizedName).HasMaxLength(256);

        // The relationships between Role and other entity types
        // Note that these relationships are configured with no navigation properties

        // Each Role can have many entries in the UserRole join table
        b.HasMany<TUserRole>().WithOne().HasForeignKey(ur => ur.RoleId).IsRequired();

        // Each Role can have many associated RoleClaims
        b.HasMany<TRoleClaim>().WithOne().HasForeignKey(rc => rc.RoleId).IsRequired();
    });

    builder.Entity<TRoleClaim>(b =>
```

```
{
    // Primary key
    b.HasKey(rc => rc.Id);

    // Maps to the AspNetRoleClaims table
    b.ToTable("AspNetRoleClaims");
});

builder.Entity<TUserRole>(b =>
{
    // Primary key
    b.HasKey(r => new { r.UserId, r.RoleId });

    // Maps to the AspNetUserRoles table
    b.ToTable("AspNetUserRoles");
});
```

Универсальные типы моделей. Identity определяет типы среды CLR по умолчанию для каждого из перечисленных выше типов сущностей. Все эти типы имеют префикс:Identity

- IdentityUser
- IdentityRole
- IdentityUserClaim
- IdentityUserToken
- IdentityUserLogin
- IdentityRoleClaim
- IdentityUserRole

Вместо прямого использования этих типов типы можно использовать в качестве базовых классов для собственных типов приложения. Классы DbContext , определенные с помощью Identity универсальных типов, могут использоваться для одного или нескольких типов сущностей в модели. Эти универсальные типы также позволяют изменять тип данных первичного User ключа (PK).

При использовании Identity с поддержкой IdentityDbContext ролей следует использовать класс. Пример:

C#Копировать

```
    // Uses all the built-in Identity types
    // Uses `string` as the key type
    public class IdentityDbContext
        : IdentityDbContext<IdentityUser, IdentityRole, string>
    {
    }

    // Uses the built-in Identity types except with a custom User type
    // Uses `string` as the key type
    public class IdentityDbContext<TUser>
        : IdentityDbContext<TUser, IdentityRole, string>
            where TUser : IdentityUser
    {
    }

    // Uses the built-in Identity types except with custom User and Role types
    // The key type is defined by TKey
    public class IdentityDbContext<TUser, TRole, TKey> : IdentityDbContext<
        TUser, TRole, TKey, IdentityUserClaim<TKey>, IdentityUserRole<TKey>,
        IdentityUserLogin<TKey>, IdentityRoleClaim<TKey>,
IdentityUserToken<TKey>>
            where TUser : IdentityUser<TKey>
            where TRole : IdentityRole<TKey>
            where TKey : IEquatable<TKey>
    {
    }

    // No built-in Identity types are used; all are specified by generic arguments
    // The key type is defined by TKey
    public abstract class IdentityDbContext<
        TUser, TRole, TKey, TUserClaim, TUserRole, TUserLogin, TRoleClaim,
TUserToken>
        : IdentityUserContext<TUser, TKey, TUserClaim, TUserLogin,
TUserToken>
            where TUser : IdentityUser<TKey>
```

```
        where TRole : IdentityRole<TKey>
        where TKey : IEquatable<TKey>
        where TUserClaim : IdentityUserClaim<TKey>
        where TUserRole : IdentityUserRole<TKey>
        where TUserLogin : IdentityUserLogin<TKey>
        where TRoleClaim : IdentityRoleClaim<TKey>
        where TUserToken : IdentityUserToken<TKey>
```

Кроме того, можно использовать Identity без ролей (только утверждений), в этом случае IdentityUserContext<TUser> следует использовать класс:

C#Копировать

```
// Uses the built-in non-role Identity types except with a custom User type
// Uses `string` as the key type
public class IdentityUserContext<TUser>
    : IdentityUserContext<TUser, string>
        where TUser : IdentityUser
{
}

// Uses the built-in non-role Identity types except with a custom User type
// The key type is defined by TKey
public class IdentityUserContext<TUser, TKey> : IdentityUserContext<
    TUser, TKey, IdentityUserClaim<TKey>, IdentityUserLogin<TKey>,
    IdentityUserToken<TKey>>
        where TUser : IdentityUser<TKey>
        where TKey : IEquatable<TKey>
{
}

// No built-in Identity types are used; all are specified by generic arguments, with no roles
// The key type is defined by TKey
public abstract class IdentityUserContext<
    TUser, TKey, TUserClaim, TUserLogin, TUserToken> : DbContext
        where TUser : IdentityUser<TKey>
        where TKey : IEquatable<TKey>
```

```
        where TUserClaim : IdentityUserClaim<TKey>
        where TUserLogin : IdentityUserLogin<TKey>
        where TUserToken : IdentityUserToken<TKey>
    {
    }
```

Настройка модели. Отправной точкой настройки модели является наследование от соответствующего типа контекста. См. раздел "Универсальные типы модели ". Этот тип контекста обычно вызывается ApplicationDbContext и создается шаблонами ASP.NET Core.

Контекст используется для настройки модели двумя способами:

• Предоставление типов сущностей и ключей для параметров универсального типа.

• Переопределение OnModelCreating для изменения сопоставления этих типов.

При переопределении OnModelCreatingbase.OnModelCreating сначала следует вызывать конфигурацию переопределения. EF Core Как правило, для конфигурации имеется политика последней победы. Например, если ToTable метод для типа сущности вызывается сначала с одним именем таблицы, а затем позже с другим именем таблицы используется имя таблицы во втором вызове.

Пользовательские данные пользователя. Пользовательские данные поддерживаются путем наследования от IdentityUser. Это обычно называется следующим типом ApplicationUser:

C#Копировать

```
public class ApplicationUser : IdentityUser
{
   public string CustomTag { get; set; }
}
```

ApplicationUser Используйте тип в качестве универсального аргумента для контекста:

C#Копировать

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
   public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
```

```
        : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
    }
}
```

В классе не нужно переопределяться OnModelCreatingApplicationDbContext . EF CoreCustomTag сопоставляет свойство по соглашению. Однако для создания нового CustomTag столбца необходимо обновить базу данных. Чтобы создать столбец, добавьте миграцию, а затем обновите базу данных, как описано в Identity разделе " EF Core Миграция".

Обновление Pages/Shared/_LoginPartial.cshtml и замена IdentityUser на ApplicationUser:

CSHTMLКопировать

```
@using Microsoft.AspNetCore.Identity
@using WebApp1.Areas.Identity.Data
@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager
```

Обновление Areas/Identity/IdentityHostingStartup.cs или Startup.ConfigureServices замена IdentityUser на ApplicationUser.

C#Копировать

```
services.AddDefaultIdentity<ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddEntityFrameworkStores<ApplicationDbContext>();
```

Вызов AddDefaultIdentity эквивалентен следующему коду:

C#Копировать

```
services.AddAuthentication(o =>
{
    o.DefaultScheme = IdentityConstants.ApplicationScheme;
    o.DefaultSignInScheme = IdentityConstants.ExternalScheme;
})
```

```
.AddIdentityCookies(o => { });

services.AddIdentityCore<TUser>(o =>
{
   o.Stores.MaxLengthForKeys = 128;
   o.SignIn.RequireConfirmedAccount = true;
})
.AddDefaultUI()
.AddDefaultTokenProviders();
```

Identity предоставляется в виде библиотеки Razor классов. Дополнительные сведения см. в разделе Шаблоны Identity в проектах ASP.NET Core. Следовательно, для предыдущего кода требуется AddDefaultUIвызов . Identity Если шаблон использовался для добавления Identity файлов в проект, удалите вызовAddDefaultUI.

Изменение типа первичного ключа. Изменение типа данных столбца PK после создания базы данных проблематично во многих системах баз данных. Изменение PK обычно включает удаление и повторное создание таблицы. Поэтому при создании базы данных необходимо указать типы ключей при первоначальной миграции.

Чтобы изменить тип PK, выполните следующие действия.

1. Если база данных была создана до изменения PK, выполните команду Drop-Database (PMC) или dotnet ef database drop (.NET Core CLI), чтобы удалить ее.

2. После подтверждения удаления базы данных удалите начальную миграцию с Remove-Migration помощью (PMC) или dotnet ef migrations remove (.NET Core CLI).

3. Обновите класс, производный ApplicationDbContext от IdentityDbContext<TUser,TRole,TKey>. Укажите новый тип ключа для TKey. Например, чтобы использовать Guid тип ключа:

C#Копировать

```
public class ApplicationDbContext
   : IdentityDbContext<IdentityUser<Guid>, IdentityRole<Guid>, Guid>
{
   public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
```

```
        : base(options)
    {
    }
}
```

В приведенном выше коде универсальные классы IdentityUser<TKey> и IdentityRole<TKey> должны быть указаны для использования нового типа ключа.

Startup.ConfigureServices необходимо обновить для использования универсального пользователя:

C#Копировать

```
services.AddDefaultIdentity<IdentityUser<Guid>>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddEntityFrameworkStores<ApplicationDbContext>();
```

4. Если используется пользовательский ApplicationUser класс, обновите класс, наследуемый от IdentityUser. Пример:

C#Копировать

```
using System;
using Microsoft.AspNetCore.Identity;

public class ApplicationUser : IdentityUser<Guid>
{
    public string CustomTag { get; set; }
}
```

Обновите ApplicationDbContext ссылку на пользовательский ApplicationUser класс:

C#Копировать

```
public class ApplicationDbContext
    : IdentityDbContext<ApplicationUser, IdentityRole<Guid>, Guid>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }
}
```

Зарегистрируйте пользовательский класс контекста базы данных при добавлении Identity службы в Startup.ConfigureServices:

C#Копировать

```
services.AddDefaultIdentity<ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddEntityFrameworkStores<ApplicationDbContext>();
```

Тип данных первичного ключа выводится путем анализа DbContext объекта.

Identity предоставляется в виде библиотеки Razor классов. Дополнительные сведения см. в разделе Шаблоны Identity в проектах ASP.NET Core. Следовательно, для предыдущего кода требуется AddDefaultUIвызов . Identity Если шаблон использовался для добавления Identity файлов в проект, удалите вызов AddDefaultUI.

5. Если используется пользовательский ApplicationRole класс, обновите класс, наследуемый от IdentityRole<TKey>. Пример:

C#Копировать

```
using System;
using Microsoft.AspNetCore.Identity;

public class ApplicationRole : IdentityRole<Guid>
{
   public string Description { get; set; }
}
```

Обновление ApplicationDbContext для ссылки на пользовательский ApplicationRole класс. Например, следующий класс ссылается на пользовательский ApplicationUser и настраиваемый ApplicationRole:

C#Копировать

```
using System;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext :
   IdentityDbContext<ApplicationUser, ApplicationRole, Guid>
{
   public    ApplicationDbContext(DbContextOptions<ApplicationDbContext>
```

```
options)
        : base(options)
    {
    }
}
```

Зарегистрируйте пользовательский класс контекста базы данных при добавлении Identity службы в Startup.ConfigureServices:

C#Копировать

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });

    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));

    services.AddIdentity<ApplicationUser, ApplicationRole>()
            .AddEntityFrameworkStores<ApplicationDbContext>()
            .AddDefaultUI()
            .AddDefaultTokenProviders();

    services.AddMvc()
            .SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```

Тип данных первичного ключа выводится путем анализа DbContext объекта.

Identity предоставляется в виде библиотеки Razor классов. Следовательно, для предыдущего кода требуется AddDefaultUIвызов . Identity Если шаблон использовался для добавления Identity файлов в проект, удалите вызовAddDefaultUI.

Добавление свойств навигации. Изменение конфигурации модели для

связей может быть сложнее, чем внесение других изменений. Необходимо соблюдать осторожность, чтобы заменить существующие связи, а не создавать новые, дополнительные связи. В частности, измененная связь должна указывать то же свойство внешнего ключа (FK), что и существующее отношение. Например, связь между Users и UserClaims по умолчанию указана следующим образом:

C#Копировать

```
builder.Entity<TUser>(b =>
{
   // Each User can have many UserClaims
   b.HasMany<TUserClaim>()
    .WithOne()
    .HasForeignKey(uc => uc.UserId)
    .IsRequired();
});
```

FK для этой связи указывается в качестве UserClaim.UserId свойства. HasMany и WithOne вызываются без аргументов для создания связи без свойств навигации.

Добавьте свойство навигации, в ApplicationUser которое можно UserClaims ссылаться от пользователя:

C#Копировать

```
public class ApplicationUser : IdentityUser
{
   public virtual ICollection<IdentityUserClaim<string>> Claims { get; set; }
}
```

IdentityUserClaim<TKey> For TKey — это тип, указанный для PK пользователей. В этом случае TKey используются string значения по умолчанию. Это не тип PK для типа сущности UserClaim .

Теперь, когда свойство навигации существует, его необходимо настроить в следующих элементах OnModelCreating:

C#Копировать

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
   public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
```

```
        : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<ApplicationUser>(b =>
        {
            // Each User can have many UserClaims
            b.HasMany(e => e.Claims)
                .WithOne()
                .HasForeignKey(uc => uc.UserId)
                .IsRequired();
        });
    }
}
```

Обратите внимание, что связь настроена точно так же, как и раньше, только со свойством навигации, указанным в вызове HasMany.

Свойства навигации существуют только в модели EF, а не в базе данных. Так как FK для связи не изменился, такое изменение модели не требует обновления базы данных. Это можно проверить, добавив миграцию после внесения изменений. Методы Up пусты Down .

Добавление всех свойств навигации пользователя. Используя приведенный выше раздел в качестве руководства, в следующем примере настраивается однонаправленная навигация для всех связей пользователя:

C#Копировать

```
public class ApplicationUser : IdentityUser
{
    public virtual ICollection<IdentityUserClaim<string>> Claims { get; set; }
    public virtual ICollection<IdentityUserLogin<string>> Logins { get; set; }
    public virtual ICollection<IdentityUserToken<string>> Tokens { get; set; }
    public virtual ICollection<IdentityUserRole<string>> UserRoles { get; set; }
}
```

C#Копировать

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
        : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<ApplicationUser>(b =>
        {
            // Each User can have many UserClaims
            b.HasMany(e => e.Claims)
                .WithOne()
                .HasForeignKey(uc => uc.UserId)
                .IsRequired();

            // Each User can have many UserLogins
            b.HasMany(e => e.Logins)
                .WithOne()
                .HasForeignKey(ul => ul.UserId)
                .IsRequired();

            // Each User can have many UserTokens
            b.HasMany(e => e.Tokens)
                .WithOne()
                .HasForeignKey(ut => ut.UserId)
                .IsRequired();

            // Each User can have many entries in the UserRole join table
            b.HasMany(e => e.UserRoles)
```

```
            .WithOne()
            .HasForeignKey(ur => ur.UserId)
            .IsRequired();
        });
    }
}
```

Добавление свойств навигации пользователей и ролей. Используя приведенный выше раздел в качестве руководства, в следующем примере настраивается свойства навигации для всех связей с пользователем и ролью:

C#Копировать

```
public class ApplicationUser : IdentityUser
{
    public virtual ICollection<IdentityUserClaim<string>> Claims { get; set; }
    public virtual ICollection<IdentityUserLogin<string>> Logins { get; set; }
    public virtual ICollection<IdentityUserToken<string>> Tokens { get; set; }
    public virtual ICollection<ApplicationUserRole> UserRoles { get; set; }
}

public class ApplicationRole : IdentityRole
{
    public virtual ICollection<ApplicationUserRole> UserRoles { get; set; }
}

public class ApplicationUserRole : IdentityUserRole<string>
{
    public virtual ApplicationUser User { get; set; }
    public virtual ApplicationRole Role { get; set; }
}
```

C#Копировать

```
public class ApplicationDbContext
    : IdentityDbContext<
        ApplicationUser, ApplicationRole, string,
        IdentityUserClaim<string>, ApplicationUserRole,
IdentityUserLogin<string>,
        IdentityRoleClaim<string>, IdentityUserToken<string>>
```

```
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
            : base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<ApplicationUser>(b =>
            {
                // Each User can have many UserClaims
                b.HasMany(e => e.Claims)
                    .WithOne()
                    .HasForeignKey(uc => uc.UserId)
                    .IsRequired();

                // Each User can have many UserLogins
                b.HasMany(e => e.Logins)
                    .WithOne()
                    .HasForeignKey(ul => ul.UserId)
                    .IsRequired();

                // Each User can have many UserTokens
                b.HasMany(e => e.Tokens)
                    .WithOne()
                    .HasForeignKey(ut => ut.UserId)
                    .IsRequired();

                // Each User can have many entries in the UserRole join table
                b.HasMany(e => e.UserRoles)
                    .WithOne(e => e.User)
                    .HasForeignKey(ur => ur.UserId)
```

```
            .IsRequired();
        });

        modelBuilder.Entity<ApplicationRole>(b =>
        {
            // Each Role can have many entries in the UserRole join table
            b.HasMany(e => e.UserRoles)
                .WithOne(e => e.Role)
                .HasForeignKey(ur => ur.RoleId)
                .IsRequired();
        });

    }
}
```

Добавление всех свойств навигации. Используя приведенный выше раздел в качестве руководства, в следующем примере настраивается свойства навигации для всех связей для всех типов сущностей:

C#Копировать

```
public class ApplicationUser : IdentityUser
{
    public virtual ICollection<ApplicationUserClaim> Claims { get; set; }
    public virtual ICollection<ApplicationUserLogin> Logins { get; set; }
    public virtual ICollection<ApplicationUserToken> Tokens { get; set; }
    public virtual ICollection<ApplicationUserRole> UserRoles { get; set; }
}

public class ApplicationRole : IdentityRole
{
    public virtual ICollection<ApplicationUserRole> UserRoles { get; set; }
    public virtual ICollection<ApplicationRoleClaim> RoleClaims { get; set; }
}

public class ApplicationUserRole : IdentityUserRole<string>
{
```

```
    public virtual ApplicationUser User { get; set; }
    public virtual ApplicationRole Role { get; set; }
}

public class ApplicationUserClaim : IdentityUserClaim<string>
{
    public virtual ApplicationUser User { get; set; }
}

public class ApplicationUserLogin : IdentityUserLogin<string>
{
    public virtual ApplicationUser User { get; set; }
}

public class ApplicationRoleClaim : IdentityRoleClaim<string>
{
    public virtual ApplicationRole Role { get; set; }
}

public class ApplicationUserToken : IdentityUserToken<string>
{
    public virtual ApplicationUser User { get; set; }
}
```

C#Копировать

```
public class ApplicationDbContext
    : IdentityDbContext<
        ApplicationUser, ApplicationRole, string,
        ApplicationUserClaim, ApplicationUserRole, ApplicationUserLogin,
        ApplicationRoleClaim, ApplicationUserToken>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
        : base(options)
    {
    }
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<ApplicationUser>(b =>
    {
        // Each User can have many UserClaims
        b.HasMany(e => e.Claims)
            .WithOne(e => e.User)
            .HasForeignKey(uc => uc.UserId)
            .IsRequired();

        // Each User can have many UserLogins
        b.HasMany(e => e.Logins)
            .WithOne(e => e.User)
            .HasForeignKey(ul => ul.UserId)
            .IsRequired();

        // Each User can have many UserTokens
        b.HasMany(e => e.Tokens)
            .WithOne(e => e.User)
            .HasForeignKey(ut => ut.UserId)
            .IsRequired();

        // Each User can have many entries in the UserRole join table
        b.HasMany(e => e.UserRoles)
            .WithOne(e => e.User)
            .HasForeignKey(ur => ur.UserId)
            .IsRequired();
    });

    modelBuilder.Entity<ApplicationRole>(b =>
    {
        // Each Role can have many entries in the UserRole join table
```

```
            b.HasMany(e => e.UserRoles)
                .WithOne(e => e.Role)
                .HasForeignKey(ur => ur.RoleId)
                .IsRequired();

            // Each Role can have many associated RoleClaims
            b.HasMany(e => e.RoleClaims)
                .WithOne(e => e.Role)
                .HasForeignKey(rc => rc.RoleId)
                .IsRequired();
        });
    }
}
```

Использование составных ключей. В предыдущих разделах показано изменение типа ключа, используемого в Identity модели. Изменение ключевой Identity модели для использования составных ключей не поддерживается или не рекомендуется. Использование составного ключа с Identity включением изменения взаимодействия Identity кода диспетчера с моделью. Эта настройка выходит за рамки этого документа.

Изменение имен таблиц и столбцов и аспектов. Чтобы изменить имена таблиц и столбцов, вызовите .base.OnModelCreating Затем добавьте конфигурацию, чтобы переопределить любой из значений по умолчанию. Например, чтобы изменить имя всех Identity таблиц:

C#Копировать

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<IdentityUser>(b =>
    {
        b.ToTable("MyUsers");
    });

    modelBuilder.Entity<IdentityUserClaim<string>>(b =>
    {
```

```
        b.ToTable("MyUserClaims");
    });

    modelBuilder.Entity<IdentityUserLogin<string>>(b =>
    {
        b.ToTable("MyUserLogins");
    });

    modelBuilder.Entity<IdentityUserToken<string>>(b =>
    {
        b.ToTable("MyUserTokens");
    });

    modelBuilder.Entity<IdentityRole>(b =>
    {
        b.ToTable("MyRoles");
    });

    modelBuilder.Entity<IdentityRoleClaim<string>>(b =>
    {
        b.ToTable("MyRoleClaims");
    });

    modelBuilder.Entity<IdentityUserRole<string>>(b =>
    {
        b.ToTable("MyUserRoles");
    });
}
```

В этих примерах используются типы по умолчанию Identity . При использовании типа приложения, например ApplicationUser, настройте этот тип вместо типа по умолчанию.

В следующем примере изменяются некоторые имена столбцов:

C#Копировать

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
```

```
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<IdentityUser>(b =>
    {
        b.Property(e => e.Email).HasColumnName("EMail");
    });

    modelBuilder.Entity<IdentityUserClaim<string>>(b =>
    {
        b.Property(e => e.ClaimType).HasColumnName("CType");
        b.Property(e => e.ClaimValue).HasColumnName("CValue");
    });
}
```

Некоторые типы столбцов базы данных можно настроить с определенными аспектами (например, максимально допустимой string длиной). В следующем примере задается максимальная длина столбца для нескольких string свойств в модели:

C#Копировать

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<IdentityUser>(b =>
    {
        b.Property(u => u.UserName).HasMaxLength(128);
        b.Property(u => u.NormalizedUserName).HasMaxLength(128);
        b.Property(u => u.Email).HasMaxLength(128);
        b.Property(u => u.NormalizedEmail).HasMaxLength(128);
    });

    modelBuilder.Entity<IdentityUserToken<string>>(b =>
    {
        b.Property(t => t.LoginProvider).HasMaxLength(128);
        b.Property(t => t.Name).HasMaxLength(128);
    });
```

```
}
```

Сопоставление с другой схемой. Схемы могут вести себя по-разному в разных поставщиках баз данных. Для SQL Server по умолчанию создается все таблицы в схеме dbo. Таблицы можно создать в другой схеме. Пример:

```
C#Копировать
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
   base.OnModelCreating(modelBuilder);

   modelBuilder.HasDefaultSchema("notdbo");
}
```

Отложенная загрузка. В этом разделе добавлена поддержка отложенных прокси-серверов в Identity модели. Отложенная загрузка полезна, так как она позволяет использовать свойства навигации без предварительной загрузки.

Типы сущностей можно использовать для отложенной загрузки несколькими способами, как описано в документацииEF Core. Для простоты используйте отложенные прокси-серверы, которые требуют:

- Установка пакета Microsoft.EntityFrameworkCore.Proxies .
- Вызов внутри UseLazyLoadingProxiesAddDbContext.
- Общедоступные типы сущностей со свойствами public virtual навигации.

В следующем примере демонстрируется вызов UseLazyLoadingProxies:Startup.ConfigureServices

```
C#Копировать
services
   .AddDbContext<ApplicationDbContext>(
      b => b.UseSqlServer(connectionString)
           .UseLazyLoadingProxies())
   .AddDefaultIdentity<ApplicationUser>()
   .AddEntityFrameworkStores<ApplicationDbContext>();
```

5.4. Добавление, загрузка и удаление пользовательских данных в Identity в проекте ASP.NET Core

Пример проекта создается из веб-приложения Razor Pages, но инструкции

аналогичны для веб-приложения ASP.NET Core MVC.

Необходимые условия

Пакет SDK для .NET 6.0

Создание веб-приложения Razor

- Visual Studio
- Интерфейс командной строки .NET Core
- В меню Файл Visual Studio выберите Команду Создать проект >. Присвойте проекту имя WebApp1, если вы хотите, чтобы он соответствовал пространству имен загружаемого примера кода.
- Выберите ASP.NET основное веб-приложение > ОК
- Выберите Веб-приложение > ОК
- Выполните сборку и запустите проект.

Запуск скаффапола удостоверений

- Visual Studio
- Интерфейс командной строки .NET Core
- В обозревателе решений щелкните правой кнопкой мыши проект > Добавить > новый элемент формирования шаблонов.
- В левой области диалогового окна Добавление шаблона выберите Удостоверение > Добавить.
- В диалоговом окне Добавление удостоверения используются следующие параметры:
  - Выберите существующий файл макета ~/Pages/Shared/_Layout.cshtml
  - Выберите следующие файлы для переопределения:
    - Учетная запись/Регистрация
    - Аккаунт/Управление/Индекс
  - Нажмите кнопку +, чтобы создать новый класс контекста данных. Примите тип (WebApp1.Models.WebApp1Context, если проект называется WebApp1).
  - Нажмите кнопку +, чтобы создать новый класс User. Примите тип (WebApp1User, если проект называется WebApp1) > Add.
- Выберите Добавить.

Использование Аутентификация и макет, чтобы выполнить следующие действия.

- Создайте миграцию и обновите базу данных.
- Добавить в программу .csUseAuthentication

- Добавьте в файл макета. <partial name="_LoginPartial" />
- Протестируйте приложение:

o Регистрация пользователя

o Выберите новое имя пользователя (рядом со ссылкой Выход). Возможно, потребуется развернуть окно или выбрать значок панели навигации, чтобы отобразить имя пользователя и другие ссылки.

o Выберите вкладку Личные данные.

o Нажмите кнопку Загрузить и изучите файл.PersonalData.json

o Проверьте кнопку Удалить, которая удаляет вошедшего в систему пользователя.

Добавление пользовательских данных в базу данных удостоверений. Обновите производный класс пользовательскими свойствами. Если проект назван WebApp1, файл будет называться . Обновите файл следующим кодом:IdentityUserAreas/Identity/Data/WebApp1User.cs

C #Копировать

```
using Microsoft.AspNetCore.Identity;

namespace WebApp1.Areas.Identity.Data;

public class WebApp1User : IdentityUser
{
   [PersonalData]
   public string ? Name { get; set; }
   [PersonalData]
   public DateTime DOB { get; set; }
}
```

Свойства с атрибутом PersonalData:

- Удаляется при вызове Razor Page .Areas/Identity/Pages/Account/Manage/DeletePersonalData.cshtmlUserManager.Delete
- Включено в данные, загруженные страницей Razor.Areas/Identity/Pages/Account/Manage/DownloadPersonalData.cshtml

Обновление страницы Account/Manage/Index.cshtml. Обновите вход следующим выделенным кодом: InputModelAreas/Identity/Pages/Account/Manage/Index.cshtml.cs

```
C #Копировать
public class IndexModel : PageModel
{
    private readonly UserManager<WebApp1User> _userManager;
    private readonly SignInManager<WebApp1User> _signInManager;

    public IndexModel(
        UserManager<WebApp1User> userManager,
        SignInManager<WebApp1User> signInManager)
    {
        _userManager = userManager;
        _signInManager = signInManager;
    }

    /// <summary>
    ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
    ///     directly from your code. This API may change or be removed in future releases.
    /// </summary>
    public string Username { get; set; }

    // Remaining API warnings ommited.

    [TempData]
    public string StatusMessage { get; set; }

    [BindProperty]
    public InputModel Input { get; set; }

    public class InputModel
    {
        [Обязательно]
  [Тип данных(Тип данных.Текст)]
  [Отображение(Имя = "Полное имя")] публичная строка Name { get;
```

```
комплект; }

    [Обязательно]
     [Отображение(Имя = "Дата рождения")]
     [Тип данных(Тип данных.Дата)] общественный DateTime DOB { get; набор; }

            [Phone]
            [Display(Name = "Phone number")]
            public string PhoneNumber { get; set; }
        }

        private async Task LoadAsync(WebApp1User user)
        {
            var userName = await _userManager.GetUserNameAsync(user);
            var phoneNumber = await _userManager.GetPhoneNumberAsync(user);

            Username = userName;

            Input = new InputModel
            {
      Имя = пользователь. Имя
      DOB = пользователь. Дата рождения,
                PhoneNumber = phoneNumber
            };
        }

        public async Task<IActionResult> OnGetAsync()
        {
            var user = await _userManager.GetUserAsync(User);
            if (user == null)
            {
                return NotFound($"Unable to load user with ID '{_userManager.GetUserId(User)}'.");
            }
```

```
            await LoadAsync(user);
            return Page();
        }

        public async Task<IActionResult> OnPostAsync()
        {
            var user = await _userManager.GetUserAsync(User);
            if (user == null)
            {
                return NotFound($"Unable to load user with ID
'{_userManager.GetUserId(User)}'.");
            }

            if (!ModelState.IsValid)
            {
                await LoadAsync(user);
                return Page();
            }

            var phoneNumber = await _userManager.GetPhoneNumberAsync(user);
            if (Input.PhoneNumber != phoneNumber)
            {
                var setPhoneResult = await _userManager.SetPhoneNumberAsync(user,
Input.PhoneNumber);
                if (!setPhoneResult.Succeeded)
                {
                    StatusMessage = "Unexpected error when trying to set phone
number.";
                    return RedirectToPage();
                }
            }

            if (Input.Name != пользователь. Имя)
    {
```

```
пользователь. Имя = Input.Name;
} if (Input.DOB != user. Дата рождения)
{
пользователь. DOB = Вход.DOB;
} await _userManager.UpdateAsync(user);
      await _signInManager.RefreshSignInAsync(user);
      StatusMessage = "Your profile has been updated";
      return RedirectToPage();
    }
  }
```

Обновите следующую выделенную разметку: Areas/Identity/Pages/Account/Manage/Index.cshtml

CSHTMLКопировать

```
@page
@model IndexModel
@{
  ViewData["Title"] = "Profile";
  ViewData["ActivePage"] = ManageNavPages.Index;
}

<h3>@ViewData["Title"]</h3>
<partial name="_StatusMessage" for="StatusMessage" />
<div class="row">
  <div class="col-md-6">
    <form id="profile-form" method="post">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-floating">
        <input asp-for="Username" class="form-control" disabled />
        <label asp-for="Username" class="form-label"></label>
      </div>
 <div class="form-group"> <label asp-for="Input.Name" class="form-control"></label> <input asp-for="Input.Name" class="form-label" /> </div> <div class="form-group">  <label asp-for="Input.DOB" class="form-control"></label> <input asp-for="Input.DOB" class="form-label" /> </div>
```

```
            <div class="form-floating">
                <input asp-for="Input.PhoneNumber" class="form-control" />
                <label asp-for="Input.PhoneNumber" class="form-label"></label>
                <span asp-validation-for="Input.PhoneNumber" class="text-danger"></span>
            </div>
            <button id="update-profile-button" type="submit" class="w-100 btn btn-lg btn-primary">Save</button>
        </form>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Обновление страницыAccount/Register.cshtml. Обновите вход следующим выделенным кодом: InputModelAreas/Identity/Pages/Account/Register.cshtml.cs

C #Копировать

```
public class RegisterModel : PageModel
{
    private readonly SignInManager<WebApp1User> _signInManager;
    private readonly UserManager<WebApp1User> _userManager;
    private readonly IUserStore<WebApp1User> _userStore;
    private readonly IUserEmailStore<WebApp1User> _emailStore;
    private readonly ILogger<RegisterModel> _logger;
    private readonly IEmailSender _emailSender;

    public RegisterModel(
        UserManager<WebApp1User> userManager,
        IUserStore<WebApp1User> userStore,
        SignInManager<WebApp1User> signInManager,
        ILogger<RegisterModel> logger,
        IEmailSender emailSender)
    {
        _userManager = userManager;
```

```
            _userStore = userStore;
            _emailStore = GetEmailStore();
            _signInManager = signInManager;
            _logger = logger;
            _emailSender = emailSender;
        }

        /// <summary>
        ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
        ///     directly from your code. This API may change or be removed in future releases.
        /// </summary>
        [BindProperty]
        public InputModel Input { get; set; }

        // Remaining API warnings ommited.
        public string ReturnUrl { get; set; }

        public IList<AuthenticationScheme> ExternalLogins { get; set; }

        public class InputModel
        {
            [Обязательно]
    [Тип данных(Тип данных.Текст)]
    [Отображение(Имя = "Полное имя")] публичная строка Name { get; комплект; }

    [Обязательно]
    [Отображение(Имя = "Дата рождения")]
    [Тип данных(Тип данных.Дата)] общественный DateTime DOB { get; набор; }

            [Required]
            [EmailAddress]
```

```
            [Display(Name = "Email")]
            public string Email { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at
max {1} characters long.", MinimumLength = 6)]
            [DataType(DataType.Password)]
            [Display(Name = "Password")]
            public string Password { get; set; }

            [DataType(DataType.Password)]
            [Display(Name = "Confirm password")]
            [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
            public string ConfirmPassword { get; set; }
        }


        public async Task OnGetAsync(string returnUrl = null)
        {
            ReturnUrl = returnUrl;
            ExternalLogins                    =                    (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
        }

        public async Task<IActionResult> OnPostAsync(string returnUrl = null)
        {
            returnUrl ??= Url.Content("~/");
            ExternalLogins                    =                    (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
            if (ModelState.IsValid)
            {
                var user = CreateUser();

    пользователь. Имя = Input.Name;
```

```
пользователь. DOB = Вход.DOB;

                await _userStore.SetUserNameAsync(user, Input.Email, CancellationToken.None);
                await _emailStore.SetEmailAsync(user, Input.Email, CancellationToken.None);
                var result = await _userManager.CreateAsync(user, Input.Password);

                if (result.Succeeded)
                {
                    _logger.LogInformation("User created a new account with password.");

                    var userId = await _userManager.GetUserIdAsync(user);
                    var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
                    code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
                    var callbackUrl = Url.Page(
                        "/Account/ConfirmEmail",
                        pageHandler: null,
                        values: new { area = "Identity", userId = userId, code = code, returnUrl = returnUrl },
                        protocol: Request.Scheme);

                    await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
                        $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

                    if (_userManager.Options.SignIn.RequireConfirmedAccount)
                    {
                        return RedirectToPage("RegisterConfirmation", new { email = Input.Email, returnUrl = returnUrl });
                    }
```

```
                else
                {
                    await _signInManager.SignInAsync(user, isPersistent: false);
                    return LocalRedirect(returnUrl);
                }
            }
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }

        // If we got this far, something failed, redisplay form
        return Page();
    }

    private WebApp1User CreateUser()
    {
        try
        {
            return Activator.CreateInstance<WebApp1User>();
        }
        catch
        {
            throw new InvalidOperationException($"Can't create an instance of '{nameof(WebApp1User)}'. " +
                $"Ensure that '{nameof(WebApp1User)}' is not an abstract class and has a parameterless constructor, or alternatively " +
                $"override the register page in /Areas/Identity/Pages/Account/Register.cshtml");
        }
    }

    private IUserEmailStore<WebApp1User> GetEmailStore()
    {
```

```
            if (!_userManager.SupportsUserEmail)
            {
                throw new NotSupportedException("The default UI requires a user store with email support.");
            }
            return (IUserEmailStore<WebApp1User>)_userStore;
        }
    }
}
```

Обновите следующую выделенную разметку: Areas/Identity/Pages/Account/Register.cshtml

CSHTMLКопировать

```
@page
@model RegisterModel
@{
    ViewData["Title"] = "Register";
}

<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-4">
        <form id="registerForm" asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <h2>Create a new account.</h2>
            <hr />
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

 <div class="form-floating"> <label asp-for="Input.Name"></label> <input asp-for="Input.Name" class="form-control" /> <span asp-validation-for="Input.Name" class="text-danger"></ span> </div> <div class="form-floating"> <label asp-for="Input.DOB"></label> <input asp-for="Input.DOB" class="form-control" /> <span asp-validation-for="Input.DOB" class="text-danger"></span> </div>
```

```
            <div class="form-floating">
                <input asp-for="Input.Email" class="form-control" autocomplete="username" aria-required="true" />
                <label asp-for="Input.Email"></label>
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
            <div class="form-floating">
                <input asp-for="Input.Password" class="form-control" autocomplete="new-password" aria-required="true" />
                <label asp-for="Input.Password"></label>
                <span asp-validation-for="Input.Password" class="text-danger"></span>
            </div>
            <div class="form-floating">
                <input asp-for="Input.ConfirmPassword" class="form-control" autocomplete="new-password" aria-required="true" />
                <label asp-for="Input.ConfirmPassword"></label>
                <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
            </div>
            <button id="registerSubmit" type="submit" class="w-100 btn btn-lg btn-primary">Register</button>
        </form>
    </div>
    <div class="col-md-6 col-md-offset-2">
        <section>
            <h3>Use another service to register.</h3>
            <hr />
            @{
                if ((Model.ExternalLogins?.Count ?? 0) == 0)
                {
                    <div>
                        <p>
                            There are no external authentication services configured. See
```

```
this <a href="https://go.microsoft.com/fwlink/?LinkID=532715">article
                    about setting up this ASP.NET application to support logging
in via external services</a>.
                </p>
            </div>
        }
        else
        {
            <form id="external-account" asp-page="./ExternalLogin" asp-
route-returnUrl="@Model.ReturnUrl" method="post" class="form-horizontal">
                <div>
                    <p>
                        @foreach (var provider in Model.ExternalLogins!)
                        {
                            <button type="submit" class="btn btn-primary"
name="provider" value="@provider.Name" title="Log in using your
@provider.DisplayName account">@provider.DisplayName</button>
                        }
                    </p>
                </div>
            </form>
        }
    }
</section>
</div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Постройте проект.

Обновление макета. Добавление миграции для пользовательских данных

- Visual Studio
- Интерфейс командной строки .NET Core

В консоли диспетчера пакетов Visual Studio:

Оболочка PowerShellКопировать

Add-Migration CustomUserData

Update-Database

Тестирование создания, просмотра, загрузки, удаления пользовательских данных.

Протестируйте приложение:

- Зарегистрируйте нового пользователя.
- Просмотр пользовательских данных на странице./Identity/Account/Manage
- Скачивайте и просматривайте персональные данные пользователей со страницы./Identity/Account/Manage/PersonalData

5.5. Подтверждение учетной записи и восстановление пароля в ASP.NET Core

Необходимые условия

- Пакет SDK для .NET Core 6.0 или более поздней версии
- Успешная отправка электронной почты из консольного приложения C#.

Создание и тестирование веб-приложения с проверкой подлинности. Выполните следующие команды, чтобы создать веб-приложение с проверкой подлинности.

Интерфейс командной строки .NETКопировать

dotnet new webapp -au Individual -o WebPWrecover

cd WebPWrecover

dotnet run

Зарегистрируйте пользователя с помощью имитации подтверждения по электронной почте. Запустите приложение, выберите ссылку Регистрация и зарегистрируйте пользователя. После регистрации вы будете перенаправлены на страницу to, которая содержит ссылку для имитации подтверждения по электронной почте: /Identity/Account/RegisterConfirmation

- Выберите ссылку.Click here to confirm your account
- Выберите ссылку Вход и войдите в систему с теми же учетными данными.
- Выберите ссылку, которая перенаправляет на страницу.Hello YourEmail@provider.com!/Identity/Account/Manage/PersonalData
- Перейдите на вкладку Личные данные слева и нажмите кнопку Удалить.

Ссылка отображается, так как IEmailSender не был реализован и не зарегистрирован в контейнере внедрения каталога. См. источник RegisterConfirmation.Click here to confirm your account

Настройка поставщика услуг электронной почты. В этом учебнике SendGrid используется для отправки электронной почты. Для отправки электронной почты требуется учетная запись SendGrid и ключ. Другие поставщики услуг электронной почты. Мы рекомендуем использовать SendGrid или другую почтовую службу для отправки электронной почты, а не SMTP. SMTP трудно защитить и правильно настроить.

Учетная запись SendGrid может потребовать добавления отправителя.

Создайте класс для получения защищенного ключа электронной почты. Для этого примера создайте: Services/AuthMessageSenderOptions.cs

C #Копировать

```
namespace WebPWrecover.Services;

public class AuthMessageSenderOptions
{
   public string? SendGridKey { get; set; }
}
```

Настройка пользовательских секретов SendGrid. Установите его с помощью инструмента secret-manager. Например:SendGridKey

Интерфейс командной строки .NETКопировать

```
dotnet user-secrets set SendGridKey <key>
```

Successfully saved SendGridKey to the secret store.

В Windows Secret Manager хранит пары ключ/значение в файле в каталоге.secrets.json%APPDATA%/Microsoft/UserSecrets/<WebAppName-userSecretsId>

Содержимое файла не шифруется. В следующей разметке показан файл. Значение удалено.secrets.jsonsecrets.jsonSendGridKey

JSONКопировать

```
{
  "SendGridKey": "<key removed>"
}
```

Установка SendGrid. Из этого туториала Вы узнаете, как добавить

уведомления по электронной почте через SendGrid, но можно использовать и других поставщиков услуг электронной почты.

Установите пакет NuGet:SendGrid

- Visual Studio
- Интерфейс командной строки .NET Core

В консоли диспетчера пакетов введите следующую команду:

Оболочка PowerShellКопировать

Install-Package SendGrid

Реализация IEmailSender. Чтобы реализовать, создайте с кодом, похожим на следующий:IEmailSenderServices/EmailSender.cs

C #Копировать

```
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.Extensions.Options;
using SendGrid;
using SendGrid.Helpers.Mail;

namespace WebPWrecover.Services;

public class EmailSender : IEmailSender
{
   private readonly ILogger _logger;

   public EmailSender(IOptions<AuthMessageSenderOptions> optionsAccessor,
                      ILogger<EmailSender> logger)
   {
      Options = optionsAccessor.Value;
      _logger = logger;
   }

   public AuthMessageSenderOptions Options { get; } //Set with Secret Manager.

   public async Task SendEmailAsync(string toEmail, string subject, string message)
```

```
        {
            if (string.IsNullOrEmpty(Options.SendGridKey))
            {
                throw new Exception("Null SendGridKey");
            }
            await Execute(Options.SendGridKey, subject, message, toEmail);
        }

        public async Task Execute(string apiKey, string subject, string message, string
toEmail)
        {
            var client = new SendGridClient(apiKey);
            var msg = new SendGridMessage()
            {
                From = new EmailAddress("Joe@contoso.com", "Password Recovery"),
                Subject = subject,
                PlainTextContent = message,
                HtmlContent = message
            };
            msg.AddTo(new EmailAddress(toEmail));

            // Disable click tracking.
            // See https://sendgrid.com/docs/User_Guide/Settings/tracking.html
            msg.SetClickTracking(false, false);
            var response = await client.SendEmailAsync(msg);
            _logger.LogInformation(response.IsSuccessStatusCode
                                   ? $"Email to {toEmail} queued successfully!"
                                   : $"Failure Email to {toEmail}");
        }
    }
```

Настройка приложения для поддержки электронной почты. Добавьте в файл следующий код :Program.cs

- Добавить как временную службу.EmailSender
- Зарегистрируйте экземпляр конфигурации.AuthMessageSenderOptions

C #Копировать

```
using Microsoft.AspNetCore.Identity;
использование Microsoft.aspNetCore.Identity.UI.Services;
using Microsoft.EntityFrameworkCore;
using WebPWrecover.Data;
использование WebPWrecover.Сервисы;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddRazorPages();

строитель. Services.AddTransient<IEmailSender, EmailSender>();
строитель. Services.Configure<AuthMessageSenderOptions>(builder. Конфигурация);

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}
```

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapRazorPages();

app.Run();

Отключить проверку учетной записи по умолчанию, если шаблон Account.RegisterConfirmation был сформирован

Пользователь перенаправляется туда, где он может выбрать ссылку для подтверждения учетной записи. Значение по умолчанию используется только для тестирования, автоматическая проверка учетной записи должна быть отключена в рабочем приложении. Account.RegisterConfirmationAccount.RegisterConfirmation

Чтобы запросить подтвержденную учетную запись и предотвратить немедленный вход при регистрации, установите в файле scaffolded:DisplayConfirmAccountLink = false/Areas/Identity/Pages/Account/RegisterConfirmation.cshtml.cs

C #Копировать

// Licensed to the .NET Foundation under one or more agreements.
// The .NET Foundation licenses this file to you under the MIT license.
#nullable disable

using System;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

```
using Microsoft.AspNetCore.WebUtilities;

namespace WebPWrecover.Areas.Identity.Pages.Account
{
    [AllowAnonymous]
    public class RegisterConfirmationModel : PageModel
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly IEmailSender _sender;

        public RegisterConfirmationModel(UserManager<IdentityUser> userManager, IEmailSender sender)
        {
            _userManager = userManager;
            _sender = sender;
        }

        /// <summary>
        ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
        ///     directly from your code. This API may change or be removed in future releases.
        /// </summary>
        public string Email { get; set; }

        /// <summary>
        ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
        ///     directly from your code. This API may change or be removed in future releases.
        /// </summary>
        public bool DisplayConfirmAccountLink { get; set; }

        /// <summary>
        ///     This API supports the ASP.NET Core Identity default UI infrastructure
```

```
and is not intended to be used
        ///     directly from your code. This API may change or be removed in future releases.
        /// </summary>
        public string EmailConfirmationUrl { get; set; }

        public async Task<IActionResult> OnGetAsync(string email, string returnUrl = null)
        {
            if (email == null)
            {
                return RedirectToPage("/Index");
            }
            returnUrl = returnUrl ?? Url.Content("~/");

            var user = await _userManager.FindByEmailAsync(email);
            if (user == null)
            {
                return NotFound($"Unable to load user with email '{email}'.");
            }

            Email = email;
            // Once you add a real email sender, you should remove this code that lets you confirm the account
            DisplayConfirmAccountLink = false;
            if (DisplayConfirmAccountLink)
            {
                var userId = await _userManager.GetUserIdAsync(user);
                var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
                code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
                EmailConfirmationUrl = Url.Page(
                    "/Account/ConfirmEmail",
                    pageHandler: null,
```

```
                values: new { area = "Identity", userId = userId, code = code,
returnUrl = returnUrl },
                protocol: Request.Scheme);
            }

            return Page();
        }
    }
}
```

Регистрация, подтверждение адреса электронной почты и сброс пароля. Запустите веб-приложение и протестируйте процесс подтверждения учетной записи и восстановления пароля.

- Запустите приложение и зарегистрируйте нового пользователя
- Проверьте свою электронную почту на наличие ссылки для подтверждения учетной записи. См. раздел Отладка электронной почты, если вы не получили письмо.
- Нажмите на ссылку, чтобы подтвердить адрес электронной почты.
- Войдите в систему, используя свой адрес электронной почты и пароль.
- Выйдите из системы.

Тестовый сброс пароля.

- Если вы выполнили вход, выберите Выйти.
- Выберите ссылку Войти и выберите ссылку Забыли пароль?.
- Введите адрес электронной почты, который вы использовали для регистрации учетной записи.
- Будет отправлено письмо со ссылкой для сброса пароля. Проверьте свою электронную почту и нажмите на ссылку, чтобы сбросить пароль. После успешного сброса пароля вы можете войти в систему, используя свой адрес электронной почты и новый пароль.

Повторная отправка подтверждения по электронной почте. Выберите ссылку Повторно отправить подтверждение по электронной почте на странице Вход.

Изменение времени ожидания электронной почты и активности. По умолчанию время ожидания бездействия составляет 14 дней. Следующий код устанавливает время ожидания бездействия равным 5 дням:

C #Копировать

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.EntityFrameworkCore;
using WebPWrecover.Data;
using WebPWrecover.Services;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddRazorPages();

builder.Services.AddTransient<IEmailSender, EmailSender>();
builder.Services.Configure<AuthMessageSenderOptions>(builder.Configuration);

строитель. Services.ConfigureApplicationCookie(o => {
 o.ExpireTimeSpan = TimeSpan.FromDays(5);
 o.Скользящаяэкспирация = true;
});

var app = builder.Build();

// Code removed for brevity
```

Изменение сроков действия всех маркеров защиты данных. Следующий код изменяет время ожидания всех маркеров защиты данных на 3 часа:

C #Копировать

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.EntityFrameworkCore;
using WebPWrecover.Data;
using WebPWrecover.Services;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddRazorPages();

builder.Services.AddTransient<IEmailSender, EmailSender>();
builder.Services.Configure<AuthMessageSenderOptions>(builder.Configuration);

строитель. Services.Configure<DataProtectionTokenProviderOptions>(o =>
 o.TokenLifespan = TimeSpan.FromHours(3));

var app = builder.Build();

// Code removed for brevity.
```

Встроенные маркеры пользователя Identity (см. AspNetCore/src/Identity/Extensions.Core/src/TokenOptions.cs) имеют тайм-аут в один день.

Изменение срока службы маркера электронной почты. Срок действия маркера по умолчанию для маркеров пользователя Identity составляет один день. В этом разделе показано, как изменить срок службы маркера

электронной почты.

Добавьте пользовательский DataProtectorTokenProvider<TUser> и DataProtectionTokenProviderOptions:

C #Копировать

```
public class CustomEmailConfirmationTokenProvider<TUser>
                    :  DataProtectorTokenProvider<TUser> where TUser : class
{
   public CustomEmailConfirmationTokenProvider(
      IDataProtectionProvider dataProtectionProvider,
      IOptions<EmailConfirmationTokenProviderOptions> options,
      ILogger<DataProtectorTokenProvider<TUser>> logger)
                          : base(dataProtectionProvider, options, logger)
   {

   }
}
public class EmailConfirmationTokenProviderOptions : DataProtectionTokenProviderOptions
{
   public EmailConfirmationTokenProviderOptions()
   {
      Name = "EmailDataProtectorTokenProvider";
      TokenLifespan = TimeSpan.FromHours(4);
   }
}
```

Добавьте пользовательского поставщика в контейнер службы:

C #Копировать

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.EntityFrameworkCore;
using WebPWrecover.Data;
using WebPWrecover.Services;
using WebPWrecover.TokenProviders;

var builder = WebApplication.CreateBuilder(args);
```

```
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<IdentityUser>(config =>
{
    config.SignIn.RequireConfirmedEmail = true;
конфиг. Tokens.ProviderMap.Add("CustomEmailConfirmation", новый TokenProviderDescriptor(
typeof(CustomEmailConfirmationTokenProvider<IdentityUser>)));
конфиг. Tokens.EmailConfirmationTokenProvider = "CustomEmailConfirmation";
}). AddEntityFrameworkStores<ApplicationDbContext>();

строитель. Services.AddTransient<CustomEmailConfirmationTokenProvider<IdentityUser>>();

builder.Services.AddRazorPages();

builder.Services.AddTransient<IEmailSender, EmailSender>();
builder.Services.Configure<AuthMessageSenderOptions>(builder.Configuration);

var app = builder.Build();

// Code removed for brevity.
```

Отладка электронной почты. Если вы не можете заставить электронную почту работать:

- Установите точку останова для проверки вызова.EmailSender.ExecuteSendGridClient.SendEmailAsync
- Создайте консольное приложение для отправки электронной почты,

используя код, аналогичный коду .EmailSender.Execute

- Просмотрите страницу Активность электронной почты.
- Проверьте папку со спамом.
- Попробуйте использовать другой псевдоним электронной почты у другого поставщика услуг электронной почты (Microsoft, Yahoo, Gmail и т. д.)
- Попробуйте отправить сообщения на разные учетные записи электронной почты.

Рекомендуется не использовать секреты производства при тестировании и разработке. Если вы публикуете приложение в Azure, задайте секреты SendGrid в качестве параметров приложения на портале веб-приложений Azure. Система конфигурации настроена на чтение ключей из переменных среды.

Объединение социальных и локальных учетных записей. Вы можете объединить локальные и социальные учетные записи, нажав на ссылку электронной почты. В следующей последовательности "RickAndMSFT@gmail.com" сначала создается как локальный логин; однако вы можете сначала создать учетную запись в качестве входа в социальную сеть, а затем добавить локальный логин.

Нажмите на ссылку Управление. Обратите внимание на 0 внешних (социальных логинов), связанных с этой учетной записью.

Щелкните ссылку на другую службу входа и примите запросы приложения. На следующем рисунке Facebook является внешним поставщиком аутентификации:

Эти два счета были объединены. Вы можете войти в систему с помощью любой учетной записи. Возможно, вы захотите, чтобы ваши пользователи добавили локальные учетные записи в случае, если их служба аутентификации входа в социальную сеть отключена или, что более вероятно, они потеряли доступ к своей учетной записи в социальной сети.

Включение подтверждения учетной записи после того, как на сайте появятся пользователи. Включение подтверждения учетной записи на сайте с пользователями блокирует всех существующих пользователей. Существующие пользователи заблокированы, поскольку их учетные записи не

подтверждены. Чтобы обойти существующую блокировку пользователей, используйте один из следующих подходов:

• Обновите базу данных, чтобы пометить всех существующих пользователей как подтвержденных.

• Подтвердите существующих пользователей. Например, пакетная отправка электронных писем со ссылками для подтверждения.

Выводы по главе 6

Понятия «телекоммуникационные системы» и «информационной безопасности» тесно связаны. Телекоммуникационные системы представляют собой технические средства, предназначенные для передачи больших объемов информации через различные каналы линий связи. В зависимости от целей использования при оценке информации учитываются аспекты информационной безопасности. Информационная безопасность телекоммуникационных систем с десятками миллионов пользователей предполагает защиту локальных информационных систем и их баз данных эффективными системами авторизация и аутентификация пользователей.

В разделе рассмотрена одну из самых эффективных систем авторизация и аутентификация ASP.NET Identity, встроенную в ASP.NET.

Рассмотрены процедуры создания учетных записи, аутентифицированния, управления учетными записями. Определена технология доступа с использованием учетных записей внешних провайдеров, таких как Facebook, Google, Microsoft, Twitter и других.

Определена авторизация на основе ролей, Claim-Based авторизацию, аутентификацию через соцсети, двухфакторную аутентификацию с подтверждением по смс или по электронной почте и др.

Заключение

В результате исследования впервые были созданы программы для ЭВМ и базы данных с динамическим формированием навигации сайта и адаптивной настройкой поиска в базе данных с меньшими временными затратами. Построены адаптивные сайты типа 2.0 в среде ASP.NET MVC 5 для формирования моделей объектов внешней среды, их последующей обработки, виазуализации и принятия решений. Полученные результаты защищены свидетельствами о регистрации программ для ЭВМ и баз данных.

Результаты внедрены в учебный процесс Уральского государственноголесотехнического университета для направлений бакалавриата, магистратуры и аспирантуры и в управление предприятиями лесного сектора экономики.

Получены новые знания о фундаментальных закономерностях принятия решений в условиях неопределенности и построения адекватных моделей. Изучены структуры полученных моделей и формирования множества правил вывода, образующих базу знаний информационной системы.

Были разработаны: методика проектирования взаимосвязанных структур баз данных и знаний; новые технологии защиты сайта и базы данных, новые закономерности самонастраивающихся нечетких моделей для принятия решений; новое программное обеспечение в среде Visual Studio.

Полученные новые знания будут использоватья в образовательных программах обучения магистров направления "Управление качеством: направленность-производственный менеджмент в лесном секторе экономике". Полученные новые технологии и программные продукты будут использованы в образовательных программах обучения бакалавром и магистров по направлению "Прикладная информатика". Полученные новые результаты и знания будут использовать в образовательных программах обучения бакалавров и магистров по направлениям "Менеджмент" и "Бизнес информатика".

Список использованных источников

Часовских В.П., Лабунец В.Г., Федорова Т.С., Остхаймер Е. Семейство обобщенных преобразований хаара. Эко-потенциал. 2016. № 2 (14). С. 90-100.

Стаин Д.А., Часовских В.П. Модель образовательного процесса университета в среде технологии интернет. Вестник Московского государственного университета леса - Лесной вестник. 2016. Т. 20. № 2. С. 233-237.

Стаин Д.А., Часовских В.П. Исходные данные модели образовательного процесса вуза в среде современных web-технологий. Актуальные проблемы гуманитарных и естественных наук. 2016. № 1-5. С. 20-22.

Часовских В.П., Акчурина Г.А., Слободин А.В., Азаренок М.В., Воронов М.П., Кох Е.В., Анянова Е.В., Крайнова Т.С., Богословская О.А. Информационные технологии управления. Международный журнал экспериментального образования. 2016. № 6-1. С. 151-152.

Usoltsev V.A., Noritsina Yu.V., Noritsina Yu.V., Chasovskikh V.P. Modelli di regressione e tavoli per la stima della biomassa di struttura ad albero per il telerilevamento di pinete dell'eurasia. Italian Science Review. 2016. № 1 (34). С. 126-129.

Часовских В.П., Кох Е.В., Стаин Д.А. Синхронное и (или) асинхронное взаимодействия посредством сети «интернет» между участниками образовательного процесса в электронной информационно-образовательной среде вуза. Эко-потенциал. 2016. № 1 (13). С. 53-56.

Усольцев В.А., Норицина Ю.В., Норицин Д.В., Часовских В.П., Габделхаков А.К., Касаткин А.С., Жанабаева А.С. Аллометрические модели фитомассы деревьев лиственных пород евразии и перспективы их использования при дистанционном зондировании лесов. Эко-потенциал. 2016. № 1 (13). С. 7-19. 0

Часовских В.П., Лабунец В.Г., Комаров Д.Е., Остхаймер Е. Многопараметрические вейвлет-преобразования. Эко-потенциал. 2016. № 2 (14). С. 76-89.

Часовских В.П., Лабунец В.Г., Федорова Т.С., Остхаймер Е.Семейство обобщенных преобразований хаара. Эко-потенциал. 2016. № 2 (14). С. 90-100.

Лабунец В.Г., Часовских В.П., Остхаймер Е. Обобщенные классическая и квантовая теории сигналов на гипергруппах. часть 1. Классическая теория сигналов Эко-потенциал. 2016. № 3 (15). С. 56-64.

Лабунец В.Г., Часовских В.П., Остхаймер Е. Обобщенные классическая и квантовая теории сигналов на гипергруппах. часть 2. Квантовая теория сигналов. Эко-потенциал. 2016. № 3 (15). С. 65-71.

Jucker T., Wedeux B.M.M., Coomes D.A., Caspersen J., Haeni M., Waldner P., Zimmermann N.E., Chave J., Antin C., Barbier N., Momo S., Ploton P., Bongers F., Poorter L., Sterck F.J., Dalponte M., van Ewijk K.Y., Forrester D.I., Higgins S.I., Holdaway R.J. et al. Allometric

equations for integrating remote sensing imagery into forest monitoring programmes. Global Change Biology. 2016.

Usoltsev V.A., Noritsina Y.V., Chasovskikh V.P., Kokh E.V. Methods and results of studying the geographical trends in the structure of single-tree biomass of larches and two-needled pines in Eurasia. Russian Journal of Ecology. 2016. Т. 47. № 5. С. 442-452.

Usoltsev V.A., Noritsina Yu.V., Noritsina Yu.V., Chasovskikh V.P. Modelli di regressione e tavoli per la stima della biomassa di struttura ad albero per il telerilevamento di pinete dell'eurasia. Italian Science Review. 2016. № 1 (34). С. 126-129.

Воронов М.П., Часовских В.П. Модель структуры субд adabas как средство проектирования информационных экономических систем. В сборнике: Новые информационные технологии в науке нового времени сборник статей международной научно-практической конференции. 2016. С. 35-37.

Воронов М.П., Часовских В.П. Многоуровневая модель информационной системы коммерческого предприятия. В сборнике: Современное состояние и перспективы развития научной мысли сборник статей международной научно-практической конференции. 2016. С. 7-8.

Воронов М.П., Крайнова Т.С., Часовских В.П. Структура информационной системы определения биологической продуктивности экосистем. Образование и наука в современных условиях. 2016. № 4 (9). С. 132-134.

В.П. Часовских, М.П. Воронов. Проектирование моделей и баз знаний в среде самонастраивающихся нечетких моделей для информационных систем поддержки принятия решений. Монография. – Екатеринбург: Урал. гос. лесотехн. ун-т, 2014. 375 с. ISBN 978-5-94984-481-6

Часовских В.П., Воронов М.П. Исследование системных связей и закономерностей функционирования корпоративной информационной системы лесопромышленного предприятия в среде ADABAS и Natural: Монография. – Екатеринбург: Урал. гос. лесотехн. ун-т, 2012. 180 с. - ISBN 978-5-94984-213-3. - ISBN 978-5-94984-423-6.

Воронов М.П., Усольцев В.А., Часовских В.П. Исследование методов и разработка информационной системы определения и картирования депонируемого лесами углерода в среде Natural: Монография. – Екатеринбург: Урал. гос. лесотехн. ун-т, 2012. 192 с. - ISBN 978–5–94984–296–6. – ISBN 978-5-94984-426-7.

Usoltsev V.A. Forest biomass and primary production database for Eurasia. CD-version. The second edition, enlarged and re-harmonized. Yekaterinburg: Ural State Forest Engineering University, 2013 (http://elar.usfeu.ru/handle/123456789/3059). - ISBN 978-5-94984-438-0.

Усольцев В.А., Воробейчик Е.Л., Бергман И.Е. Биологическая продуктивность лесов Урала в условиях техногенного загрязнения: Исследование системы связей и закономерностей. Екатеринбург: УГЛТУ, 2012. 365 с.

(http://elar.usfeu.ru/handle/123456789/458). - ISBN 978-5-94984-405-2.

Усольцев В.А. Продукционные показатели и конкурентные отношения деревьев. Исследование зависимостей. Екатеринбург: УГЛТУ, 2013. 553 с. (http://elar.usfeu.ru/handle/123456789/2627). - ISBN 978-5-94984-444-1.

Усольцев В.А. Вертикально-фракционная структура фитомассы деревьев. Исследование закономерностей. Екатеринбург: УГЛТУ, 2013. 603 с. (http://elar.usfeu.ru/handle/123456789/2771). - ISBN 978-5-94984-439-7.

Фримен, Адам. MVC 5 с примерами на C# для профессионалов, 5-изд..: Пер. с анг. – М.: ООО «И.Д. Вильямс», 2015. – 736 стр. IBN 978-5-8459-1911-3

Фримен, Адам. jQuery для профессионалов, 5-изд..: Пер. с анг. –М.: ООО «И.Д. Вильямс», 2015. – 1040 стр. IBN 978-5-8459-1919-9

Столбовский Д.Н. Основы разработки Web-приложений на ASP.NET: Учебное пособие /Д.Н. Столбовский,— М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2011. — 304 с.: ил., табл. —(Основы информационных технологий). ISBN 978-5-94774-991-5 (БИНОМ.ЛЗ)

Эспозито Дино. Программирование на основе Microsoft ASP.NET MVC. 2-е издание / Пер. с англ. — М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2012. — 464 стр.: ил. ISBN 978-5-7502-0414-4 («Русская редакция») ISBN 978-5-9775-0885-8 («БХВ-Петербург»)

Фримен. Адам, Сандерсон. Стивен. ASP.NET MVC 3 Framework с примерами на C# для профессионалов, 3-е изд. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2012. — 672 с. : ил. — Парал. тит. англ. ISBN 978-5-8459-1758-4 (рус.)

Эспозито Д. Программирование с использованием Microsoft ASP.NET 4. — СПб.: Питер, 2013. — 880 с.: ил. ISBN 978-5-459-00346-8

Мак-Дональд. Мэтью. Фримен. Адам, Шпушта, Марио. Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов, 4-е изд. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1424 с. : ил. — Парал. тит. англ. ISBN 978-5-8459-1702-7 (рус.)

Макки, Алекс. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2010. — 416 с. : ил. — Парал. тит. англ.ISBN 978-5-8459-1639-6 (рус.)

Мак-Дональд. Мэтью. WPF 4: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1024 с. : ил. — Парал. тит. англ.ISBN 978-5-8459-1657-0 (рус.)

Эспозито Д. Разработка веб-приложений с использованием ASP.NET и AJAX. — СПб.: Пи¬тер, 2012. — 400 с.: ил. ISBN 978-5-459-00347-5

Рендольф, Ник, Гарднер, Дэвид, Минутилло, Майкл, Андерсон, Крис. Visual Studio 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1184 с.: ил. — Парал. тит. англ. ISBN 978-5-8459-1683-9 (рус.)

Мак-Дональд. Мэтью. Фримен. Адам, Шпушта, Марио. Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов, 4-е изд. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1424 с. : ил. — Парал. тит. англ. ISBN 978-5-8459-1702-7 (рус.)

Макки, Алекс. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2010. — 416 с. : ил. — Парал. тит. англ.ISBN 978-5-8459-1639-6 (рус.)

Мак-Дональд. Мэтью. WPF 4: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1024 с. : ил. — Парал. тит. англ.ISBN 978-5-8459-1657-0 (рус.)

Эспозито Д. Разработка веб-приложений с использованием ASP.NET и AJAX. — СПб.: Пи¬тер, 2012. — 400 с.: ил. ISBN 978-5-459-00347-5

Рендольф, Ник, Гарднер, Дэвид, Минутилло, Майкл, Андерсон, Крис. Visual Studio 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1184 с.: ил. — Парал. тит. англ. ISBN 978-5-8459-1683-9 (рус.)

Майо Дж. M14 Самоучитель Microsoft Visual Studio 2010. - 464 с.: ил.ISBN 978-5-9775-0609-0

Голощапов А. JI. 0 Microsoft® Visual Studio 2010. — СПб.: БХВ-Петербург, 2011. — 544 с.: ил.+ CD-ROM — (В подлиннике) ISBN 978-5-9775-0617-5

Хестер Н. Сайт с нуля в Expression Studio. - М.: ДМК Пресс, 2011. - 252 с.: ил.

ISBN 978-5-94074-700-0 Постолит А. В. Visual Studio .NET: разработка приложений баз данных. — СПб.: БХВ-Петербург, 2003. — 544 с.: ил. ISBN 5-94157-309-X

Лоран Буньон. 1 Создание web-сайтов в Silverlight; Пер. с англ. Слинкина А.А. - М.: ДМК Пресс, 2012. - 528 с.: ил. ISBN 978-5-94074-307-1

Майк Ганделрой. ADO и ADO.NET. Полное руководство.: Пер. с англ.— К.: ВЕК+, СПб.: КОРОНА принт, К.: НТИ, М.: Энтроп, 2011.— 912 с. ISBN 978-5-7931-0862-1

Зиборов В. В. Visual C#2010 на примерах. — СПб.: БХВ-Петербург, 2011. —432 с.: ил. + CD-ROM ISBN 978-5-9775-0698-4

Зиборов В. В. Visual C# 2012 на примерах.—СПб.: БХВ-Петербург, 2013.— 480 с.: ил. ISBN 978-5-9775-0888-9

Мак-Дональд, Мэтью. Silverlight 5 с примерами на C# для профессионалов, 4-е изд. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2013. — 848 с. : ил. — Парад, тит. англ. ISBN 978-5-8459-1784-3 (рус.)

Браун П. Silverlight. Практическое руководство. — СПб.: Питер, 2012. — 816 с.: ил. ISBN 978-5-459-00408-3

Макфарланд Д. JavaScript и jQuery : исчерпывающее руководство / Дэвид Мак¬фарланд ; пер. с англ. С. В. Черникова. — 2-е изд.— М. : Эксмо, 2012. — 688 с. + 1 DVD. — (Мировой компьютерный бестселлер). ISBN 978-5-699-56185-8

Котеров, Д. В. PHP 5 / Д. В. Котеров, А. Ф. Костарев. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2012. — 1104 с.: ил. — (В подлиннике) ISBN 978-5-9775-0315-Маккоу А. Веб-приложения на JavaScript. — СПб.: Питер, 2012. — 288 с.: ил. ISBN 978-5-459-01504-1

Горохов Д. Методы организации хранения данных в СУБД / Д. Горохов, В. Чернов СУБД 2003 № 3.

Джеффри Д. Ульман. Реляционные базы данных / Джеффри Д. Ульман, Дженнифер

Уидом – М.: Лори, 2014 г. 384 с.

Джон Гудсон. Практическое руководство по доступу к данным./ Джеффри Д. Ульман, Дженнифер Уидом. – СПб.: БХВ-Петербург, 2013 г. 304 с.

Калитеевский Р.Е.. Информационные технологии в лесопилении / Р.Е. Калитеевский, А.М. Артеменков, А.А. Тамби. – М.: Профи, 2010. – 192 с.

Карпова И.П. Базы данных. Учебное пособие для вузов. СПб.: Питер, 2014. 240 с.

Радчук Л.И. Основы конструирования изделий из древесины Учебное пособие. — М.: Московский государственный университет леса (МГУЛ), 2006. — 200 с.

Уильям Р. Станек. Microsoft SQL Server 2012. Справочник администратора. СПб.: Русская Редакция, БХВ-Петербург, 2013 г. 576 с.

Эспозито Д. Программирование с использованием Microsoft ASP.NET 4. СПб.: Питер, 2013. 880 с.

Bruce Johnson. Professional Visual Studio 2013. John Wiley and Sons, Ltd, 2014. 1104 с.

Часовских В.П., Стаин Д.А. Структура, содержание и среда разработки веб-сайта вуза// Эко-Потенциал. 2013. № 3-4. С. 160-172 (http://management-usfeu.ru/Gurnal)

Часовских В.П. Сайт преподавателя вуза – реальное приложение. Эко-потенциал. 2015. № 1 (9). С. 61-78. ISSN 2310 – 2888.

Часовских В.П., Кох Е.В. Сайт преподавателя вуза – база данных и первая страница. Эко-потенциал. 2015. № 1 (9). С. 79-90. ISSN 2310 – 2888.

Часовских В.П., Кох Е.В. Сайт преподавателя вуза – проект MVC в Visual Studio 2013. Эко-потенциал. 2015. № 1 (9). С. 91-94. ISSN 2310 – 2888.

Часовских В.П., Стаин Д.А. Структура, содержание и среда разработки веб-сайта вуза//Эко – Потенциал: журнал мульти дисциплинарных научных публикаций, Уральский государственный лесотехнический университет. Екатеринбург. 2013. № 3-4, с. 160-173. ISSN 2310 – 2888.

Татур Ю.Г. Высшее образование: методология и опыт проектирования: Логос, Университетская книга; Москва; 2006. Размер: 636 Кб. ISBN 5-98704-136-8

Баканова М. В. Интеграция образовательных ресурсов в процессе разработки web-сайта выпускающей кафедры вуза // Известия ПГПУ им. В. Г. Белинского. 2009. № 17. с. 70-74.

Веряева Ю. А.,Максимов А. В.,Рязанов М. А. Разработка информационной структуры веб-сайта кафедры вуза// Известия АГУ. № 1-1 / том 69 / 2011. С. 64-70.

Приказ Федеральной службы по надзору в сфере образования и науки (Рособрнадзор) от 29 мая 2014 г. N 785 г. Москва.

В.П. Часовских, Е.В. Кох, Д.А. Стаин. Исследование системных связей, закономерностей функционирования образовательной системы вуза и повышение эффективности её управления за счет создания портфолио студента современными средствами Web-технологий//Эко – Потенциал: журнал мульти дисциплинарных научных публикаций, Уральский государственный лесотехнический университет. Екатеринбург. 2015. № 2(10), с. 106-109. ISSN 2310 – 2888.

Антонниоу Г., Грос П., Хармелен ван Ф., Хоекстра Р. Семантический веб/ пер. с англ. Т.Шульга.- М.: ДМК Пресс, 2016. – 240 с. ISBN 978-5-97060-333-8/