Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(МОСКОВСКИЙ ПОЛИТЕХ)

КУРСОВАЯ РАБОТА

Проектирование и разработка баз данных

Факультет «Информационные технологии»

Кафедра «Прикладная информатика»

Группа: 233-322

Студент: Ежов Д.А.

Преподаватель: Дедёхина О.В.

Москва, 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc195565195)

[Теоретическая часть 5](#_Toc195565196)

[Практическая часть 7](#_Toc195565197)

[Структура базы данных 7](#_Toc195565198)

[Реализуемые запросы 9](#_Toc195565199)

[Описание форм интерфейса 11](#_Toc195565200)

[ЗАКЛЮЧЕНИЕ 28](#_Toc195565201)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc195565202)

[ПРИЛОЖЕНИЕ 31](#_Toc195565203)

[Таблица со списком таблиц 31](#_Toc195565204)

[Листинг кода 32](#_Toc195565205)

ВВЕДЕНИЕ

В современном мире информационные технологии играют ключевую роль в образовательном процессе, особенно в сфере обучения программированию. Появление множества фреймворков и технологий требует новых подходов к практическому освоению материала. Традиционные методы обучения постепенно уступают место современным образовательным платформам, предоставляющим возможность получения практических навыков через решение реальных задач разработки.

Целью данной работы является разработка образовательной платформы "Практикум по фреймворкам", направленной на практическое изучение современных фреймворков через решение реальных задач разработки. Платформа работает по принципу, схожему с LeetCode или CodeWars, но с фокусом на практическое применение фреймворков вместо алгоритмических задач.

Основные направления работы над проектом включают комплекс мероприятий, направленных на создание полноценной образовательной платформы с широким функционалом.

1. Первая задача заключается в создании системы регистрации и авторизации пользователей с разделением ролей. Это предполагает разработку механизмов аутентификации, реализацию системы ролей (пользователь, администратор) с различными уровнями доступа и обеспечение безопасного хранения данных.
2. Вторая задача - реализация функционала просмотра и решения заданий по различным фреймворкам. Необходимо создать каталог заданий с возможностью фильтрации по языкам программирования и фреймворкам, разработать систему оценки сложности заданий и начисления баллов за их выполнение, обеспечить возможность просмотра решений других пользователей.
3. Третья задача направлена на разработку механизма создания собственных заданий. Это включает создание интерфейса для добавления новых заданий, реализацию системы прикрепления начального кода задания и разработку функционала для описания условий задания и критериев его выполнения.
4. Четвертая задача предполагает внедрение системы подсказок с элементами геймификации. Требуется разработать систему подсказок разной стоимости, создать механизм оплаты подсказок заработанными баллами и реализовать учет использованных подсказок для каждого пользователя.
5. Пятая задача связана с созданием системы оценки решений и заданий других пользователей. Необходимо разработать систему лайков для оценки качественных решений и заданий.
6. Шестая задача включает разработку административного интерфейса для управления контентом. Это подразумевает создание панели управления пользователями, разработку инструментов модерации контента и реализацию управления языками программирования и фреймворками.

Актуальность разработки подобной платформы обусловлена растущим спросом на специалистов, владеющих современными фреймворками. Образовательные учреждения часто не успевают за быстрым развитием технологий, что создает потребность в дополнительных инструментах практического обучения. Предлагаемая платформа позволяет эффективно осваивать современные фреймворки через практику, что повышает качество обучения программированию.

Теоретическая часть

Разработка современных образовательных платформ требует тщательного подхода к выбору технологий и архитектурных решений. При создании платформы "Практикум по фреймворкам" были учтены не только современные тенденции веб-разработки, но и специфические требования к образовательным системам.

В основе проекта лежит современный веб-фреймворк ASP.NET Core MVC, обеспечивающий высокую производительность и безопасность приложения. Выбор именно этого решения обусловлен его способностью эффективно обрабатывать большое количество одновременных подключений, что особенно важно для образовательной платформы с потенциально большим числом пользователей. Система разделена на несколько слоев: представление (интерфейс пользователя), бизнес-логика (обработка данных) и доступ к данным (взаимодействие с базой данных). Такая архитектура позволяет эффективно организовать кодовую базу проекта, упрощая поддержку и дальнейшее развитие системы.

Для хранения данных используется PostgreSQL - мощная объектно-реляционная система управления базами данных с открытым исходным кодом. Вместо автоматических средств работы с базой данных применяется микро-ORM Dapper, позволяющая писать оптимизированные SQL-запросы при минимальных накладных расходах. Это решение было выбрано для достижения максимальной производительности при работе с большими объемами данных.

Управление изменениями в структуре базы данных реализовано через систему миграций FluentMigrator, что позволяет контролировать эволюцию структуры базы данных по мере развития проекта. Каждое изменение документируется и может быть применено на всех окружениях - от разработчика до продакшена.

Особое внимание уделено безопасности системы. Для аутентификации пользователей используется современный механизм JWT (JSON Web Tokens), обеспечивающий безопасную авторизацию и управление сессиями. Хеширование паролей выполняется с использованием надежного алгоритма BCrypt, гарантирующего защиту учетных данных пользователей даже в случае компрометации базы данных.

Пользовательский интерфейс платформы создан с использованием Bootstrap и JavaScript, что обеспечивает адаптивность интерфейса и кросс-браузерную совместимость. Применение этих технологий позволяет создать удобный и современный интерфейс без избыточной сложности.

Проект следует принципам чистой архитектуры с четким разделением на слои и использованием паттернов Repository и Unit of Work. Эти архитектурные решения помогают централизовать доступ к данным, обеспечить атомарность операций и упростить тестирование системы. Каждая сущность имеет свой репозиторий, реализующий базовый набор операций и дополнительные методы, специфичные для конкретной задачи.

Хранение файлов (начальный код заданий и решения пользователей) реализовано через облачное хранилище Yandex Cloud или совместимый сервис с использованием ключей S3. Такой подход обеспечивает надежное хранение файлов, их быстрый доступ и масштабируемость системы.

Выбранный технологический стек (ASP.NET Core MVC, PostgreSQL, Dapper, FluentMigrator, JWT, BCrypt, Bootstrap, JavaScript) и архитектурные решения обеспечивают создание современной, производительной и масштабируемой образовательной платформы, следуя лучшим практикам разработки веб-приложений и учитывая специфику образовательных систем.

Практическая часть

Структура базы данных

База данных является ключевым компонентом информационной системы, обеспечивающим хранение, управление и эффективный доступ к данным. При проектировании базы данных для платформы "Практикум по фреймворкам" были учтены особенности образовательных систем, требования к производительности, безопасности и надежности данных. В контексте данной системы база данных используется для хранения информации о пользователях, заданиях, решениях, языках программирования, фреймворках и других связанных данных.

Основные компоненты структуры базы данных включают в себя:

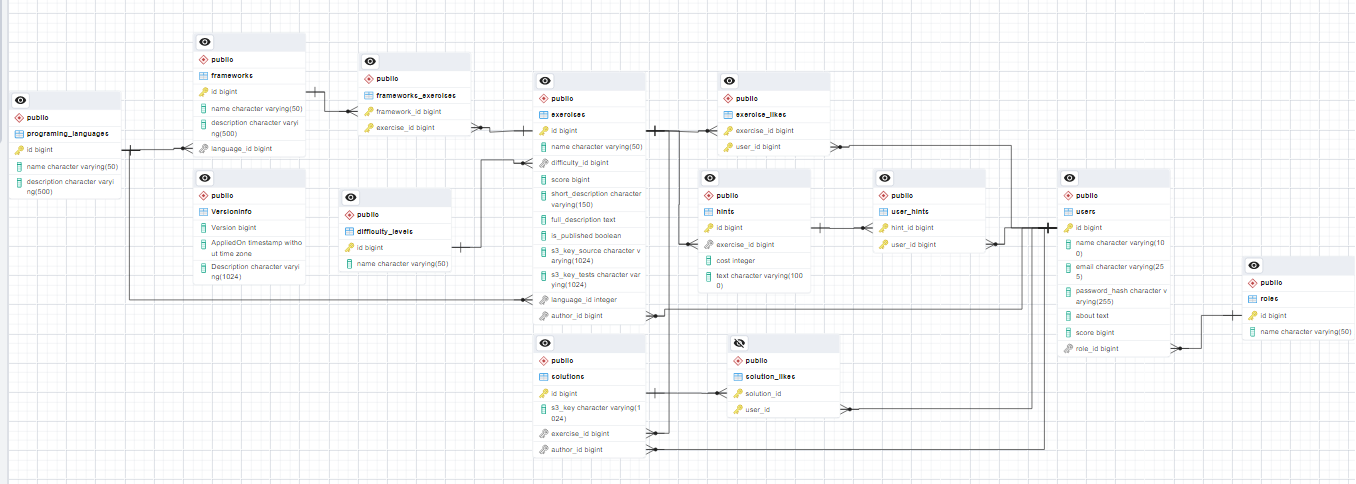
**Таблицы**: База данных состоит из нескольких ключевых таблиц, каждая из которых отвечает за хранение определенного типа данных. Например, таблица users содержит информацию о пользователях платформы, таблиц exercises — данные о заданиях, таблиц solutions — решения пользователей, а таблица hints — подсказки для заданий. Эти таблицы организованы таким образом, чтобы обеспечивать эффективное хранение и быстрый доступ к данным.

**Поля**: Каждая таблица включает набор полей, описывающих атрибуты объектов. Например, таблица users содержит поля для email, хешированного пароля, роли пользователя и других профильных данных. Таблица exercises включает поля для названия задания, описания, уровня сложности, привязки к языку программирования и фреймворку, а также начального кода задания. Эти поля позволяют точно описать характеристики каждого объекта в системе.

**Отношения**: Между таблицами установлены отношения, которые обеспечивают целостность данных и упрощают выполнение сложных запросов. Например, таблица solutions связана с таблицами users и exercises, что позволяет отслеживать решения конкретного пользователя для определенного задания. Аналогично, таблица hints связана с таблицей exercises, что позволяет организовать систему подсказок для каждого задания. Такие связи позволяют создавать запросы для анализа прогресса пользователей, популярности заданий и других метрик.

**Индексы**: Для ускорения обработки запросов используются индексы, которые создаются для часто используемых полей. Например, индексы созданы для полей author\_id в таблицах solutions и hints, а также для поля exercise\_id в таблице solutions. Это помогает значительно ускорить выполнение операций, таких как выборка всех решений пользователя или подсказок для конкретного задания.

**Ограничения**: В базе данных применяются ограничения для обеспечения целостности данных. Например, для поля author\_id в таблице solutions используется внешний ключ, который ссылается на существующие записи в таблице users. Это гарантирует, что каждое решение будет связано с реальным пользователем. Также применяются ограничения уникальности для таких полей, как email в таблице users, чтобы избежать дублирования данных.



Эта структура базы данных эффективно поддерживает работу платформы, позволяя безопасно и быстро обрабатывать данные, связанные с пользователями, заданиями, решениями и другими аспектами системы.

Реализуемые запросы

Для полноценной работы образовательной платформы "Практикум по фреймворкам" необходимо реализовать ряд запросов, связанных с регистрацией и авторизацией пользователей, управлением заданиями, решениями, языками программирования и фреймворками. Рассмотрим каждый запрос отдельно:

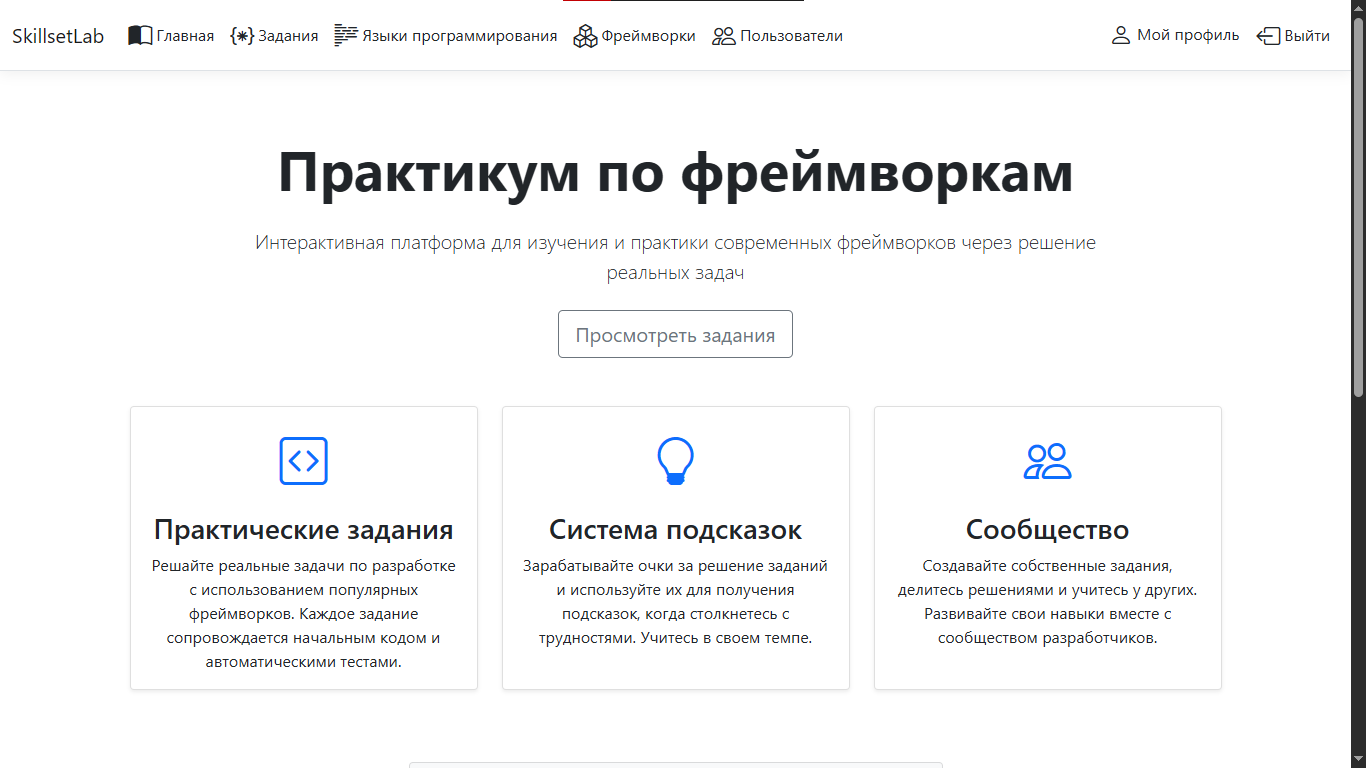
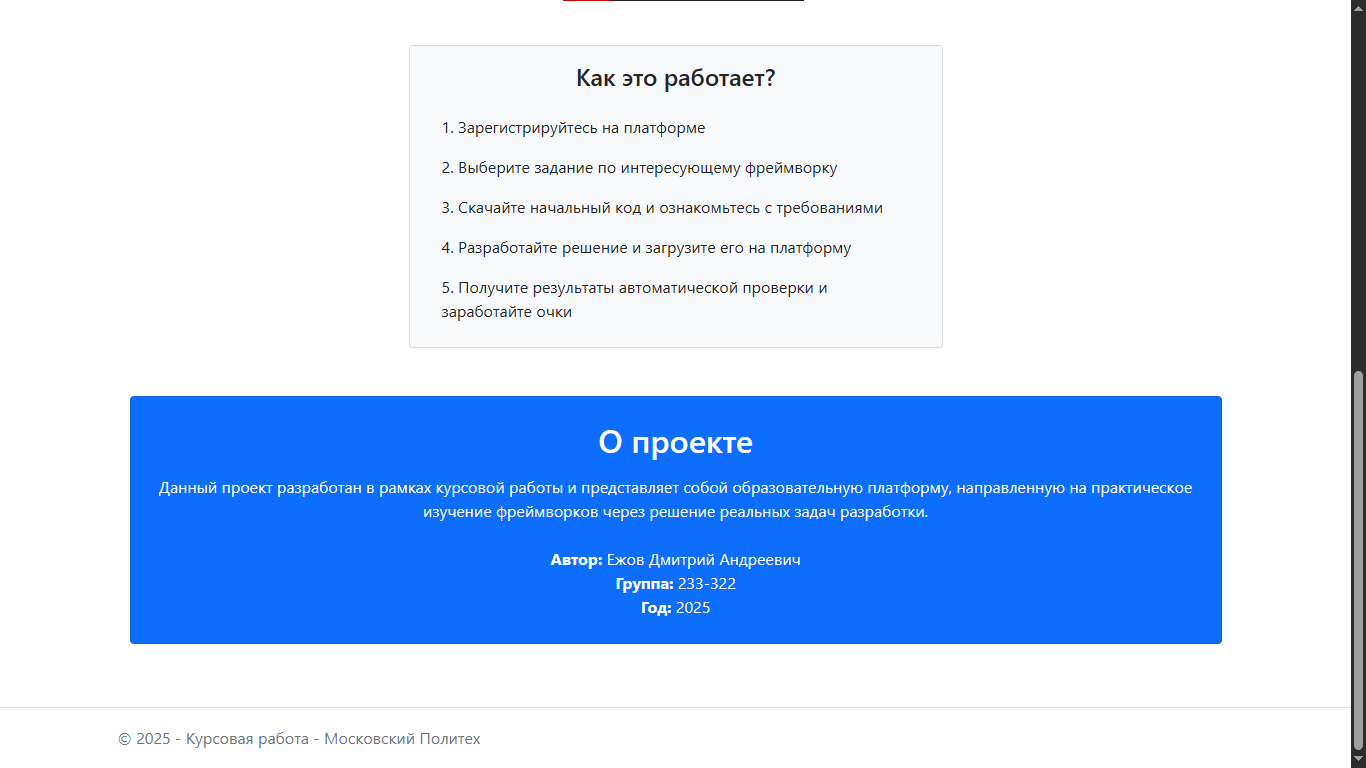
1. **Вход в систему**: Запрос позволяет пользователю войти в систему, проверяя его учетные данные (email и пароль). После успешной аутентификации создается JWT-токен, который сохраняется в cookies для обеспечения безопасной сессии.
2. **Регистрация нового пользователя**: Запрос используется для добавления нового пользователя в систему. При регистрации сохраняются данные о пользователе, включая email, зашифрованный пароль и роль (пользователь или администратор).
3. **Создание задания**: Запрос позволяет добавить новое задание в систему. Это включает в себя вставку записи в таблицу exercises с указанием названия задания, описания, уровня сложности, привязки к языку программирования и фреймворку, а также начального кода задания.
4. **Обновление задания**: Запрос позволяет пользователю изменить свое задание, если он хочет улучшить его или исправить ошибки.
5. **Удаление задания**: Запрос удаляет запись о задании из базы данных. Используется в случае, если пользователь хочет удалить свое задание.
6. **Создание решения**: Запрос используется для отправки решения пользователя на проверку. Включает в себя вставку записи в таблицу solutions с указанием идентификатора пользователя, задания и самого решения.
7. **Обновление решения**: Запрос позволяет пользователю изменить свое решение, если он хочет улучшить его или исправить ошибки.
8. **Удаление решения**: Запрос удаляет запись о решении из базы данных. Используется в случае, если пользователь хочет удалить свое решение.
9. **Оценка решения**: Запрос позволяет пользователю поставить лайк заданию другого участника. Включает вставку записи в таблицу exercise\_likes.
10. **Открытие подсказки**: Запрос позволяет пользователю получить доступ к подсказке за определенное количество баллов. Списываются баллы с счета пользователя, и подсказка становится доступной для просмотра.
11. **Оценка решения**: Запрос позволяет пользователю поставить лайк решению другого участника. Включает вставку записи в таблицу solution\_likes.
12. **Добавление нового языка программирования**: Запрос используется для добавления нового языка программирования в систему. Включает вставку записи в таблицу programming\_languages. Этот запрос доступен только администраторам.
13. **Обновление информации о языке программирования**: Запрос позволяет редактировать данные о существующем языке программирования. Этот запрос доступен только администраторам.
14. **Удаление языка программирования**: Запрос удаляет запись о языке программирования из базы данных. Этот запрос доступен только администраторам.
15. **Добавление нового фреймворка**: Запрос используется для добавления нового фреймворка в систему. Включает вставку записи в таблицу frameworks с указанием привязки к соответствующему языку программирования. Этот запрос доступен только администраторам.
16. **Обновление информации о фреймворке**: Запрос позволяет редактировать данные о существующем фреймворке.
17. **Удаление фреймворка**: Запрос удаляет запись о фреймворке из базы данных. Этот запрос доступен только администраторам.
18. **Обновление профиля пользователя**: Запрос позволяет пользователю изменить информацию о себе. Администраторы могут также изменять роли пользователей и количество баллов.
19. **Удаление пользователя**: Запрос удаляет запись о пользователе из базы данных. Этот запрос доступен только администраторам.

Каждый из этих запросов играет важную роль в функционировании образовательной платформы, обеспечивая удобный и эффективный доступ к данным о пользователях, заданиях, решениях, языках программирования и фреймворках.

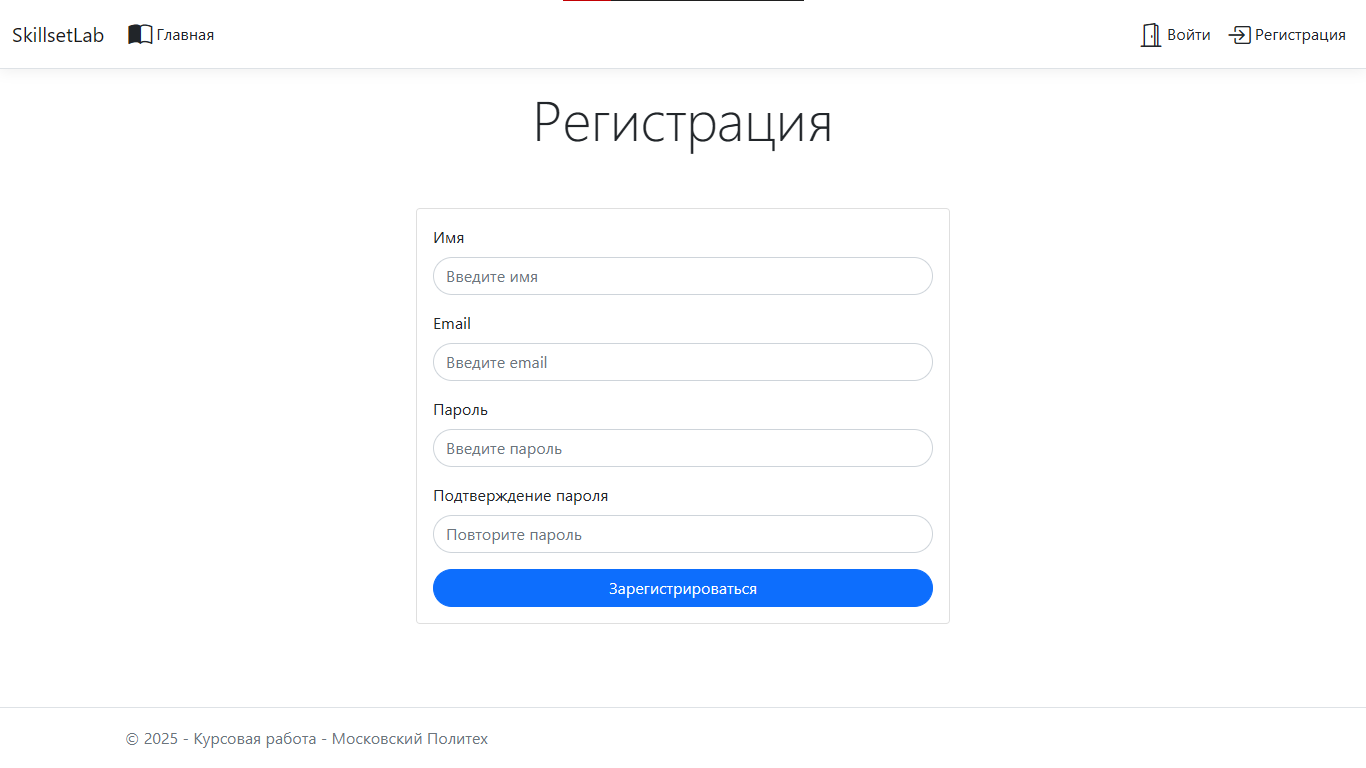
Описание форм интерфейса

Веб-интерфейс платформы "Практикум по фреймворкам" играет ключевую роль в обеспечении удобства использования и функциональности системы. Ниже приведено описание основных страниц интерфейса, которые реализованы в системе:

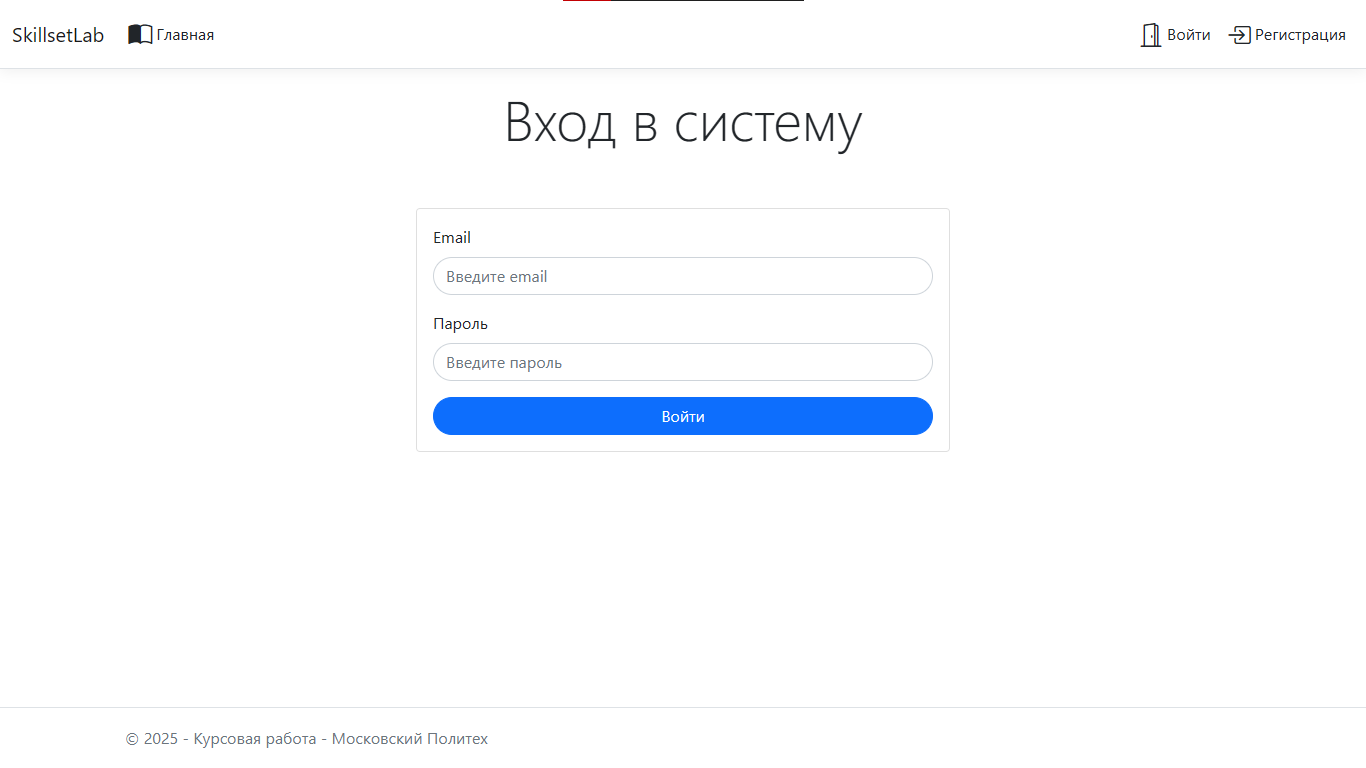
1. **Главная страница (Home):** Предоставляет пользователям общее представление о платформе и её возможностях. Страница включает заголовок "Практикум по фреймворкам", описание концепции, три информационных блока ("Практические задания", "Система подсказок", "Сообщество"), инструкцию "Как это работает?" и информацию о проекте. Для авторизованных пользователей доступна кнопка "Просмотреть задания", а для неавторизованных — кнопка "Начать обучение".

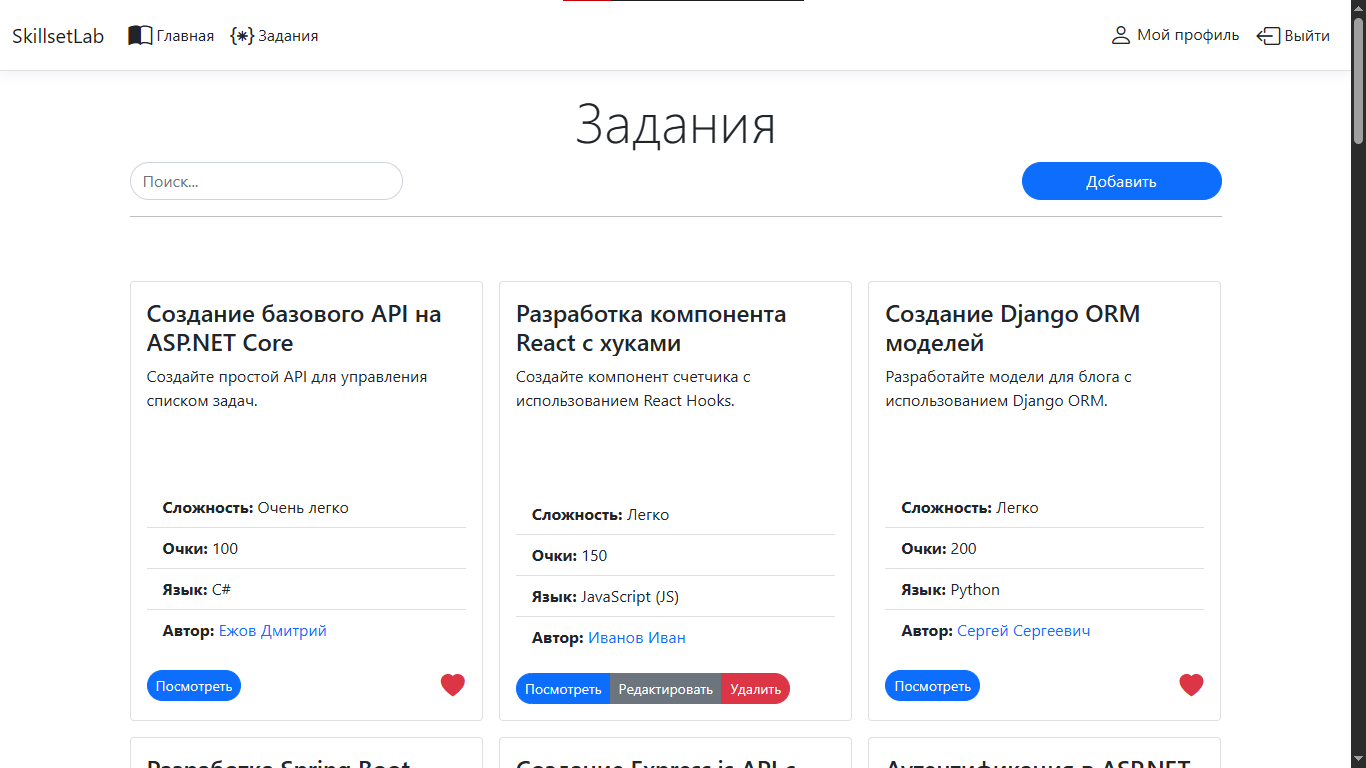
1. **Страница регистрации (Auth/Register):** Позволяет новым пользователям создать учетную запись на платформе. Форма включает поля для ввода имени, email, пароля и подтверждения пароля. После заполнения формы данные сохраняются в базе данных, и пользователь автоматически авторизуется в системе.

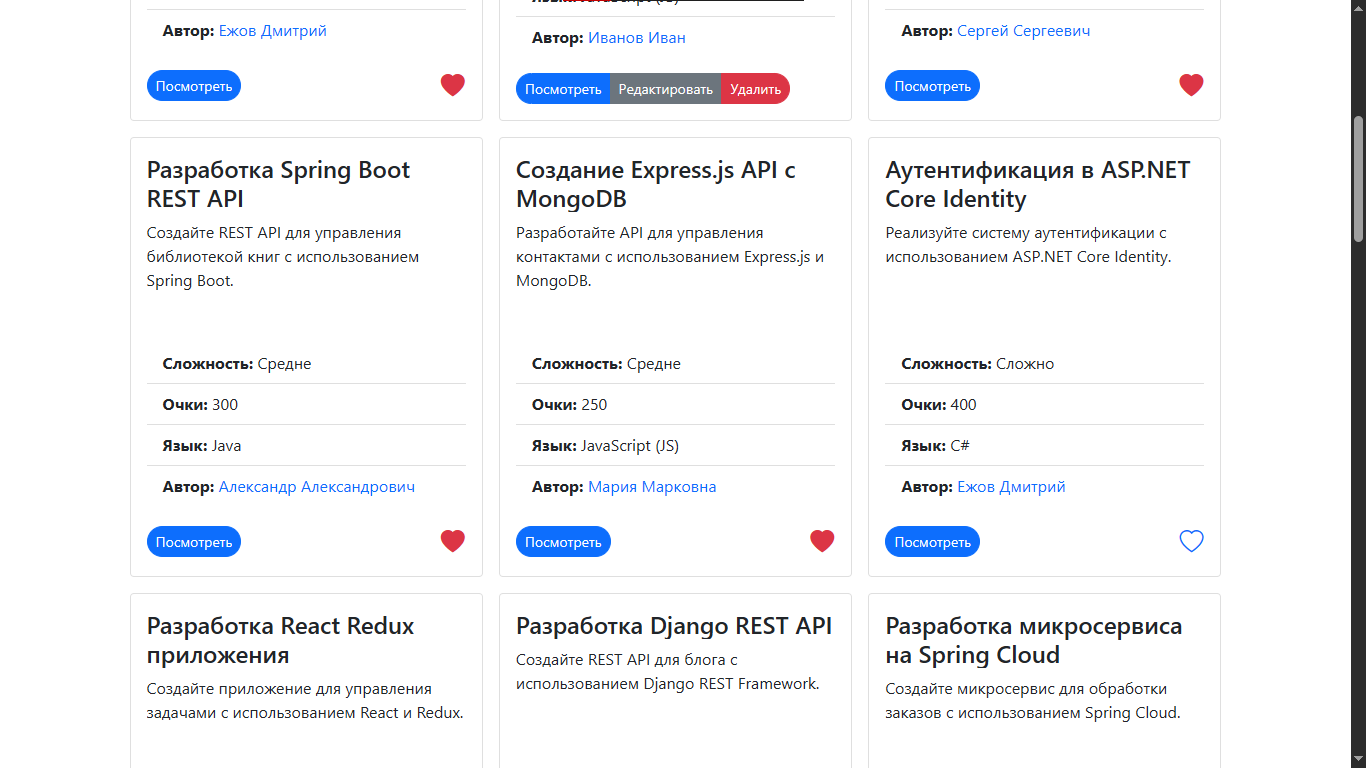


1. **Страница входа (Auth/Login):** Предоставляет пользователям возможность войти в свою учетную запись. Форма включает поля для ввода email и пароля, которые проверяются в базе данных для обеспечения правильности введенных данных и авторизации пользователя.

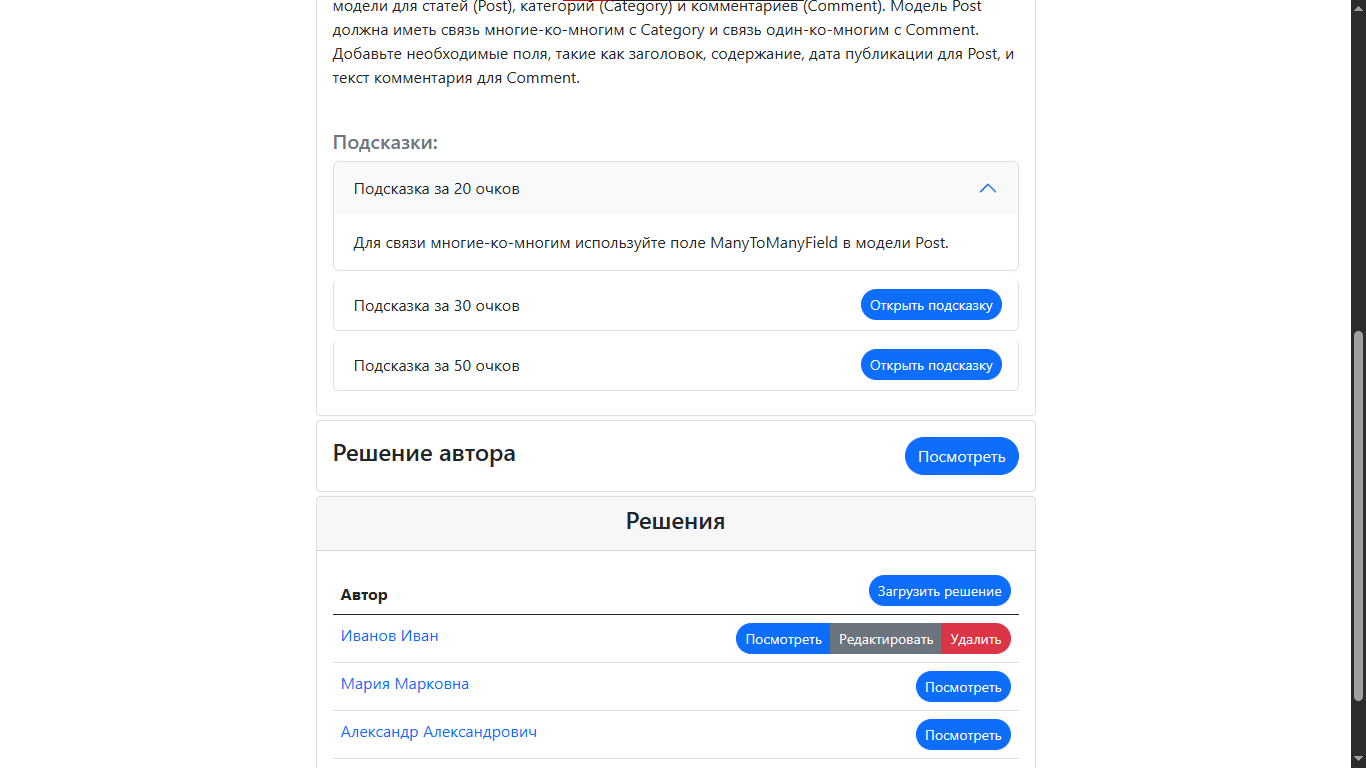


1. **Список заданий (Exercises):** Отображает карточки всех доступных заданий с информацией о названии, сложности, языке программирования и используемых фреймворках. На странице также есть поле поиска для фильтрации заданий. Есть кнопка "Добавить" (для создания нового задания), лайка/отмены лайка задания, а также перехода к просмотру задания.

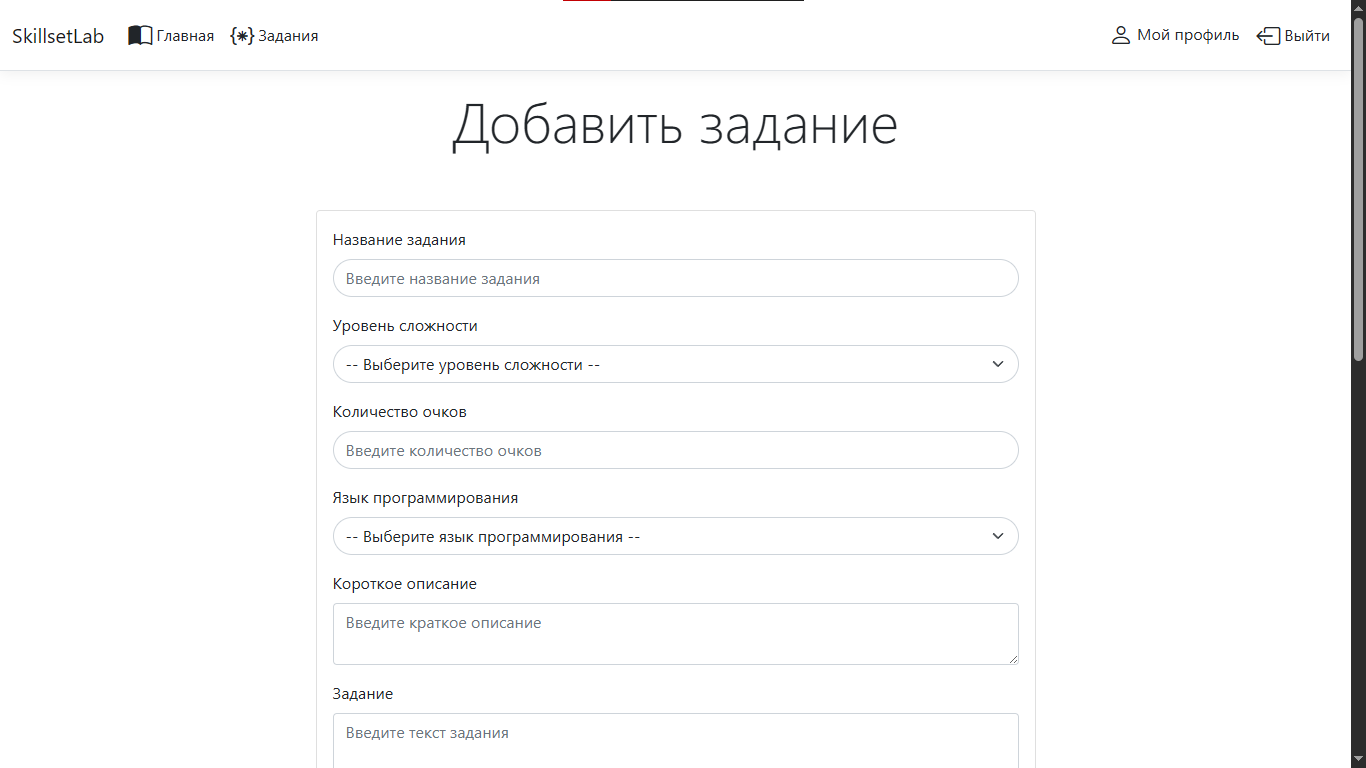


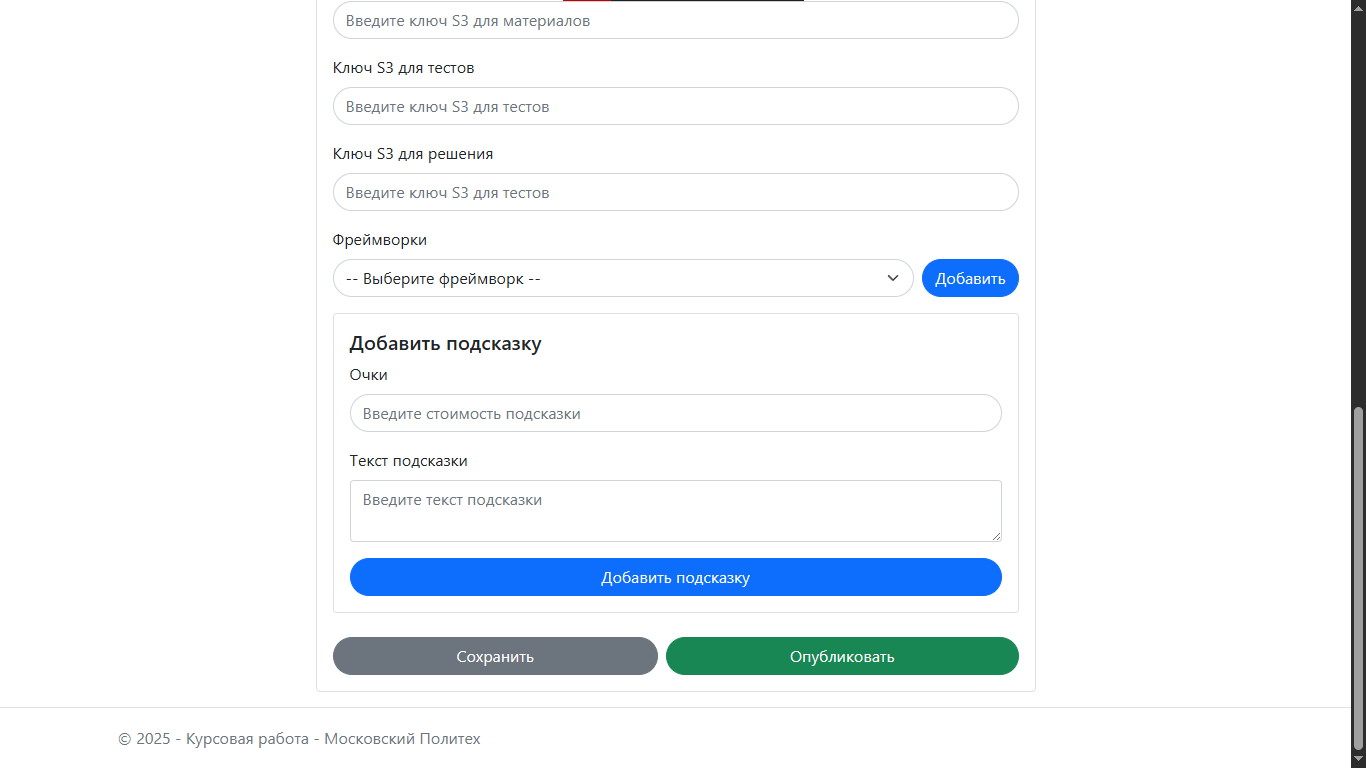


1. **Просмотр задания (Exercises/View):** Предоставляет детальную информацию о выбранном задании, включая название, сложность, количество очков, язык программирования, используемые фреймворки, автора, счетчик лайков, краткое и полное описание задания, список подсказок и список решений. Для пользователей доступны кнопки для лайка/отмены лайка задания, открытия подсказок и загрузки решения.

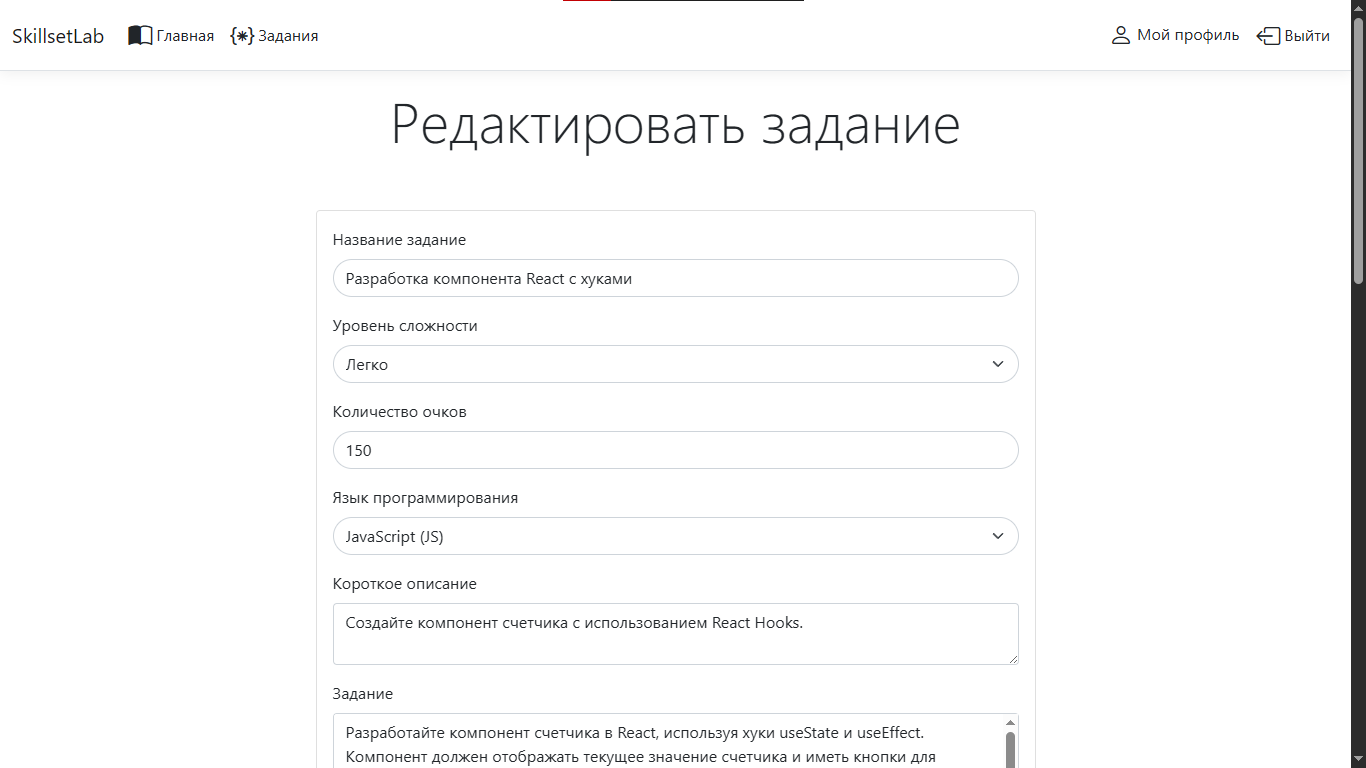
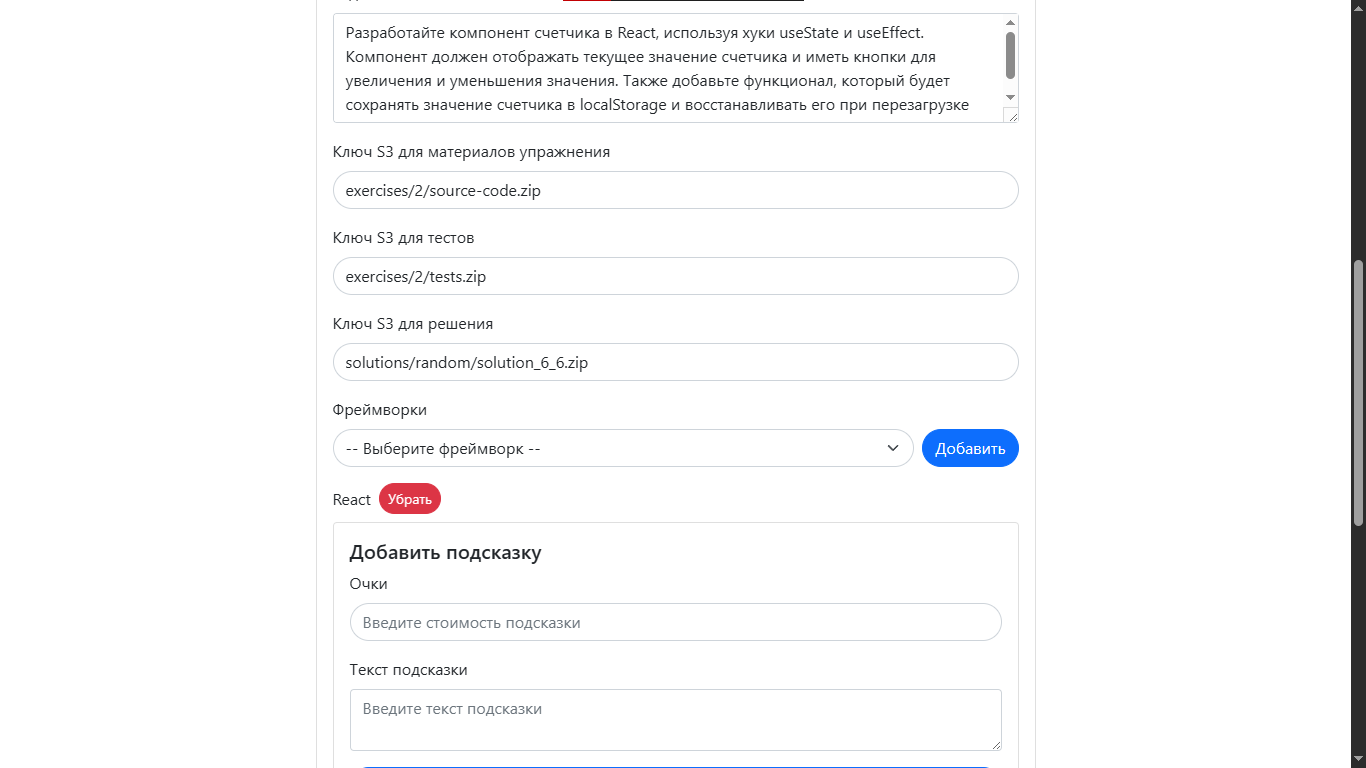
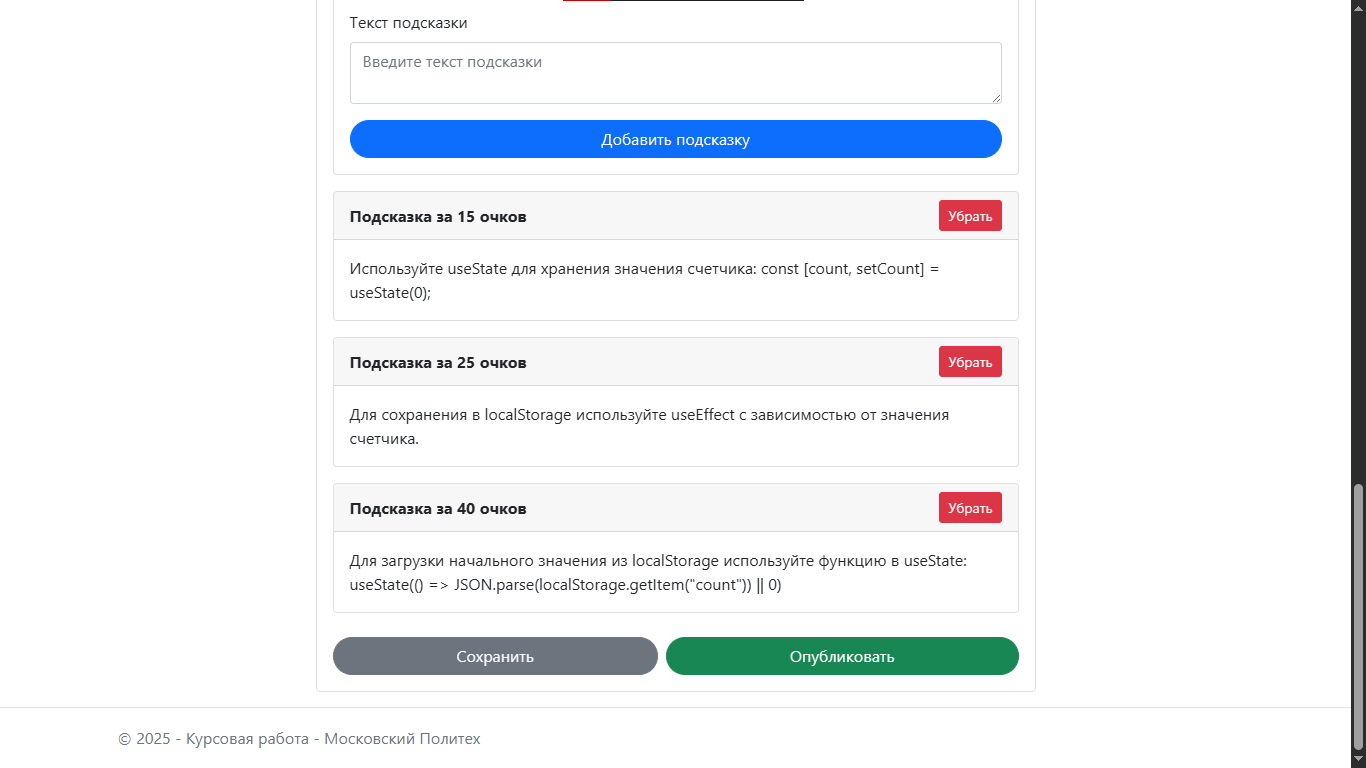
  


1. **Создание задания (Exercises/Create):** Позволяет авторизованным пользователям создавать новые задания. Форма включает поля для ввода названия задания, уровня сложности, количества очков, языка программирования, краткого и полного описания, S3-ключей для исходного кода, тестов и авторского решения, а также выбор используемых фреймворков. Задание можно сохранить как черновик или опубликовать.

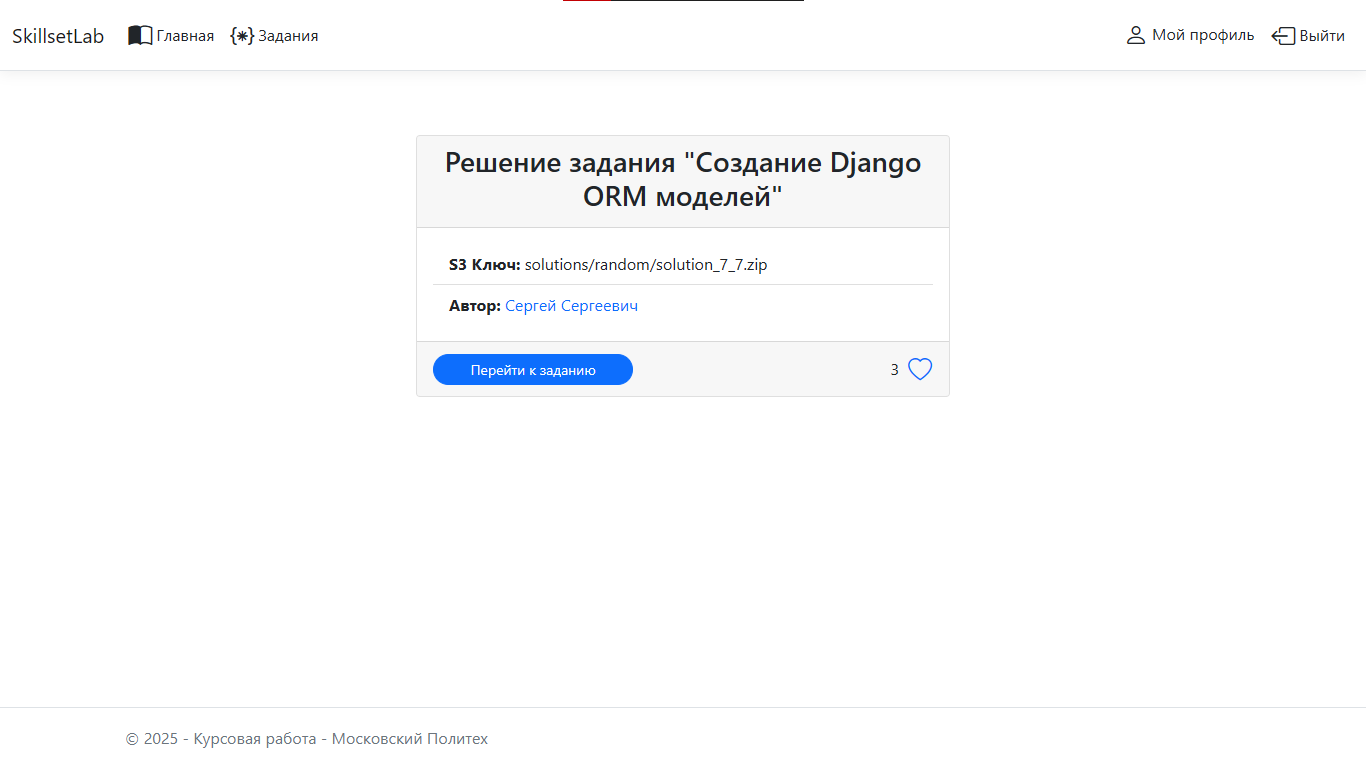




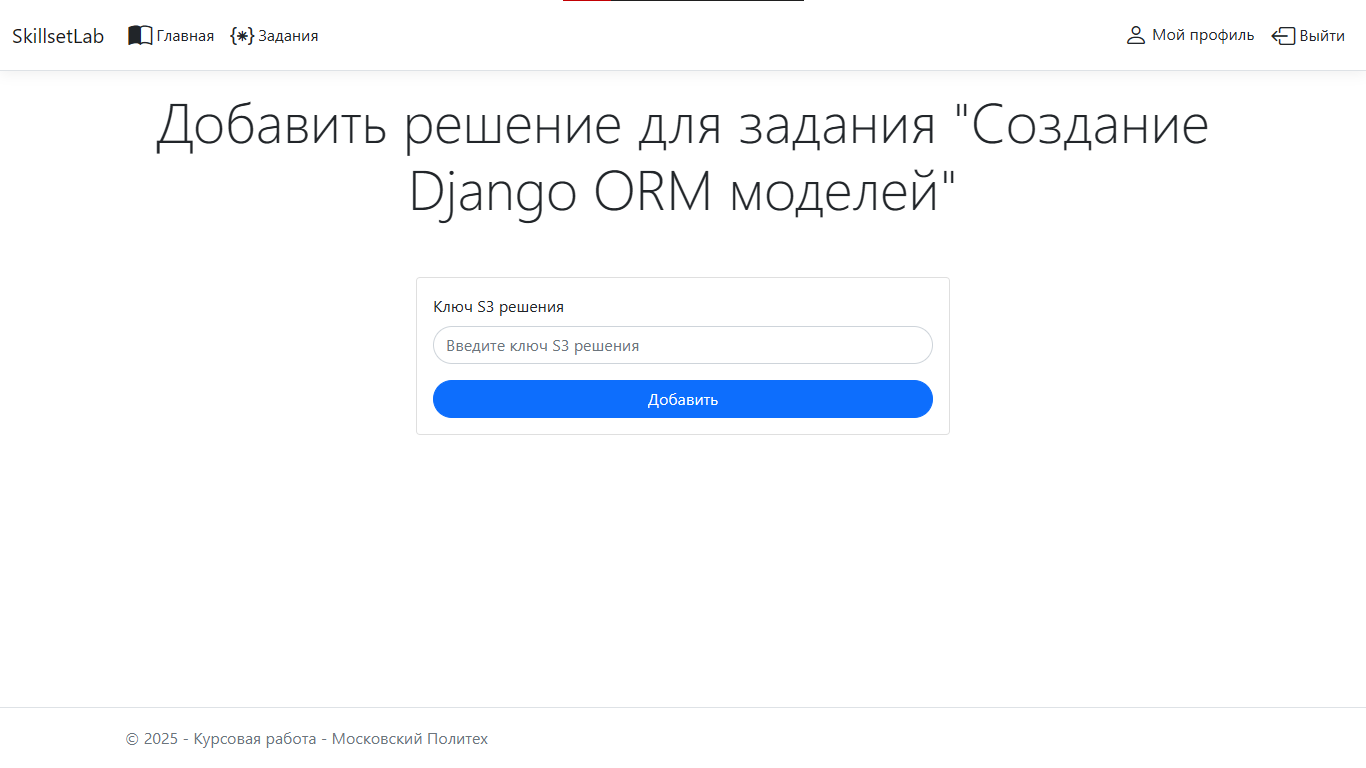
1. **Редактирование задания (Exercises/Update):** Предоставляет возможность автору или администратору редактировать существующее задание. Форма аналогична форме создания задания и позволяет обновить все параметры задания, включая текстовое описание, файлы и метаданные.

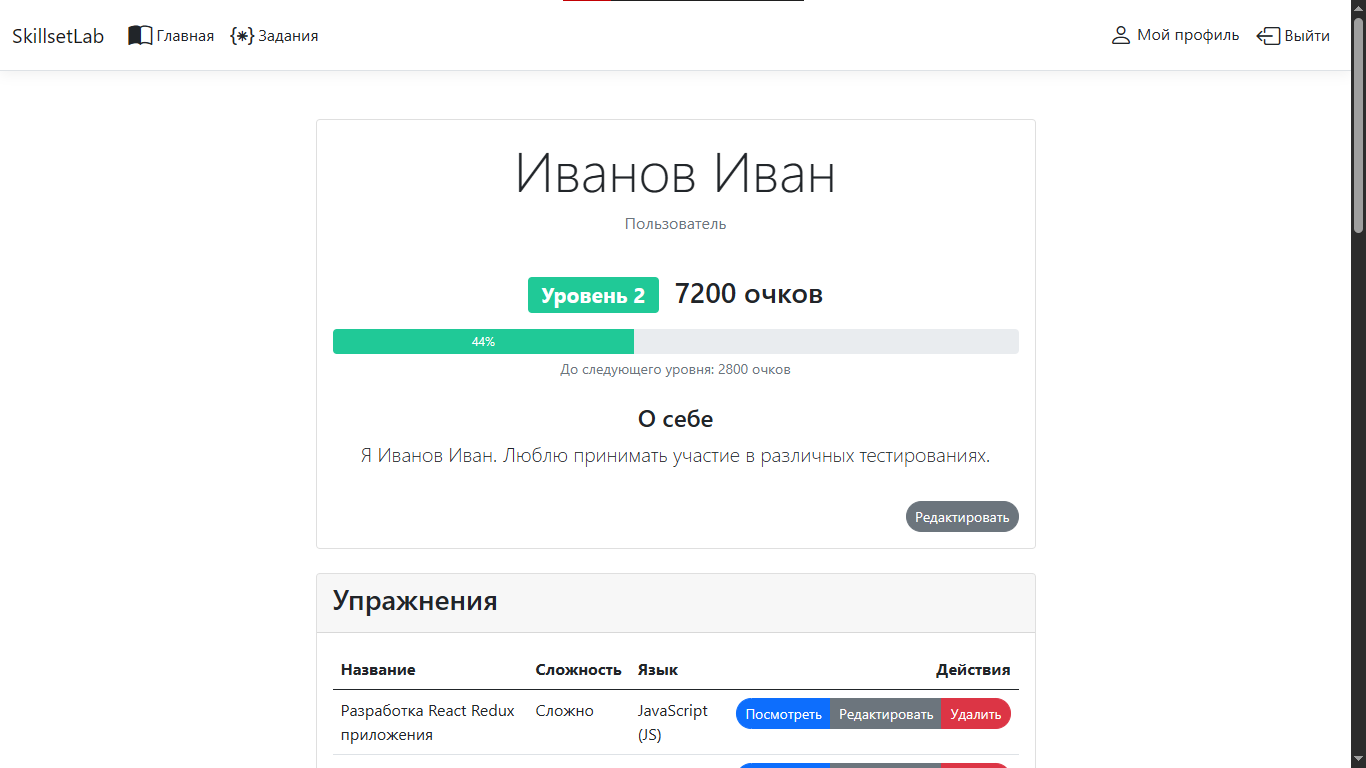
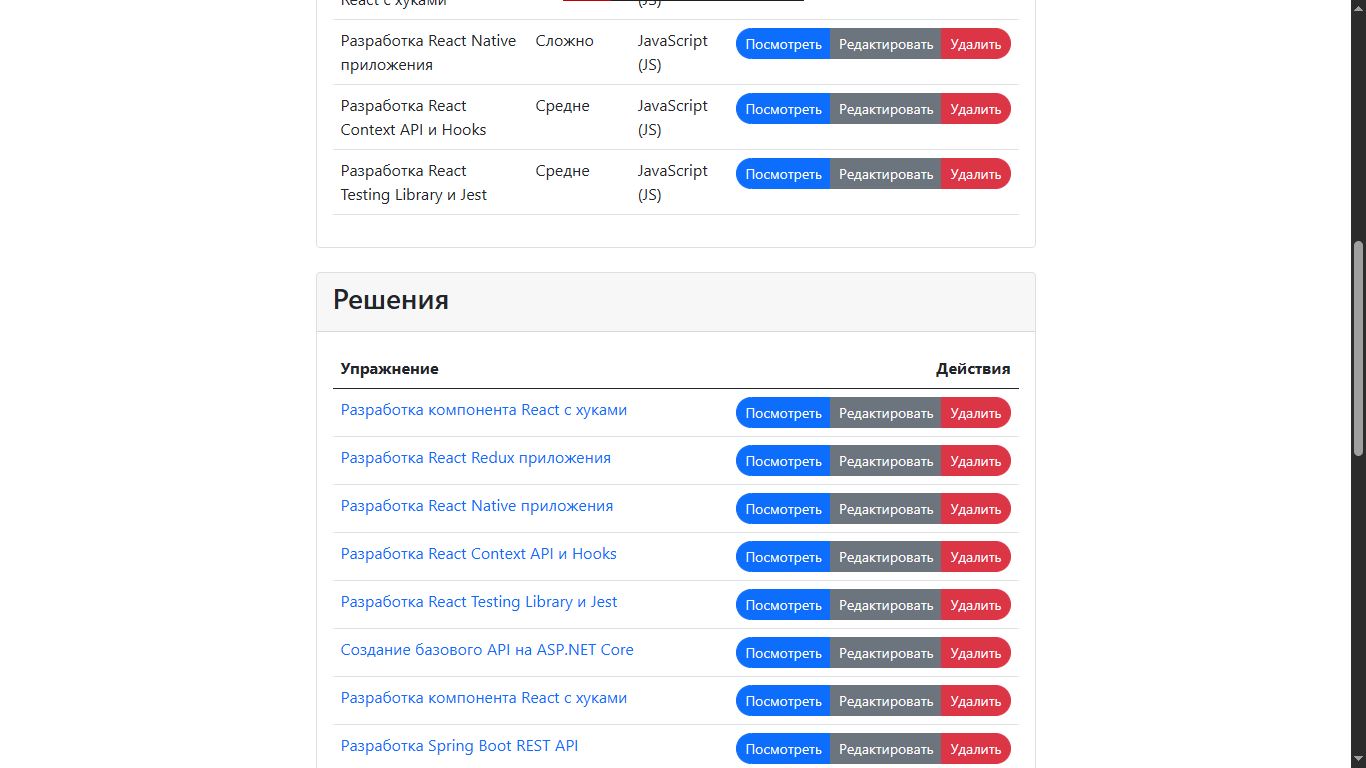
1. **Просмотр решения (Solution/View):** Отображает информацию о конкретном решении задания, включая название задания, S3-ключ решения, автора решения и счетчик лайков. Для пользователей доступны кнопки для лайка/отмены лайка решения и перехода к заданию.

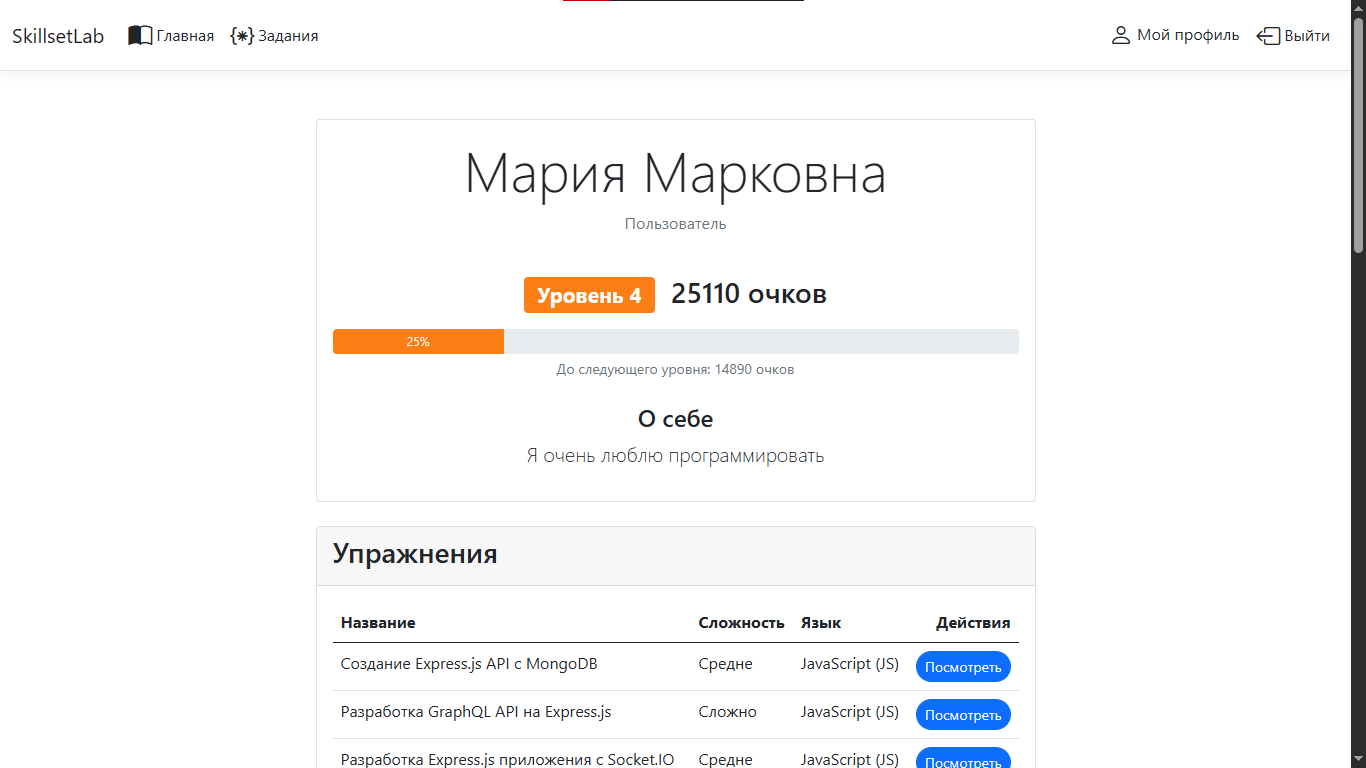


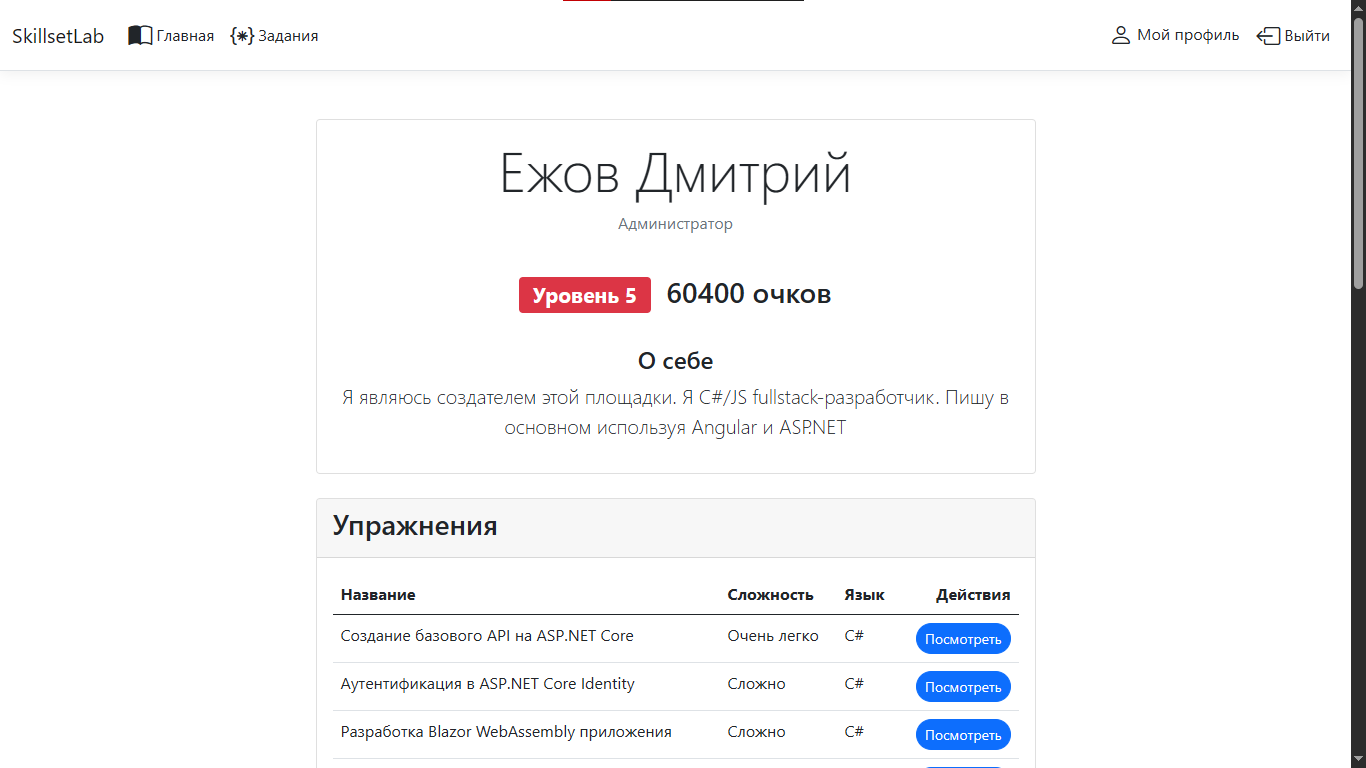
1. **Создание решения (Solution/Create):** Позволяет пользователям отправить решение для выбранного задания. Форма включает поле для загрузки файла решения через S3-ключ. После отправки решение сохраняется в базе данных и становится доступным для просмотра и оценки.



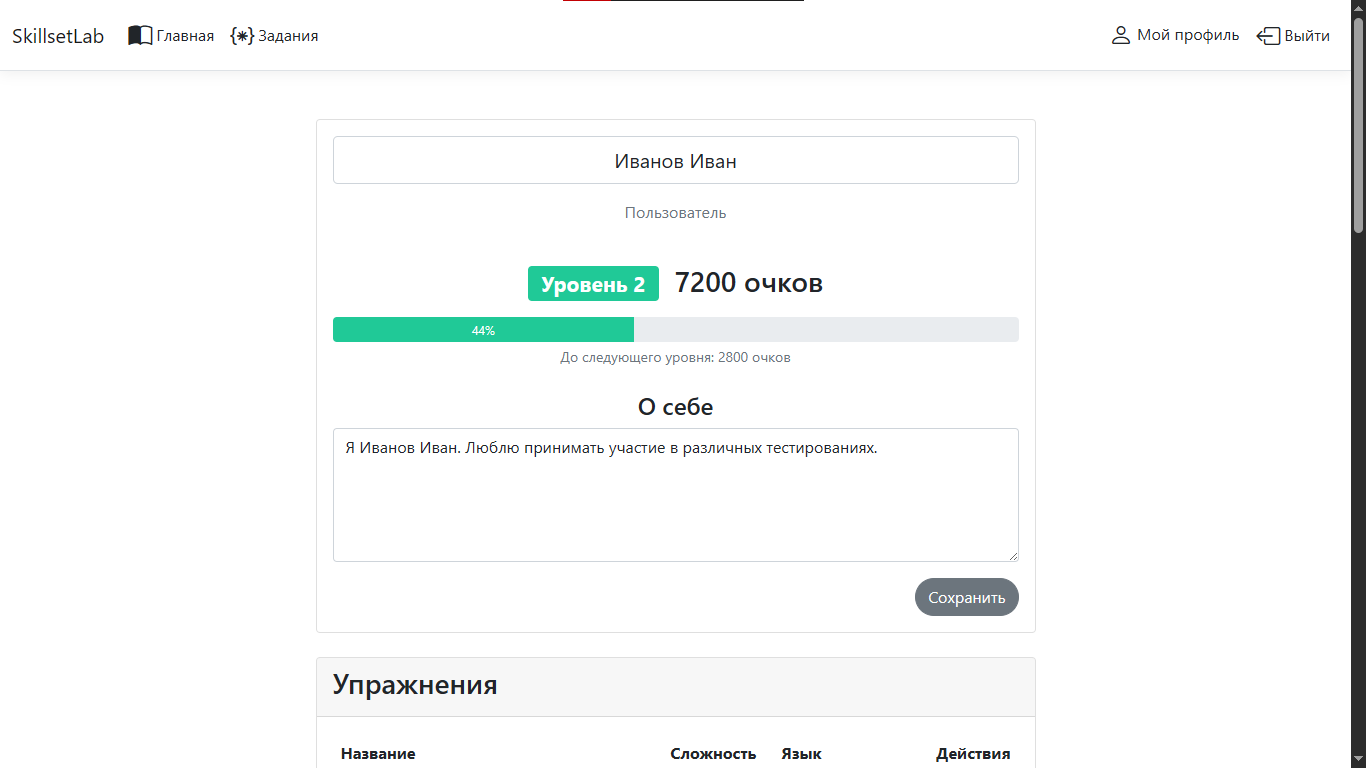
1. **Профиль пользователя (User/View):** Отображает информацию о пользователе, включая имя, email, роль, количество очков, уровень с прогресс-баром, список созданных заданий и список решений. Для владельца профиля или администратора доступны кнопки для редактирования профиля, просмотра, удаления и редактирования заданий и решений.

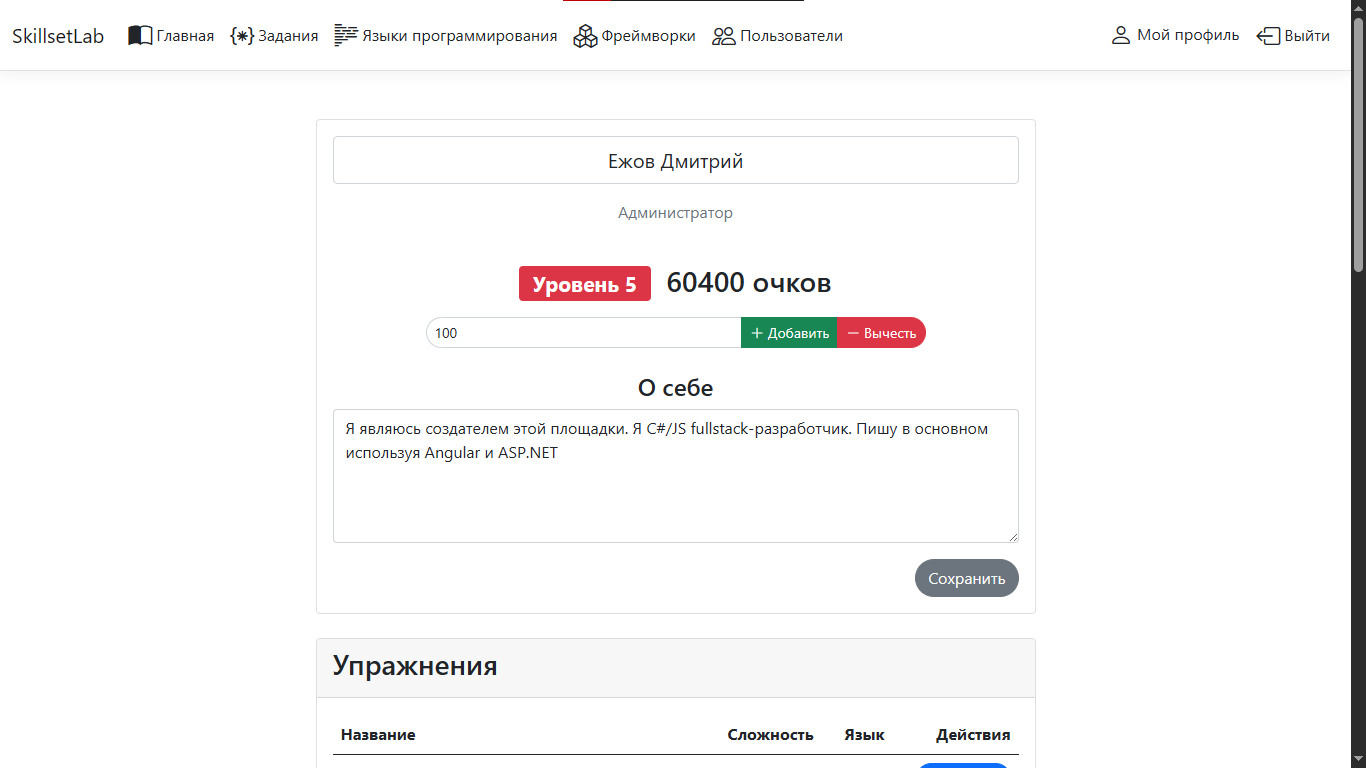
  




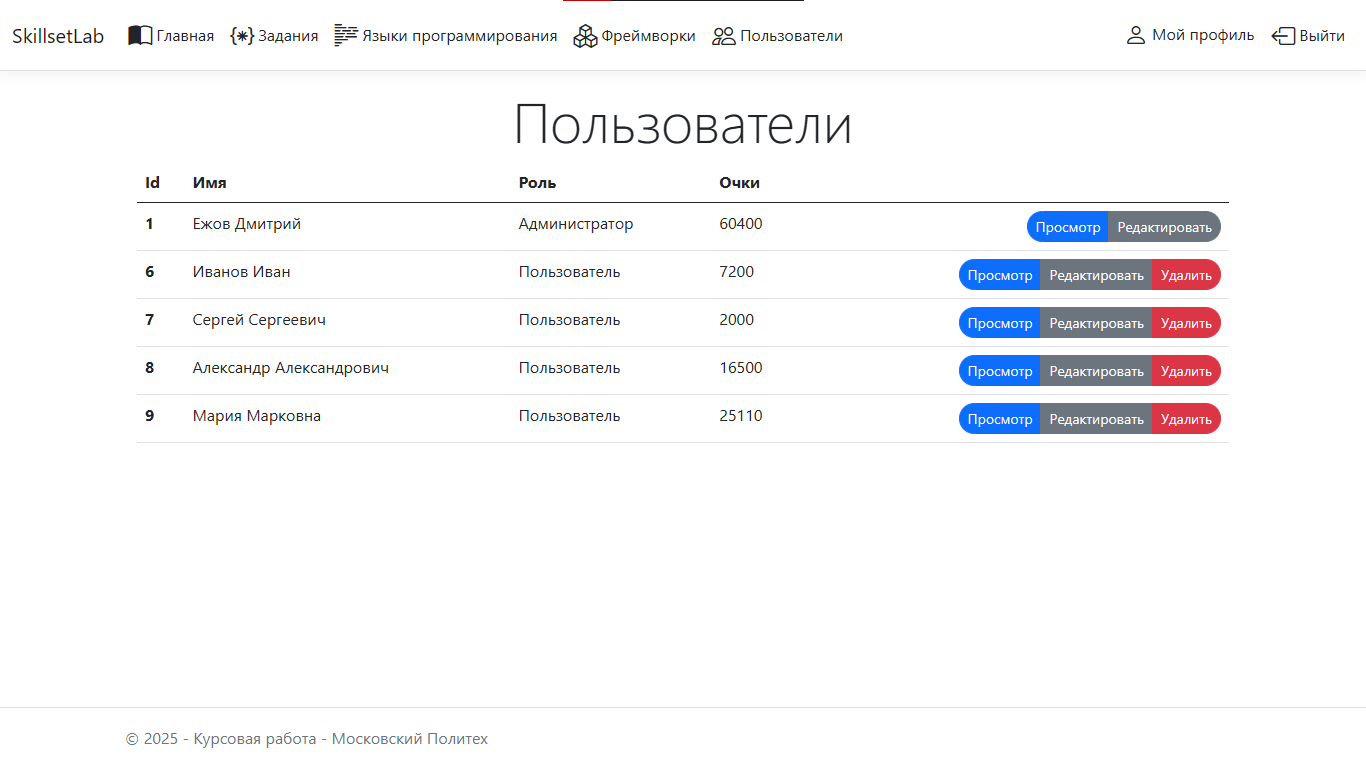


1. **Редактирование пользователя (User/Update):** Позволяет пользователю или администратору обновить информацию о профиле. Форма включает поля для изменения имени, email, информации о себе, а также роли и количества очков (доступно только администраторам).

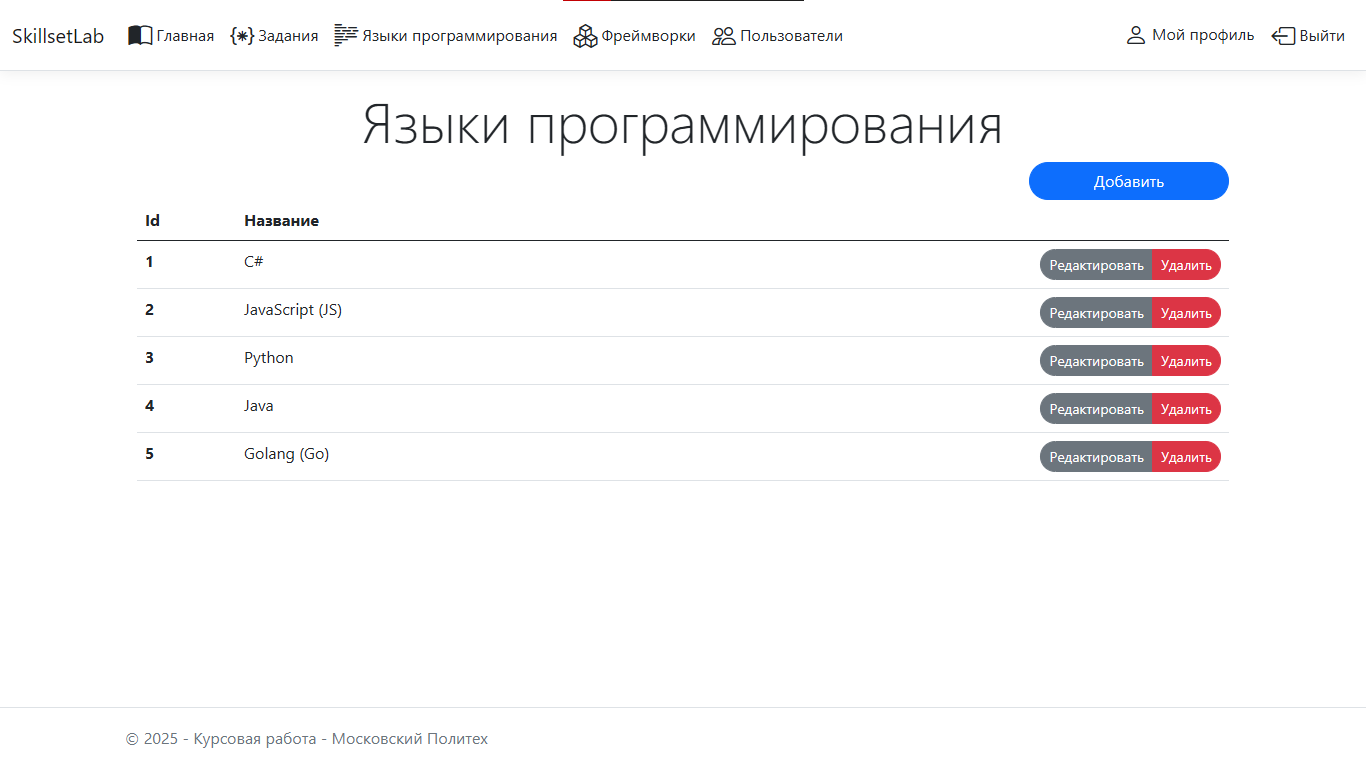




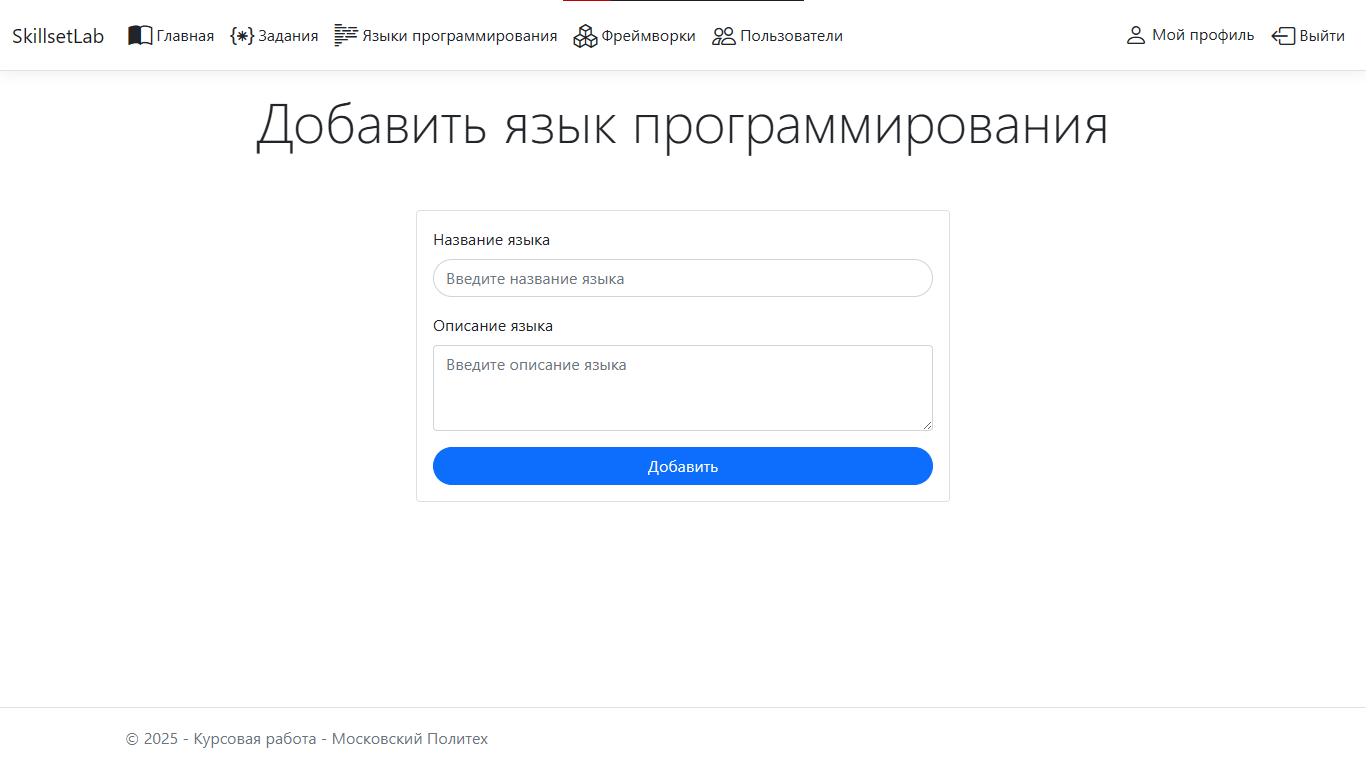
1. **Список пользователей (User):** Доступен только администраторам и отображает таблицу всех пользователей с их ID, именем, ролью и количеством очков. Для каждого пользователя доступны кнопки просмотра, удаления и редактирования профиля.



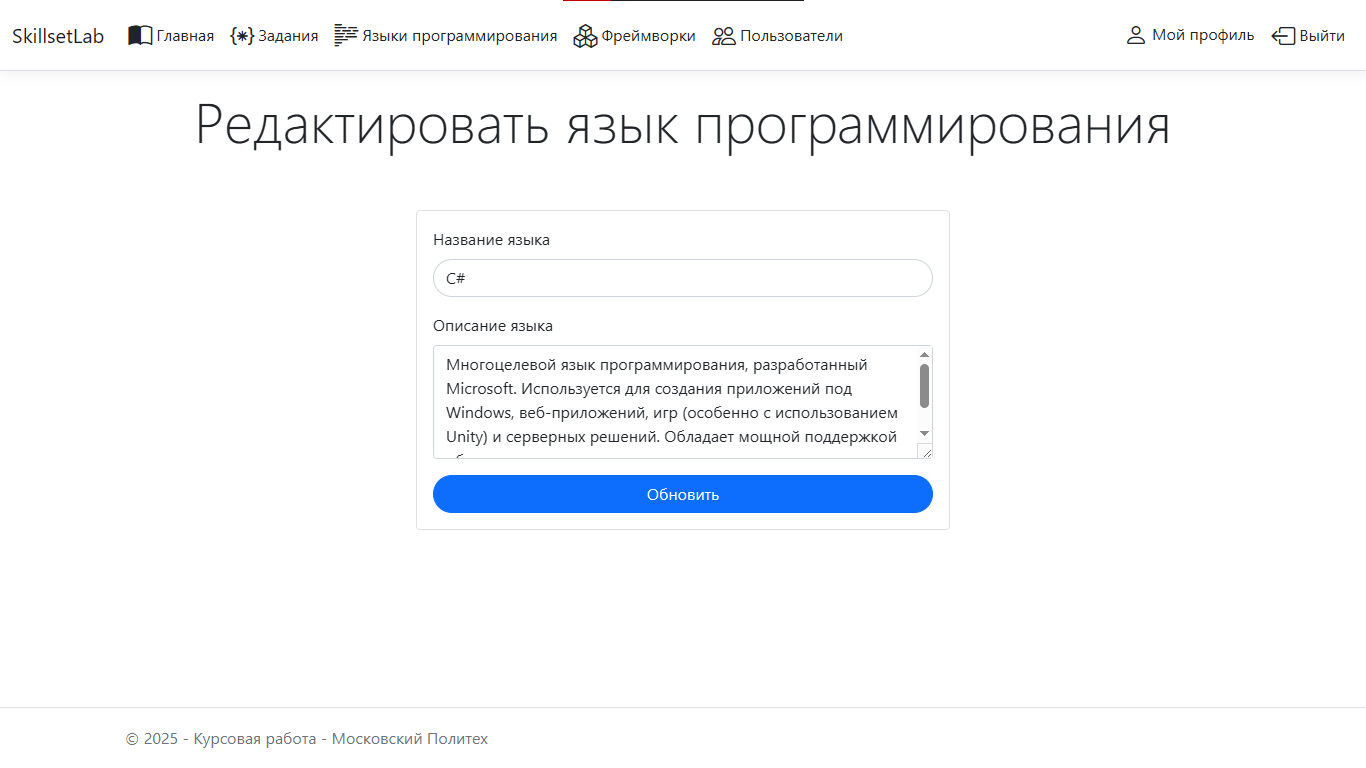
1. **Список языков программирования (ProgrammingLanguages):** Доступен только администраторам и отображает таблицу всех языков программирования с их ID и названием. На странице есть кнопка "Добавить" для создания нового языка, а также кнопки редактирования и удаления для каждого языка.



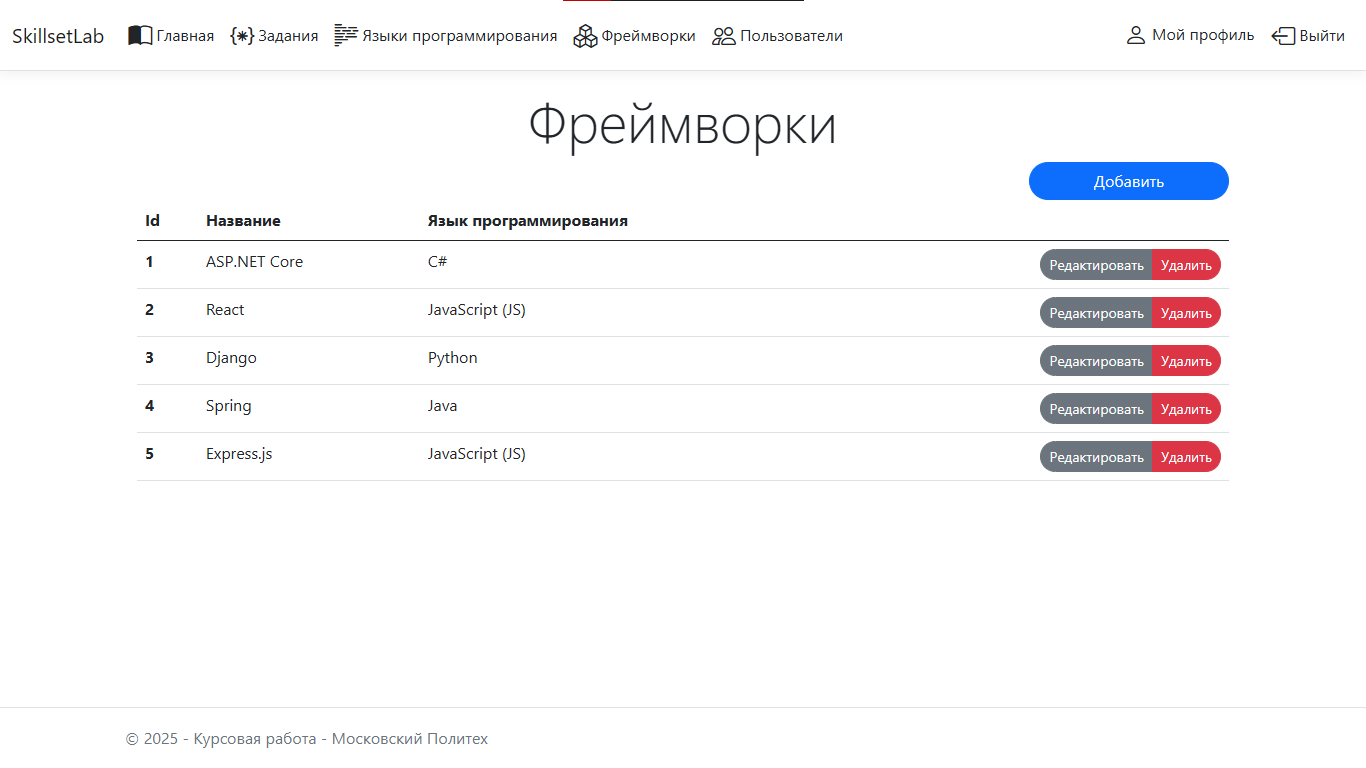
1. **Создание языка программирования (ProgrammingLanguages/Create):** Позволяет администраторам добавлять новый язык программирования в систему. Форма включает поля для ввода названия и описания языка.



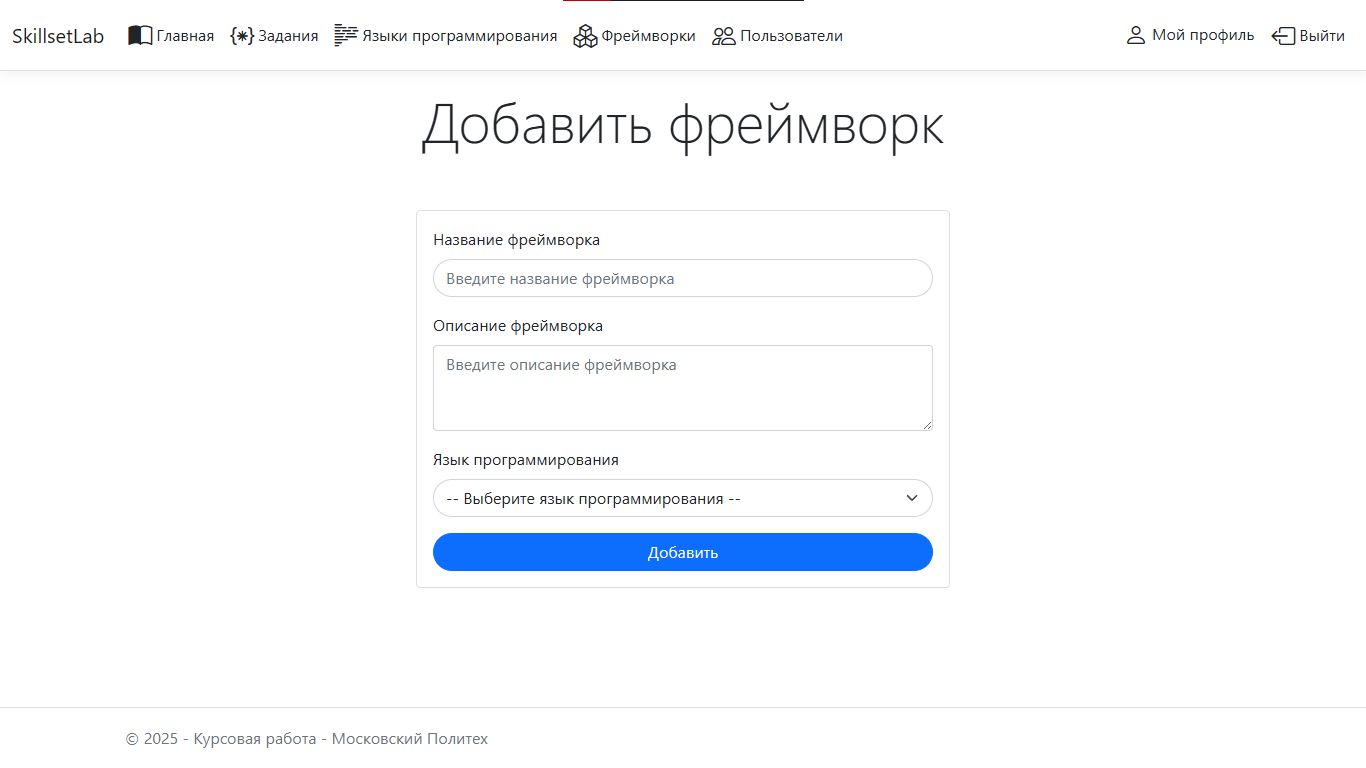
1. **Редактирование языка программирования (ProgrammingLanguages/Update):** Позволяет администраторам обновлять информацию о существующем языке программирования. Форма аналогична форме создания языка и включает поля для изменения названия и описания.



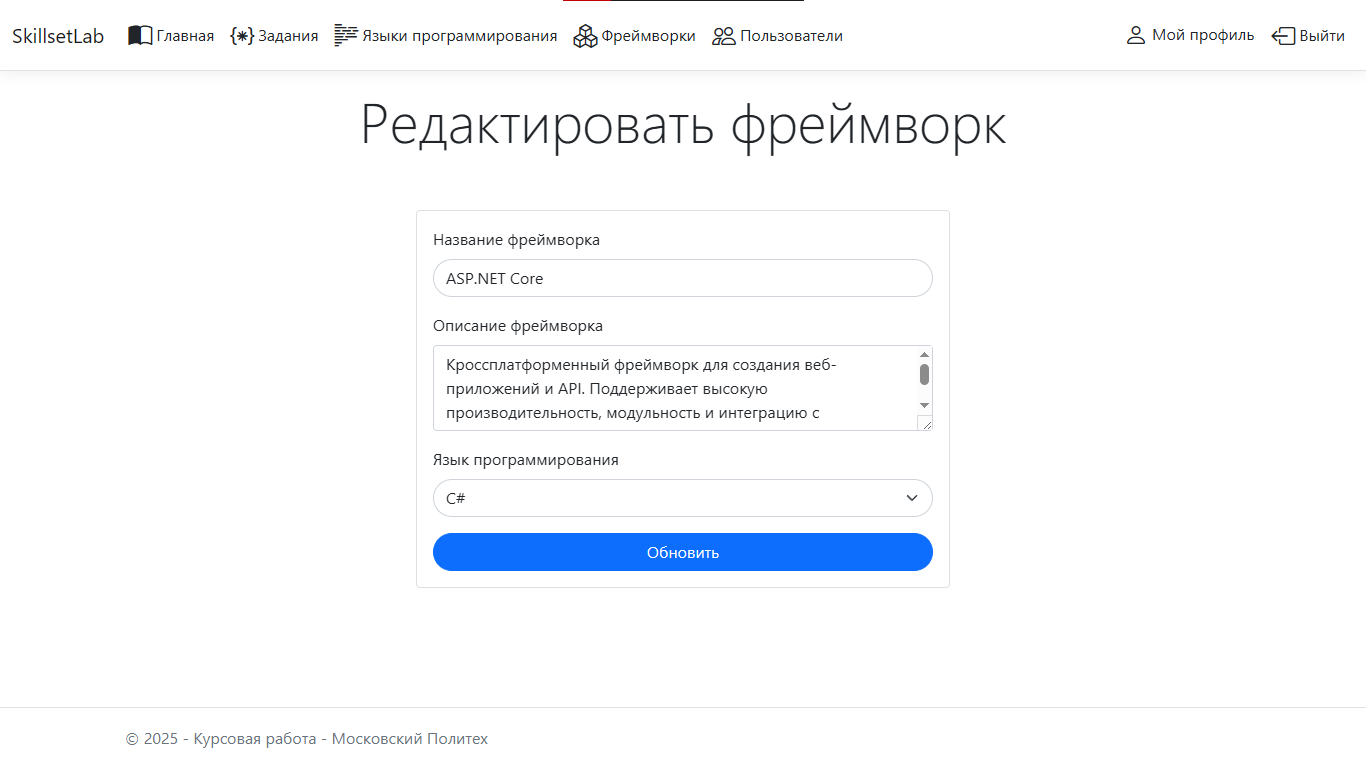
1. **Список фреймворков (Frameworks):** Доступен только администраторам и отображает таблицу всех фреймворков с их ID, названием и привязкой к языку программирования. На странице есть кнопка "Добавить" для создания нового фреймворка, а также кнопки редактирования и удаления для каждого фреймворка.



1. **Создание фреймворка (Frameworks/Create):** Позволяет администраторам добавлять новый фреймворк в систему. Форма включает поля для ввода названия, описания фреймворка и выбора языка программирования из выпадающего списка.



1. **Редактирование фреймворка (Frameworks/Update):** Позволяет администраторам обновлять информацию о существующем фреймворке. Форма аналогична форме создания фреймворка и включает поля для изменения названия, описания и выбора языка программирования.



Это описание страниц интерфейса, которые реализованы в платформе "Практикум по фреймворкам". В зависимости от дальнейшего развития системы могут быть добавлены дополнительные страницы и элементы интерфейса для удовлетворения конкретных потребностей пользователей.

ЗАКЛЮЧЕНИЕ

В ходе разработки образовательной платформы "Практикум по фреймворкам" были достигнуты ключевые цели и задачи, поставленные перед проектом. Созданная система предоставляет эффективные инструменты для практического изучения современных фреймворков через решение реальных задач программирования. В результате проведенной работы можно сделать следующие основные выводы:

1. **Удобство использования:** Разработанная платформа имеет интуитивно понятный интерфейс, что облегчает взаимодействие пользователей с системой. Участники могут легко просматривать задания, отправлять решения, использовать подсказки и отслеживать свой прогресс.
2. **Функциональность:** Система предлагает широкий спектр возможностей, включая создание и решение заданий, систему подсказок с элементами геймификации, оценку решений других пользователей и административный интерфейс для управления контентом. Все эти функции обеспечивают эффективное обучение и взаимодействие между участниками.
3. **Безопасность данных:** Для защиты учетных данных пользователей были реализованы механизмы аутентификации и авторизации, включая использование JWT-токенов и хеширование паролей с помощью алгоритма BCrypt. Это гарантирует безопасность личной информации и защищает систему от несанкционированного доступа.
4. **Гибкость и масштабируемость:** Платформа обладает гибкой архитектурой, что позволяет легко добавлять новые языки программирования, фреймворки и типы заданий. Система также может масштабироваться для поддержки растущего числа пользователей и увеличения объема данных.
5. **Эффективность и производительность:** Благодаря использованию современных технологий, таких как ASP.NET Core, PostgreSQL и облачное хранилище, система обеспечивает быструю обработку запросов и надежное хранение данных, даже при высокой нагрузке.

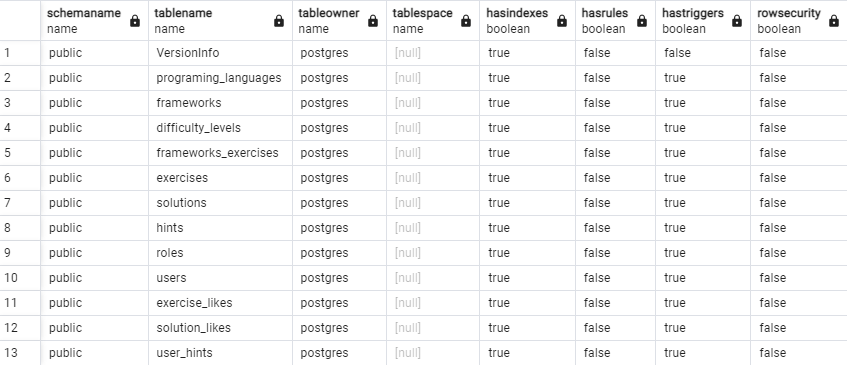
В целом, разработанная образовательная платформа представляет собой мощный инструмент для практического обучения программированию с использованием современных фреймворков. Дальнейшее развитие системы будет направлено на расширение функционала, улучшение производительности и повышение удобства работы для пользователей, что поможет обеспечить максимальную удовлетворенность участников и успех в достижении образовательных целей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фаулер, М. (2018). Чистая архитектура: Искусство разработки программного обеспечения. М.: Питер.
2. Рихтер, Д. (2020). CLR via C#: Программирование на платформе .NET. М.: Вильямс.
3. Клейн, А. (2019). Разработка веб-приложений с использованием ASP.NET Core MVC. СПб: БХВ-Петербург.
4. Шварц, М. (2021). PostgreSQL: Полное руководство по высокопроизводительным базам данных. М.: ДМК Пресс.
5. Ломбарди, Ф. (2018). Dapper: Эффективная работа с базами данных в .NET. М.: Лаборатория знаний.
6. Хилл, Г. (2020). Автоматизация миграций базы данных с помощью FluentMigrator. М.: Техносфера.
7. Брайан, К. (2019). Безопасная аутентификация и авторизация в веб-приложениях: JWT и OAuth 2.0. М.: Альфа-Пресс.
8. Кристиансен, Т. (2020). BCrypt: Надежное хеширование паролей для современных приложений. М.: Научный мир.
9. Уайт, Р. (2021). Bootstrap 5: Создание адаптивных веб-интерфейсов. М.: Диалектика.
10. Закас, Н. (2019). JavaScript для профессионалов: Современные шаблоны и практики. М.: Вильямс.

ПРИЛОЖЕНИЕ

Таблица со списком таблиц

**

Листинг кода

**Program.cs**

*using* Coursework.Database;  
*using* Coursework.Database.Infrastructure;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Interfaces.Services;  
*using* Coursework.Models;  
*using* Coursework.Services;  
*using* Microsoft.AspNetCore.Authentication.JwtBearer;  
*using* Microsoft.IdentityModel.Tokens;  
*using* System.Text;  
  
*var* builder = WebApplication.*CreateBuilder*(args);  
  
Dapper.DefaultTypeMap.MatchNamesWithUnderscores = *true*;  
  
*var* jwtSettings = builder.Configuration.GetSection("JwtSettings").Get<JwtSettings>()   
 ?? *throw new* InvalidOperationException("Не блока настроек JWT в appsettings.json");  
  
builder.Services.AddSingleton(jwtSettings);  
builder.Services.AddScoped<*IJwtService*, JwtService>();  
  
builder.Services.AddAuthentication(options =>  
{  
 options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  
 options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  
})  
.AddJwtBearer(options =>  
{  
 options.TokenValidationParameters = *new* TokenValidationParameters  
 {  
 ValidateIssuer = *true*,  
 ValidateAudience = *true*,  
 ValidateLifetime = *true*,  
 ValidateIssuerSigningKey = *true*,  
 ValidIssuer = jwtSettings.Issuer,  
 ValidAudience = jwtSettings.Audience,  
 IssuerSigningKey = *new* SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings.SecretKey)),  
 ClockSkew = TimeSpan.*Zero* };  
   
 options.Events = *new* JwtBearerEvents  
 {  
 OnMessageReceived = context =>  
 {  
 *if* (context.Request.Cookies.ContainsKey("jwt"))  
 {  
 context.Token = context.Request.Cookies["jwt"];  
 }  
 *return* Task.CompletedTask;  
 }  
 };  
});  
  
builder.Services.AddAuthorizationBuilder()  
 .AddPolicy("AdminOnly", policy => policy.RequireRole("Администратор"))  
 .AddPolicy("UserAccess", policy => policy.RequireRole("Пользователь", "Администратор"));  
  
builder.Services.AddControllersWithViews();  
builder.Services.AddTransient<DbConnectionFactory>();  
builder.Services.AddTransient<*IUnitOfWorkFactory*, UnitOfWorkFactory>();  
*var* connectionString = builder.Configuration.GetConnectionString("MainDbConnection");  
  
builder.Services.AddScoped<*IPasswordService*, PasswordService>();  
  
*var* app = builder.Build();  
  
*var* logger = app.Services.GetRequiredService<*ILogger*<Program>>();  
MigrationRunner.*RunMigrations*(connectionString, logger);  
  
*if* (!app.Environment.IsDevelopment())  
{  
 app.UseExceptionHandler("/Home/Error");  
 app.UseBrowserLink();  
 app.UseHsts();  
}  
  
app.UseHttpsRedirection();  
app.UseStaticFiles();  
  
app.UseRouting();  
  
app.UseAuthentication();  
app.UseAuthorization();  
  
app.MapControllerRoute(  
 name: "default",  
 pattern: "{controller=Home}/{action=Index}/{id?}");  
  
app.Run();

**\_ViewImports.cshtml**

@*using* Coursework  
@*using* Coursework.Models  
@*addTagHelper* \*, Microsoft.AspNetCore.Mvc.TagHelpers

**\_ViewStart.cshtml**

@{  
 Layout = "\_Layout";  
}

**appsettings.json**

{  
 "ConnectionStrings": {  
 "MainDbConnection":"Host=localhost;Database=Coursework;user id=postgres;password=postgres"  
 },  
 "JwtSettings": {  
 "SecretKey": "YourSuperSecretKey123!@#$%^&\*()\_+",  
 "Issuer": "CourseWorkApp",  
 "Audience": "CourseWorkUsers",  
 "ExpiryInMinutes": 60  
 },  
 "Logging": {  
 "LogLevel": {  
 "Default": "Information",  
 "Microsoft.AspNetCore": "Warning"  
 }  
 },  
 "AllowedHosts": "\*"  
}

**CSS/Site.css**

html {  
 font-size: 14px;  
}  
  
*@media* (min-width: 768px) {  
 html {  
 font-size: 16px;  
 }  
}  
  
.btn:focus, .btn:active:focus, .btn-link.nav-link:focus, .form-control:focus, .form-check-input:focus {  
 box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;  
}  
  
html {  
 position: relative;  
 min-height: 100%;  
}  
  
body {  
 margin-bottom: 60px;  
}  
  
.tooltip-inner {  
 max-width: 600px;  
 white-space: normal;  
 text-align: left;  
 background-color: var(--bs-white);  
 color: var(--bs-body-color);  
}  
  
  
.card-body {  
 display: flex;  
 flex-direction: column;  
}  
  
.card-text::-webkit-scrollbar, .card-title::-webkit-scrollbar {  
 width: 0;  
}  
  
.mt-auto {  
 margin-top: auto;  
}  
  
.btn-group.rounded-pill > :first-child,  
.input-group.rounded-pill > :first-child {  
 border-top-left-radius: 50rem;  
 border-bottom-left-radius: 50rem;  
}  
  
.btn-group.rounded-pill > :last-child,  
.input-group.rounded-pill > :last-child {  
 border-top-right-radius: 50rem;  
 border-bottom-right-radius: 50rem;  
}  
  
/\* Стили для аккордеона подсказок \*/  
#hintsAccordion .accordion-button:not(.collapsed) {  
 background-color: #f8f9fa;  
 color: #212529;  
}  
  
#hintsAccordion .accordion-button:focus {  
 box-shadow: none;  
 border-color: rgba(0,0,0,.125);  
}  
  
#hintsAccordion .accordion-item {  
 margin-bottom: 0.5rem;  
 border-radius: 0.375rem;  
 overflow: hidden;  
}  
  
#hintsAccordion .accordion-body {  
 white-space: pre-wrap;  
}  
  
.navbar .bi {  
 font-size: 1.5em;  
}  
  
.hide-accordion-arrow .accordion-button::after {  
 display: none *!important*;  
}

**Shared/\_Layout.cshtml**

@*using* Coursework.Extensions  
<!DOCTYPE *html*>  
<html *lang*="ru">  
<head>  
 <meta *charset*="utf-8" />  
 <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0" />  
 <title>@ViewData["Title"] - Курсовая работа</title>  
 <link *rel*="stylesheet" *href*="~/lib/bootstrap/dist/css/bootstrap.min.css" />  
 <*link rel*="stylesheet" *href*="~/css/site.css" *asp-append-version*="true" />  
 <*link rel*="stylesheet" *href*="~/Coursework.styles.css" *asp-append-version*="true" />  
 <!-- Bootstrap Icons CDN -->  
 <link *rel*="stylesheet" *href*="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">  
</head>  
<body>  
 <header>  
 <nav *class*="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">  
 <div *class*="container-fluid">  
 <*a class*="navbar-brand" *asp-area*="" *asp-controller*="Home" *asp-action*="Index">SkillsetLab</*a*>  
 <button *class*="navbar-toggler" *type*="button" *data-bs-toggle*="collapse" *data-bs-target*=".navbar-collapse" *aria-controls*="navbarSupportedContent"  
 *aria-expanded*="false" *aria-label*="Toggle navigation">  
 <span *class*="navbar-toggler-icon"></span>  
 </button>  
 <div *class*="navbar-collapse collapse d-sm-inline-flex justify-content-between">  
 <ul *class*="navbar-nav flex-grow-1">  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-area*="" *asp-controller*="Home" *asp-action*="Index">  
 <i *class*="bi bi-book-half align-middle me-1"></i><span *class*="align-middle">Главная</span>  
 </*a*>  
 </li>  
 @*if* (User.Identity?.IsAuthenticated == *true*)  
 {  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-area*="" *asp-controller*="Exercises" *asp-action*="Index">  
 <i *class*="bi bi-braces-asterisk align-middle me-1"></i><span *class*="align-middle">Задания</span>  
 </*a*>  
 </li>  
 }  
 @*if* (User.IsInRole("Администратор"))  
 {  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-area*="" *asp-controller*="ProgrammingLanguages" *asp-action*="Index">  
 <i *class*="bi bi-body-text align-middle me-1"></i><span *class*="align-middle">Языки программирования</span>  
 </*a*>  
 </li>  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-area*="" *asp-controller*="Frameworks" *asp-action*="Index">  
 <i *class*="bi bi-boxes align-middle me-1"></i><span *class*="align-middle">Фреймворки</span>  
 </*a*>  
 </li>  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-area*="" *asp-controller*="User" *asp-action*="Index">  
 <i *class*="bi bi-people align-middle me-1"></i><span *class*="align-middle">Пользователи</span>  
 </*a*>  
 </li>  
 }  
 </ul>  
 <ul *class*="navbar-nav">  
 @*if* (User.Identity?.IsAuthenticated == *true*)  
 {  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-controller*="User" *asp-action*="View" *asp-route-id*="@User.GetId()">  
 <i *class*="bi bi-person align-middle me-1"></i><span *class*="align-middle">Мой профиль</span>  
 </*a*>  
 </li>  
 <li *class*="nav-item">  
 <*form asp-controller*="Auth" *asp-action*="Logout" *method*="post">  
 <button *type*="submit" *class*="btn btn-link nav-link text-dark">  
 <i *class*="bi bi-box-arrow-left align-middle me-1"></i><span *class*="align-middle">Выйти</span>  
 </button>  
 </*form*>  
 </li>  
 }  
 *else* {  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-controller*="Auth" *asp-action*="Login">  
 <i *class*="bi bi-door-open align-middle me-1"></i><span *class*="align-middle">Войти</span>  
 </*a*>  
 </li>  
 <li *class*="nav-item">  
 <*a class*="nav-link text-dark" *asp-controller*="Auth" *asp-action*="Register">  
 <i *class*="bi bi-box-arrow-in-right align-middle me-1"></i><span *class*="align-middle">Регистрация</span>  
 </*a*>  
 </li>  
 }  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
 <div *class*="container">  
 <main *role*="main" *class*="pb-3">  
 @RenderBody()  
 </main>  
 </div>  
  
 <footer *class*="border-top footer text-muted">  
 <div *class*="container">  
 &copy; 2025 - Курсовая работа - Московский Политех  
 </div>  
 </footer>  
 <script *src*="~/lib/jquery/dist/jquery.min.js"></script>  
 <script *src*="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>  
 <*script src*="~/js/site.js" *asp-append-version*="true"></*script*>  
 @*await* RenderSectionAsync("Scripts", required: *false*)  
</body>  
</html>

**Shared/\_ValidationScriptsPartial.cshtml**

<script *src*="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>  
<script *src*="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

**Shared/\_Layout.cshtml.css**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification  
for details on configuring this project to bundle and minify static web assets. \*/  
  
a.navbar-brand {  
 white-space: normal;  
 text-align: center;  
 word-break: break-all;  
}  
  
a {  
 color: #0077cc;  
}  
  
.btn-primary {  
 color: #fff;  
 background-color: #1b6ec2;  
 border-color: #1861ac;  
}  
  
.nav-pills .nav-link.active, .nav-pills .show > .nav-link {  
 color: #fff;  
 background-color: #1b6ec2;  
 border-color: #1861ac;  
}  
  
.border-top {  
 border-top: 1px solid #e5e5e5;  
}  
.border-bottom {  
 border-bottom: 1px solid #e5e5e5;  
}  
  
.box-shadow {  
 box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);  
}  
  
button.accept-policy {  
 font-size: 1rem;  
 line-height: inherit;  
}  
  
.footer {  
 position: absolute;  
 bottom: 0;  
 width: 100%;  
 white-space: nowrap;  
 line-height: 60px;  
}

**Shared/\_AuthorLink.cshtml**

@*model* Coursework.Models.Entities.User  
  
@*if* (Model *is not null*)  
{  
 <*a asp-controller*="User" *asp-action*="View" *asp-route-id*="@Model.Id" *class*="text-primary" *style*="text-decoration: none; cursor: pointer;">  
 @Model.Name  
 </*a*>  
}  
*else*{  
 <span>Неизвестный автор</span>  
}

**Shared/\_ExerciseLink.cshtml**

@*model* Coursework.Models.Entities.Exercise  
  
@*if* (Model *is not null*)  
{  
<*a asp-controller*="Exercises" *asp-action*="View" *asp-route-id*="@Model.Id" *class*="text-primary" *style*="text-decoration: none; cursor: pointer;">  
 @Model.Name  
</*a*>  
}  
*else*{  
<span>Неизвестное задание</span>  
}

**Home/Index.cshtml**

@{  
 ViewData["Title"] = "Главная";  
}  
  
<div *class*="container py-5">  
 <div *class*="row justify-content-center mb-5">  
 <div *class*="col-md-10 text-center">  
 <h1 *class*="display-4 fw-bold mb-4">Практикум по фреймворкам</h1>  
 <p *class*="lead mb-4">Интерактивная платформа для изучения и практики современных фреймворков через решение реальных задач</p>  
 <div *class*="d-flex justify-content-center gap-3">  
 @*if* (User.Identity?.IsAuthenticated == *true*)  
 {  
 <*a asp-controller*="Exercises" *asp-action*="Index" *class*="btn btn-outline-secondary btn-lg">Просмотреть задания</*a*>  
 }  
 *else* {  
 <*a asp-controller*="Auth" *asp-action*="Register" *class*="btn btn-primary btn-lg">Начать обучение</*a*>  
 }  
 </div>  
 </div>  
 </div>  
  
 <div *class*="row mb-5">  
 <div *class*="col-md-4 mb-4">  
 <div *class*="card h-100 shadow-sm">  
 <div *class*="card-body text-center">  
 <i *class*="bi bi-code-square text-primary" *style*="font-size: 3rem;"></i>  
 <h3 *class*="card-title mt-3">Практические задания</h3>  
 <p *class*="card-text">Решайте реальные задачи по разработке с использованием популярных фреймворков. Каждое задание сопровождается начальным кодом и автоматическими тестами.</p>  
 </div>  
 </div>  
 </div>  
 <div *class*="col-md-4 mb-4">  
 <div *class*="card h-100 shadow-sm">  
 <div *class*="card-body text-center">  
 <i *class*="bi bi-lightbulb text-primary" *style*="font-size: 3rem;"></i>  
 <h3 *class*="card-title mt-3">Система подсказок</h3>  
 <p *class*="card-text">Зарабатывайте очки за решение заданий и используйте их для получения подсказок, когда столкнетесь с трудностями. Учитесь в своем темпе.</p>  
 </div>  
 </div>  
 </div>  
 <div *class*="col-md-4 mb-4">  
 <div *class*="card h-100 shadow-sm">  
 <div *class*="card-body text-center">  
 <i *class*="bi bi-people text-primary" *style*="font-size: 3rem;"></i>  
 <h3 *class*="card-title mt-3">Сообщество</h3>  
 <p *class*="card-text">Создавайте собственные задания, делитесь решениями и учитесь у других. Развивайте свои навыки вместе с сообществом разработчиков.</p>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div *class*="row mb-5">  
 <div *class*="col-md-6 offset-md-3">  
 <div *class*="card bg-light">  
 <div *class*="card-body">  
 <h4 *class*="card-title text-center mb-3">Как это работает?</h4>  
 <ol *class*="list-group list-group-numbered mb-0">  
 <li *class*="list-group-item border-0 bg-transparent">Зарегистрируйтесь на платформе</li>  
 <li *class*="list-group-item border-0 bg-transparent">Выберите задание по интересующему фреймворку</li>  
 <li *class*="list-group-item border-0 bg-transparent">Скачайте начальный код и ознакомьтесь с требованиями</li>  
 <li *class*="list-group-item border-0 bg-transparent">Разработайте решение и загрузите его на платформу</li>  
 <li *class*="list-group-item border-0 bg-transparent">Получите результаты автоматической проверки и заработайте очки</li>  
 </ol>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div *class*="row">  
 <div *class*="col-12 text-center">  
 <div *class*="card bg-primary text-white">  
 <div *class*="card-body py-4">  
 <h2 *class*="card-title mb-3">О проекте</h2>  
 <p *class*="card-text mb-4">Данный проект разработан в рамках курсовой работы и представляет собой образовательную платформу, направленную на практическое изучение фреймворков через решение реальных задач разработки.</p>  
 <p *class*="mb-0"><strong>Автор:</strong> Ежов Дмитрий Андреевич</p>  
 <p *class*="mb-0"><strong>Группа:</strong> 233-322</p>  
 <p *class*="mb-0"><strong>Год:</strong> 2025</p>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>

**ProgrammingLanguages/Create.cshtml**

@*model* Coursework.Models.Entities.ProgrammingLanguage  
  
@{  
 ViewData["Title"] = "Добавить язык программирования";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Добавить язык программирования</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="ProgrammingLanguages" *asp-action*="Create">  
 <div *class*="mb-3">  
 <label *for*="Name" *class*="form-label">Название языка</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="Name" *asp-for*="Name" *placeholder*="Введите название языка">  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <label *for*="Description" *class*="form-label">Описание языка</label>  
 <*textarea class*="form-control rounded" *id*="Description" *asp-for*="Description" *rows*="3" *placeholder*="Введите описание языка"></*textarea*>  
 <*span asp-validation-for*="Description" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Добавить</button>  
 </div>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**ProgrammingLanguages/Index.cshtml**

@*model* Coursework.Models.Entities.ProgrammingLanguage  
  
@{  
 ViewData["Title"] = "Список языков программирования";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Языки программирования</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="d-grid d-md-flex justify-content-md-end">  
 <*a   
 class*="btn btn-primary rounded-pill"  
 *style*="width: 200px;"  
 *asp-controller*="ProgrammingLanguages"  
 *asp-action*="Create">Добавить</*a*>  
 </div>  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Id</th>  
 <th *scope*="col">Название</th>  
 <th *scope*="col" *class*="text-end"></th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* language *in* ViewBag.Languages)  
 {  
 <tr *data-bs-toggle*="tooltip" *title*="@language.Description">  
 <th *scope*="row">@language.Id</th>  
 <td>@language.Name</td>  
 <td *class*="text-end">  
 <*form   
 method*="post"   
 *asp-controller*="ProgrammingLanguages"   
 *asp-action*="Delete"   
 *asp-route-id*="@language.Id"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить этот язык программирования?');">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="ProgrammingLanguages"  
 *asp-action*="Update"  
 *asp-route-id*="@language.Id">Редактировать</*a*>  
 <button *type*="submit" *class*="btn btn-sm btn-danger">Удалить</button>  
 </div>  
 </*form*>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
</div>

**ProgrammingLanguages/Update.cshtml**

@*model* Coursework.Models.Entities.ProgrammingLanguage  
  
@{  
 ViewData["Title"] = "Редактировать язык программирования";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Редактировать язык программирования</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="ProgrammingLanguages" *asp-action*="Update">  
 <div *class*="mb-3">  
 <label *for*="Name" *class*="form-label">Название языка</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="Name" *asp-for*="Name" *placeholder*="Введите название языка">  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <label *for*="Description" *class*="form-label">Описание языка</label>  
 <*textarea class*="form-control rounded" *id*="Description" *asp-for*="Description" *rows*="3" *placeholder*="Введите описание языка"></*textarea*>  
 <*span asp-validation-for*="Description" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Обновить</button>  
 </div>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**Frameworks/Create.cshtml**

@*model* Coursework.Models.Entities.Framework  
  
@{  
 ViewData["Title"] = "Добавить фреймворк";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Добавить фреймворк</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="Frameworks" *asp-action*="Create">  
 <div *class*="mb-3">  
 <label *for*="Name" *class*="form-label">Название фреймворка</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="Name" *asp-for*="Name" *placeholder*="Введите название фреймворка">  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <label *for*="Description" *class*="form-label">Описание фреймворка</label>  
 <*textarea class*="form-control rounded" *id*="Description" *asp-for*="Description" *rows*="3" *placeholder*="Введите описание фреймворка"></*textarea*>  
 <*span asp-validation-for*="Description" *class*="text-danger"></*span*>  
 </div>  
 <div *class*="mb-3">  
 <label *for*="LanguageId" *class*="form-label">Язык программирования</label>  
 <*select class*="form-select rounded-pill" *id*="LanguageId" *asp-for*="LanguageId">  
 <*option value*="">-- Выберите язык программирования --</*option*>  
 @*foreach* (*var* language *in* ViewBag.Languages)  
 {  
 <*option value*="@language.Id" *title*="@language.Description">@language.Name</*option*>  
 }  
 </*select*>  
 <*span asp-validation-for*="LanguageId" *class*="text-danger"></*span*>  
 </div>  
   
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Добавить</button>  
 </div>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**Frameworks/Index.cshtml**

@*model* Coursework.Models.Entities.Framework  
  
@{  
 ViewData["Title"] = "Список фреймворков";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Фреймворки</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="d-grid d-md-flex justify-content-md-end">  
 <*a   
 class*="btn btn-primary rounded-pill"  
 *style*="width: 200px;"  
 *asp-controller*="Frameworks"  
 *asp-action*="Create">Добавить</*a*>  
 </div>  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Id</th>  
 <th *scope*="col">Название</th>  
 <th *scope*="col">Язык программирования</th>  
 <th *scope*="col" *class*="text-end"></th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* framework *in* ViewBag.Frameworks)  
 {  
 <tr *data-bs-toggle*="tooltip" *title*="@framework.Description">  
 <th *scope*="row">@framework.Id</th>  
 <td>@framework.Name</td>  
 <td>@framework.Language.Name</td>  
 <td *class*="text-end">  
 <*form  
 method*="post"  
 *asp-controller*="Frameworks"  
 *asp-action*="Delete"  
 *asp-route-id*="@framework.Id"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить этот фреймворк?');">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="Frameworks"  
 *asp-action*="Update"  
 *asp-route-id*="@framework.Id">Редактировать</*a*>  
 <button *type*="submit" *class*="btn btn-sm btn-danger">Удалить</button>  
 </div>  
 </*form*>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
</div>

**Frameworks/Update.cshtml**

@*model* Coursework.Models.Entities.Framework  
  
@{  
 ViewData["Title"] = "Редактировать фреймворк";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Редактировать фреймворк</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="Frameworks" *asp-action*="Update">  
 <div *class*="mb-3">  
 <label *for*="Name" *class*="form-label">Название фреймворка</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="Name" *asp-for*="Name" *placeholder*="Введите название фреймворка">  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <label *for*="Description" *class*="form-label">Описание фреймворка</label>  
 <*textarea class*="form-control rounded" *id*="Description" *asp-for*="Description" *rows*="3" *placeholder*="Введите описание фреймворка"></*textarea*>  
 <*span asp-validation-for*="Description" *class*="text-danger"></*span*>  
 </div>  
 <div *class*="mb-3">  
 <label *for*="LanguageId" *class*="form-label">Язык программирования</label>  
 <*select class*="form-select rounded-pill" *id*="LanguageId" *asp-for*="LanguageId">  
 <*option value*="">-- Выберите язык программирования --</*option*>  
 @*foreach* (*var* language *in* ViewBag.Languages)  
 {  
 <*option value*="@language.Id" *title*="@language.Description">@language.Name</*option*>  
 }  
 </*select*>  
 <*span asp-validation-for*="LanguageId" *class*="text-danger"></*span*>  
 </div>  
   
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Обновить</button>  
 </div>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**Exercises/Create.cshtml**

<!-- Уровень сложности -->  
 <div *class*="mb-3">  
 <label *for*="DifficultyId" *class*="form-label">Уровень сложности</label>  
 <*select class*="form-select rounded-pill" *id*="DifficultyId" *asp-for*="DifficultyId">  
 <*option value*="">-- Выберите уровень сложности --</*option*>  
 @*foreach* (*var* difficulty *in* (ViewBag.DifficultyLevels *as IEnumerable*<DifficultyLevel>)!)  
 {  
 <*option value*="@difficulty.Id">@difficulty.Name</*option*>  
 }  
 </*select*>  
 <*span asp-validation-for*="DifficultyId" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Количество баллов -->  
 <div *class*="mb-3">  
 <label *for*="Score" *class*="form-label">Количество очков</label>  
 <*input type*="number" *class*="form-control rounded-pill" *id*="Score" *asp-for*="Score" *placeholder*="Введите количество очков">  
 <*span asp-validation-for*="Score" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Язык программирования -->  
 <div *class*="mb-3">  
 <label *for*="LanguageId" *class*="form-label">Язык программирования</label>  
 <*select class*="form-select rounded-pill" *id*="LanguageId" *asp-for*="LanguageId">  
 <*option value*="">-- Выберите язык программирования --</*option*>  
 @*foreach* (*var* language *in* ViewBag.Languages)  
 {  
 <*option value*="@language.Id" *title*="@language.Description">@language.Name</*option*>  
 }  
 </*select*>  
 <*span asp-validation-for*="LanguageId" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Краткое описание -->  
 <div *class*="mb-3">  
 <label *for*="ShortDescription" *class*="form-label">Короткое описание</label>  
 <*textarea class*="form-control rounded" *id*="ShortDescription" *asp-for*="ShortDescription" *rows*="2" *placeholder*="Введите краткое описание"></*textarea*>  
 <*span asp-validation-for*="ShortDescription" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Полное описание -->  
 <div *class*="mb-3">  
 <label *for*="FullDescription" *class*="form-label">Задание</label>  
 <*textarea class*="form-control rounded" *id*="FullDescription" *asp-for*="FullDescription" *rows*="4" *placeholder*="Введите текст задания"></*textarea*>  
 <*span asp-validation-for*="FullDescription" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Ключ S3 для материалов -->  
 <div *class*="mb-3">  
 <label *for*="S3KeySource" *class*="form-label">Ключ S3 для материалов упражнения</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3KeySource" *asp-for*="S3KeySource" *placeholder*="Введите ключ S3 для материалов">  
 <*span asp-validation-for*="S3KeySource" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Ключ S3 для тестов -->  
 <div *class*="mb-3">  
 <label *for*="S3KeyTests" *class*="form-label">Ключ S3 для тестов</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3KeyTests" *asp-for*="S3KeyTests" *placeholder*="Введите ключ S3 для тестов">  
 <*span asp-validation-for*="S3KeyTests" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Ключ S3 для решения автора -->  
 <div *class*="mb-3">  
 <label *for*="S3KeyTests" *class*="form-label">Ключ S3 для решения</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3KeyAuthorSolution" *asp-for*="S3KeyAuthorSolution" *placeholder*="Введите ключ S3 для тестов">  
 <*span asp-validation-for*="S3KeyAuthorSolution" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Выбор фреймворков -->  
 <div *class*="mb-3">  
 <label *for*="FrameworkId" *class*="form-label">Фреймворки</label>  
 <div *class*="input-group">  
 <select *class*="form-select rounded-pill" *id*="FrameworkId" *name*="FrameworkId">  
 <*option value*="">-- Выберите фреймворк --</*option*>  
 @*foreach* (*var* framework *in* ViewBag.Frameworks)  
 {  
 <*option value*="@framework.Id" *title*="@framework.Description">@framework.Name</*option*>  
 }  
 </select>  
 <button *type*="button" *class*="btn btn-primary rounded-pill ms-2" *onclick*="*addFrameworkButton*()">Добавить</button>  
 </div>  
 </div>  
  
 <!-- Список выбранных фреймворков -->  
 <div *id*="selectedFrameworks" *class*="mt-2">  
 <!-- Фреймворки будут добавляться сюда -->  
 </div>  
  
 <!-- Добавление подсказок -->  
 <div *class*="mb-3">  
 <div *class*="card p-3">  
 <h5 *class*="card-title">Добавить подсказку</h5>  
 <div *class*="mb-3">  
 <label *for*="HintCost" *class*="form-label">Очки</label>  
 <input *type*="number" *class*="form-control rounded-pill" *id*="HintCost" *placeholder*="Введите стоимость подсказки">  
 </div>  
 <div *class*="mb-3">  
 <label *for*="HintText" *class*="form-label">Текст подсказки</label>  
 <textarea *class*="form-control rounded" *id*="HintText" *rows*="2" *placeholder*="Введите текст подсказки"></textarea>  
 </div>  
 <button *type*="button" *class*="btn btn-primary rounded-pill" *onclick*="*addHintButton*()">Добавить подсказку</button>  
 </div>  
 </div>  
  
 <!-- Список добавленных подсказок -->  
 <div *id*="hints" *class*="mt-2">  
 <!-- Подсказки будут добавляться сюда -->  
 </div>  
  
 <!-- Кнопки сохранения -->  
 <div *class*="d-flex gap-2 mt-4">  
 <button *type*="submit" *class*="btn btn-secondary rounded-pill flex-grow-1" *onclick*="*setIsPublished*(*false*)">Сохранить</button>  
 <button *type*="submit" *class*="btn btn-success rounded-pill flex-grow-1" *onclick*="*setIsPublished*(*true*)">Опубликовать</button>  
 </div>  
  
 <!-- Скрытое поле для IsPublished -->  
 <input *type*="hidden" *id*="IsPublished" *name*="IsPublished" *value*="false" />  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial"/>  
  
 <script>  
 *let selectedFrameworks* = [];  
 *let hints* = [];  
   
 *function addFrameworkButton*() {  
 *const* select = *document*.getElementById('FrameworkId');  
 *const* selectedOption = select.options[select.selectedIndex];  
 *const* framework = {  
 id: selectedOption.value,  
 name: selectedOption.text  
 };  
  
 *if* (!framework.id) {  
 *alert*('Выберите фреймворк!');  
 *return*;  
 }  
  
 *if* (*selectedFrameworks*.some(f => f.id === framework.id)) {  
 *alert*('Этот фреймворк уже добавлен!');  
 *return*;  
 }  
  
 *selectedFrameworks*.push(framework);  
 *renderSelectedFrameworks*();  
 }  
  
 *function renderSelectedFrameworks*() {  
 *const* container = *document*.getElementById('selectedFrameworks');  
 container.innerHTML = '';  
  
 *selectedFrameworks*.forEach((framework, index) => {  
 *const* div = *document*.createElement('div');  
 div.className = 'd-flex align-items-center mb-2 rounded-pill';  
  
 *const* span = *document*.createElement('span');  
 span.className = 'me-2';  
 span.textContent = framework.name;  
  
 *const* button = *document*.createElement('button');  
 button.type = 'button';  
 button.className = 'btn btn-sm btn-danger rounded-pill';  
 button.textContent = 'Убрать';  
 button.onclick = () => *removeFramework*(index);  
  
 div.appendChild(span);  
 div.appendChild(button);  
  
 container.appendChild(div);  
 });  
 }  
  
 *function removeFramework*(index) {  
 *selectedFrameworks*.splice(index, 1);  
 *renderSelectedFrameworks*();  
 }  
  
 *function addHintButton*() {  
 *const* cost = *document*.getElementById('HintCost').value;  
 *const* text = *document*.getElementById('HintText').value;  
  
 *if* (!cost || !text) {  
 *alert*('Пожалуйста, заполните все поля подсказки!');  
 *return*;  
 }  
  
 *if* (*hints*.some(h => h.text === text)) {  
 *alert*('Такая подсказка уже добавлена!');  
 *return*;  
 }  
  
 *const* hint = {  
 cost: cost,  
 text: text  
 };  
  
 *hints*.push(hint);  
 *renderHints*();  
   
 *document*.getElementById('HintCost').value = '';  
 *document*.getElementById('HintText').value = '';  
 }  
  
 *function renderHints*() {  
 *const* container = *document*.getElementById('hints');  
 container.innerHTML = '';  
  
 *hints*.forEach((hint, index) => {  
 *const* card = *document*.createElement('div');  
 card.className = 'card mb-3';  
   
 *const* cardHeader = *document*.createElement('div');  
 cardHeader.className = 'card-header d-flex justify-content-between align-items-center';  
 cardHeader.innerHTML = `<strong>Подсказка за ${hint.cost} очков</strong>`;  
   
 *const* removeButton = *document*.createElement('button');  
 removeButton.type = 'button';  
 removeButton.className = 'btn btn-sm btn-danger';  
 removeButton.textContent = 'Убрать';  
 removeButton.onclick = () => *removeHint*(index);  
 cardHeader.appendChild(removeButton);  
   
 *const* cardBody = *document*.createElement('div');  
 cardBody.className = 'card-body';  
   
 *const* hintText = *document*.createElement('p');  
 hintText.className = 'card-text';  
 hintText.textContent = hint.text;  
   
 cardBody.appendChild(hintText);  
 card.appendChild(cardHeader);  
 card.appendChild(cardBody);  
   
 container.appendChild(card);  
 });  
 }  
   
 *function removeHint*(index) {  
 *hints*.splice(index, 1);  
 *renderHints*();  
 }  
   
 *function setIsPublished*(value) {  
 *document*.getElementById('IsPublished').value = value;  
 }  
  
 // Создание скрытых инпутов перед отправкой формы  
 *document*.getElementById('exerciseForm').addEventListener('submit', *function* () {  
 *const* form = *this*;  
  
 *const* oldFrameworkInputs = form.querySelectorAll('input[name^="Frameworks"]');  
 *const* oldHintInputs = form.querySelectorAll('input[name^="Hints"]');  
 [...oldFrameworkInputs, ...oldHintInputs].forEach(input => input.remove());  
   
 *selectedFrameworks*.forEach((framework, index) => {  
 *const* idInput = *document*.createElement('input');  
 idInput.type = 'hidden';  
 idInput.name = `Frameworks[${index}].Id`;  
 idInput.value = framework.id;  
  
 *const* nameInput = *document*.createElement('input');  
 nameInput.type = 'hidden';  
 nameInput.name = `Frameworks[${index}].Name`;  
 nameInput.value = framework.name;  
  
 form.appendChild(idInput);  
 form.appendChild(nameInput);  
 });  
  
 *hints*.forEach((hint, index) => {  
 *const* scoreInput = *document*.createElement('input');  
 scoreInput.type = 'hidden';  
 scoreInput.name = `Hints[${index}].Cost`;  
 scoreInput.value = hint.cost;  
  
 *const* textInput = *document*.createElement('input');  
 textInput.type = 'hidden';  
 textInput.name = `Hints[${index}].Text`;  
 textInput.value = hint.text;  
  
 form.appendChild(scoreInput);  
 form.appendChild(textInput);  
 });  
 });  
   
 *document*.addEventListener('DOMContentLoaded', *function* () {  
 @*if* (Model *is not null*)  
 {  
 @*foreach* (var framework *in* Model.Frameworks)  
 {  
 <text>*selectedFrameworks*.push({  
 id: '@framework.Id',  
 name: '@framework.Name'  
 });</text>  
 }  
  
 @*foreach* (var hint *in* Model.Hints)  
 {  
 <text>*hints*.push({  
 cost: '@hint.Id',  
 text: '@hint.Text'  
 });</text>  
 }  
 }  
 *renderSelectedFrameworks*();  
 *renderHints*();  
 });  
 </script>  
}

**Exercises/Index.cshtml**

@*using* Coursework.Extensions  
@*model* Coursework.Models.DTOs.ExerciseDto  
  
@{  
 ViewData["Title"] = "Список заданий";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Задания</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row">  
 <div *class*="d-grid d-md-flex justify-content-between">  
 <*form asp-controller*="Exercises" *asp-action*="Index" *method*="get" *class*="w-25">  
 <input  
 *type*="text"  
 *name*="search"  
 *class*="form-control rounded-pill"  
 *placeholder*="Поиск..."  
 *value*="@ViewBag.Search"  
 *aria-label*="Search">  
 </*form*>  
 <*a  
 class*="btn btn-primary rounded-pill"  
 *style*="width: 200px;"  
 *asp-controller*="Exercises"  
 *asp-action*="Create">Добавить</*a*>  
 </div>  
 </div>  
 <hr *class*="bg-black"/>  
 <div *class*="row row-cols-3 mt-5 g-3">  
 @*foreach* (*var* exercise *in* ViewBag.Exercises)  
 {  
 <div *class*="col">  
 <div *class*="card" *style*="height: 440px;">  
 <div *class*="card-body d-flex flex-column">  
 <h4 *class*="card-title" *style*="overflow-y: auto; max-height: 3.5em;">@exercise.Name</h4>  
 <p *class*="card-text" *style*="overflow-y: auto; max-height: 5.5em;">@exercise.ShortDescription</p>  
 <div *class*="mt-auto">  
 <ul *class*="list-group list-group-flush">  
 <li *class*="list-group-item">  
 <strong>Сложность:</strong> @exercise.Difficulty?.Name  
 </li>  
 <li *class*="list-group-item">  
 <strong>Очки:</strong> @exercise.Score  
 </li>  
 <li *class*="list-group-item">  
 <strong>Язык:</strong> @exercise.Language.Name  
 </li>  
 <li *class*="list-group-item">  
 <strong>Автор:</strong> <*partial name*="\_AuthorLink" *model*="@exercise.Author" />  
 </li>  
 </ul>  
 <div *class*="d-flex justify-content-between align-items-center mt-3">  
 <*form method*="post"  
 *asp-controller*="Exercises"  
 *asp-action*="Delete"  
 *asp-route-id*="@exercise.Id"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить этот упражнение?');">  
 <div *class*="btn-group rounded-pill">  
 @*if* (exercise.IsPublished)  
 {  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="Exercises"  
 *asp-action*="View"  
 *asp-route-id*="@exercise.Id">Посмотреть</*a*>  
 }  
 @*if* (User.GetId() == exercise.AuthorId || User.IsInRole("Администратор"))  
 {  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="Exercises"  
 *asp-action*="Update"  
 *asp-route-id*="@exercise.Id">Редактировать</*a*>  
 <button *type*="submit" *class*="btn btn-sm btn-danger">Удалить</button>  
 }  
 </div>  
 </*form*>  
   
 @*if* (User.GetId() != exercise.AuthorId)  
 {  
 @*if* (exercise.IsLikedByCurrentUser)  
 {  
 <*form method*="post" *asp-controller*="Exercises" *asp-action*="Unlike" *asp-route-id*="@exercise.Id" *asp-route-returnUrl*="@Context.Request.Path@Context.Request.QueryString">  
 <button *type*="submit" *class*="btn btn-link p-0">  
 <i *class*="bi bi-heart-fill text-danger" *style*="font-size: 1.5rem;"></i>  
 </button>  
 </*form*>  
 }  
 *else* {  
 <*form method*="post" *asp-controller*="Exercises" *asp-action*="Like" *asp-route-id*="@exercise.Id" *asp-route-returnUrl*="@Context.Request.Path@Context.Request.QueryString">  
 <button *type*="submit" *class*="btn btn-link p-0">  
 <i *class*="bi bi-heart" *style*="font-size: 1.5rem;"></i>  
 </button>  
 </*form*>  
 }  
 }  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 }  
 </div>  
</div>

**Exercises/Update.cshtml**

@*using* Coursework.Models.Entities  
@*model* Coursework.Models.DTOs.ExerciseDto  
  
@{  
 ViewData["Title"] = "Редактировать задание";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Редактировать задание</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-8">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="Exercises" *asp-action*="Update" *id*="exerciseForm">  
 <!-- Название -->  
 <div *class*="mb-3">  
 <label *for*="Name" *class*="form-label">Название задание</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="Name" *asp-for*="Name" *placeholder*="Введите название задания">  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Уровень сложности -->  
 <div *class*="mb-3">  
 <label *for*="DifficultyId" *class*="form-label">Уровень сложности</label>  
 <*select class*="form-select rounded-pill" *id*="DifficultyId" *asp-for*="DifficultyId">  
 <*option value*="">-- Выберите уровень сложности --</*option*>  
 @*foreach* (*var* difficulty *in* (ViewBag.DifficultyLevels *as IEnumerable*<DifficultyLevel>)!)  
 {  
 <*option value*="@difficulty.Id">@difficulty.Name</*option*>  
 }  
 </*select*>  
 <*span asp-validation-for*="DifficultyId" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Количество баллов -->  
 <div *class*="mb-3">  
 <label *for*="Score" *class*="form-label">Количество очков</label>  
 <*input type*="number" *class*="form-control rounded-pill" *id*="Score" *asp-for*="Score" *placeholder*="Введите количество очков">  
 <*span asp-validation-for*="Score" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Язык программирования -->  
 <div *class*="mb-3">  
 <label *for*="LanguageId" *class*="form-label">Язык программирования</label>  
 <*select class*="form-select rounded-pill" *id*="LanguageId" *asp-for*="LanguageId">  
 <*option value*="">-- Выберите язык программирования --</*option*>  
 @*foreach* (*var* language *in* ViewBag.Languages)  
 {  
 <*option value*="@language.Id" *title*="@language.Description">@language.Name</*option*>  
 }  
 </*select*>  
 <*span asp-validation-for*="LanguageId" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Краткое описание -->  
 <div *class*="mb-3">  
 <label *for*="ShortDescription" *class*="form-label">Короткое описание</label>  
 <*textarea class*="form-control rounded" *id*="ShortDescription" *asp-for*="ShortDescription" *rows*="2" *placeholder*="Введите краткое описание"></*textarea*>  
 <*span asp-validation-for*="ShortDescription" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Полное описание -->  
 <div *class*="mb-3">  
 <label *for*="FullDescription" *class*="form-label">Задание</label>  
 <*textarea class*="form-control rounded" *id*="FullDescription" *asp-for*="FullDescription" *rows*="4" *placeholder*="Введите текст задания"></*textarea*>  
 <*span asp-validation-for*="FullDescription" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Ключ S3 для материалов -->  
 <div *class*="mb-3">  
 <label *for*="S3KeySource" *class*="form-label">Ключ S3 для материалов упражнения</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3KeySource" *asp-for*="S3KeySource" *placeholder*="Введите ключ S3 для материалов">  
 <*span asp-validation-for*="S3KeySource" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Ключ S3 для тестов -->  
 <div *class*="mb-3">  
 <label *for*="S3KeyTests" *class*="form-label">Ключ S3 для тестов</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3KeyTests" *asp-for*="S3KeyTests" *placeholder*="Введите ключ S3 для тестов">  
 <*span asp-validation-for*="S3KeyTests" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Ключ S3 для решения автора -->  
 <div *class*="mb-3">  
 <label *for*="S3KeyTests" *class*="form-label">Ключ S3 для решения</label>  
 <*input type*="hidden" *asp-for*="AuthorSolutionId">  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3KeyAuthorSolution" *asp-for*="S3KeyAuthorSolution" *placeholder*="Введите ключ S3 для тестов">  
 <*span asp-validation-for*="S3KeyAuthorSolution" *class*="text-danger"></*span*>  
 </div>  
  
 <!-- Выбор фреймворков -->  
 <div *class*="mb-3">  
 <label *for*="FrameworkId" *class*="form-label">Фреймворки</label>  
 <div *class*="input-group">  
 <select *class*="form-select rounded-pill" *id*="FrameworkId" *name*="FrameworkId">  
 <*option value*="">-- Выберите фреймворк --</*option*>  
 @*foreach* (*var* framework *in* ViewBag.Frameworks)  
 {  
 <*option value*="@framework.Id" *title*="@framework.Description">@framework.Name</*option*>  
 }  
 </select>  
 <button *type*="button" *class*="btn btn-primary rounded-pill ms-2" *onclick*="*addFrameworkButton*()">Добавить</button>  
 </div>  
 </div>  
  
 <!-- Список выбранных фреймворков -->  
 <div *id*="selectedFrameworks" *class*="mt-2">  
 <!-- Фреймворки будут добавляться сюда -->  
 </div>  
  
 <!-- Добавление подсказок -->  
 <div *class*="mb-3">  
 <div *class*="card p-3">  
 <h5 *class*="card-title">Добавить подсказку</h5>  
 <div *class*="mb-3">  
 <label *for*="HintCost" *class*="form-label">Очки</label>  
 <input *type*="number" *class*="form-control rounded-pill" *id*="HintCost" *placeholder*="Введите стоимость подсказки">  
 </div>  
 <div *class*="mb-3">  
 <label *for*="HintText" *class*="form-label">Текст подсказки</label>  
 <textarea *class*="form-control rounded" *id*="HintText" *rows*="2" *placeholder*="Введите текст подсказки"></textarea>  
 </div>  
 <button *type*="button" *class*="btn btn-primary rounded-pill" *onclick*="*addHintButton*()">Добавить подсказку</button>  
 </div>  
 </div>  
  
 <!-- Список добавленных подсказок -->  
 <div *id*="hints" *class*="mt-2">  
 <!-- Подсказки будут добавляться сюда -->  
 </div>  
  
 <!-- Кнопки сохранения -->  
 <div *class*="d-flex gap-2 mt-4">  
 <button *type*="submit" *class*="btn btn-secondary rounded-pill flex-grow-1" *onclick*="*setIsPublished*(*false*)">Сохранить</button>  
 <button *type*="submit" *class*="btn btn-success rounded-pill flex-grow-1" *onclick*="*setIsPublished*(*true*)">Опубликовать</button>  
 </div>  
  
 <!-- Скрытое поле для IsPublished -->  
 <input *type*="hidden" *id*="IsPublished" *name*="IsPublished" *value*="false"/>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial"/>  
  
 <script>  
 *let selectedFrameworks* = [];  
 *let hints* = [];  
  
 *function addFrameworkButton*() {  
 *const* select = *document*.getElementById('FrameworkId');  
 *const* selectedOption = select.options[select.selectedIndex];  
 *const* framework = {  
 id: selectedOption.value,  
 name: selectedOption.text  
 };  
  
 *if* (!framework.id) {  
 *alert*('Выберите фреймворк!');  
 *return*;  
 }  
  
 *if* (*selectedFrameworks*.some(f => f.id === framework.id)) {  
 *alert*('Этот фреймворк уже добавлен!');  
 *return*;  
 }  
  
 *selectedFrameworks*.push(framework);  
 *renderSelectedFrameworks*();  
 }  
  
 *function renderSelectedFrameworks*() {  
 *const* container = *document*.getElementById('selectedFrameworks');  
 container.innerHTML = '';  
  
 *selectedFrameworks*.forEach((framework, index) => {  
 *const* div = *document*.createElement('div');  
 div.className = 'd-flex align-items-center mb-2 rounded-pill';  
  
 *const* span = *document*.createElement('span');  
 span.className = 'me-2';  
 span.textContent = framework.name;  
  
 *const* button = *document*.createElement('button');  
 button.type = 'button';  
 button.className = 'btn btn-sm btn-danger rounded-pill';  
 button.textContent = 'Убрать';  
 button.onclick = () => *removeFramework*(index);  
  
 div.appendChild(span);  
 div.appendChild(button);  
  
 container.appendChild(div);  
 });  
 }  
  
 *function removeFramework*(index) {  
 *selectedFrameworks*.splice(index, 1);  
 *renderSelectedFrameworks*();  
 }  
  
 *function addHintButton*() {  
 *const* cost = *document*.getElementById('HintCost').value;  
 *const* text = *document*.getElementById('HintText').value;  
  
 *if* (!cost || !text) {  
 *alert*('Пожалуйста, заполните все поля подсказки!');  
 *return*;  
 }  
  
 *if* (*hints*.some(h => h.text === text)) {  
 *alert*('Такая подсказка уже добавлена!');  
 *return*;  
 }  
  
 *const* hint = {  
 cost: cost,  
 text: text  
 };  
  
 *hints*.push(hint);  
 *renderHints*();  
  
 *document*.getElementById('HintCost').value = '';  
 *document*.getElementById('HintText').value = '';  
 }  
  
 *function renderHints*() {  
 *const* container = *document*.getElementById('hints');  
 container.innerHTML = '';  
  
 *hints*.forEach((hint, index) => {  
 *const* card = *document*.createElement('div');  
 card.className = 'card mb-3';  
  
 *const* cardHeader = *document*.createElement('div');  
 cardHeader.className = 'card-header d-flex justify-content-between align-items-center';  
 cardHeader.innerHTML = `<strong>Подсказка за ${hint.cost} очков</strong>`;  
  
 *const* removeButton = *document*.createElement('button');  
 removeButton.type = 'button';  
 removeButton.className = 'btn btn-sm btn-danger';  
 removeButton.textContent = 'Убрать';  
 removeButton.onclick = () => *removeHint*(index);  
 cardHeader.appendChild(removeButton);  
  
 *const* cardBody = *document*.createElement('div');  
 cardBody.className = 'card-body';  
  
 *const* hintText = *document*.createElement('p');  
 hintText.className = 'card-text';  
 hintText.textContent = hint.text;  
  
 cardBody.appendChild(hintText);  
 card.appendChild(cardHeader);  
 card.appendChild(cardBody);  
  
 container.appendChild(card);  
 });  
 }  
  
 *function removeHint*(index) {  
 *hints*.splice(index, 1);  
 *renderHints*();  
 }  
  
 *function setIsPublished*(value) {  
 *document*.getElementById('IsPublished').value = value;  
 }  
  
 // Создание скрытых инпутов перед отправкой формы  
 *document*.getElementById('exerciseForm').addEventListener('submit', *function* () {  
 *const* form = *this*;  
  
 *const* oldFrameworkInputs = form.querySelectorAll('input[name^="Frameworks"]');  
 *const* oldHintInputs = form.querySelectorAll('input[name^="Hints"]');  
 [...oldFrameworkInputs, ...oldHintInputs].forEach(input => input.remove());  
  
 *selectedFrameworks*.forEach((framework, index) => {  
 *const* idInput = *document*.createElement('input');  
 idInput.type = 'hidden';  
 idInput.name = `Frameworks[${index}].Id`;  
 idInput.value = framework.id;  
  
 *const* nameInput = *document*.createElement('input');  
 nameInput.type = 'hidden';  
 nameInput.name = `Frameworks[${index}].Name`;  
 nameInput.value = framework.name;  
  
 form.appendChild(idInput);  
 form.appendChild(nameInput);  
 });  
  
 *hints*.forEach((hint, index) => {  
 *const* scoreInput = *document*.createElement('input');  
 scoreInput.type = 'hidden';  
 scoreInput.name = `Hints[${index}].Cost`;  
 scoreInput.value = hint.cost;  
  
 *const* textInput = *document*.createElement('input');  
 textInput.type = 'hidden';  
 textInput.name = `Hints[${index}].Text`;  
 textInput.value = hint.text;  
  
 form.appendChild(scoreInput);  
 form.appendChild(textInput);  
 });  
 });  
  
 *document*.addEventListener('DOMContentLoaded', *function* () {  
 @*if* (Model *is not null*)  
 {  
 @*foreach* (var framework *in* Model.Frameworks)  
 {  
 <text>*selectedFrameworks*.push({  
 id: '@framework.Id',  
 name: '@Html.Raw(framework.Name)'  
 });</text>  
 }  
  
 @*foreach* (var hint *in* Model.Hints)  
 {  
 <text>*hints*.push({  
 cost: '@hint.Cost',  
 text: '@Html.Raw(hint.Text)'  
 });</text>  
 }  
 }  
 *renderSelectedFrameworks*();  
 *renderHints*();  
 });  
 </script>  
}

**Exercises/View.cshtml**

@*using* Coursework.Extensions  
@*using* Microsoft.AspNetCore.Mvc.TagHelpers  
@*model* Coursework.Models.DTOs.ExerciseDto  
  
@{  
 ViewData["Title"] = $"Задание {Model.Name}";  
}  
  
<div *class*="container mt-5">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-8">  
 <div *class*="card">  
 <div *class*="card-header">  
 <h1 *class*="card-title text-center">@Model.Name</h1>  
 </div>  
 <div *class*="card-body">  
 <!-- Основные параметры -->  
 <ul *class*="list-group list-group-flush">  
 <li *class*="list-group-item">  
 <strong>Сложность:</strong> @Model.Difficulty?.Name  
 </li>  
 <li *class*="list-group-item">  
 <strong>Очки:</strong> @Model.Score  
 </li>  
 <li *class*="list-group-item">  
 <strong>Язык программирования:</strong> @Model.Language?.Name  
 </li>  
 <li *class*="list-group-item">  
 <strong>Фреймворки:</strong> @string.*Join*(", ", Model.Frameworks.Select(f => f.Name))  
 </li>  
 <li *class*="list-group-item">  
 <strong>Автор:</strong> <*partial name*="\_AuthorLink" *model*="@Model.Author" />  
 </li>  
 <li *class*="list-group-item d-flex justify-content-end align-items-center">  
 <span *class*="me-2">@Model.LikedByUsers.Count</span>  
 @*if* (User.GetId() == Model.AuthorId)  
 {  
 <button *type*="button" *class*="btn btn-link p-0 text-secondary" *disabled*>  
 <i *class*="bi bi-heart-fill" *style*="font-size: 1.5rem;"></i>  
 </button>  
 }  
 *else* {  
 @*if* (Model.IsLikedByCurrentUser)  
 {  
 <*form method*="post" *asp-controller*="Exercises" *asp-action*="Unlike" *asp-route-id*="@Model.Id" *asp-route-returnUrl*="@Context.Request.Path">  
 <button *type*="submit" *class*="btn btn-link p-0">  
 <i *class*="bi bi-heart-fill text-danger" *style*="font-size: 1.5rem;"></i>  
 </button>  
 </*form*>  
 }  
 *else* {  
 <*form method*="post" *asp-controller*="Exercises" *asp-action*="Like" *asp-route-id*="@Model.Id" *asp-route-returnUrl*="@Context.Request.Path">  
 <button *type*="submit" *class*="btn btn-link p-0">  
 <i *class*="bi bi-heart" *style*="font-size: 1.5rem;"></i>  
 </button>  
 </*form*>  
 }  
 }  
 </li>  
 </ul>  
  
 <!-- Короткое описание -->  
 <div *class*="mt-4">  
 <h5 *class*="text-muted">Описание:</h5>  
 <p>@Model.ShortDescription</p>  
 </div>  
  
 <!-- Полное описание -->  
 <div *class*="mt-4">  
 <h5 *class*="text-muted">Задание:</h5>  
 <p *style*="white-space: pre-wrap;">@Model.FullDescription</p>  
 </div>  
   
 <!-- Подсказки -->  
 <div *class*="mt-4">  
 <h5 *class*="text-muted">Подсказки:</h5>  
 <div *class*="accordion" *id*="hintsAccordion">  
 @*foreach* (*var* hint *in* Model.Hints)  
 {  
 *var* isHintOpened = hint.OpenedByUsers.Any(u => u.Id == User.GetId()) || User.GetId() == Model.AuthorId;  
 *var* buttonClass = isHintOpened ? "" : "disabled";  
 *var* headerClass = isHintOpened ? "" : "hide-accordion-arrow";  
   
 <div *class*="accordion-item">  
 <h2 *class*="accordion-header d-flex @headerClass" *id*="heading-@hint.Id">  
 <button *class*="accordion-button collapsed @buttonClass"  
 *type*="button"  
 *data-bs-toggle*="@(isHintOpened ? "collapse" : "")"  
 *data-bs-target*="#collapse-@hint.Id"  
 *aria-expanded*="false"  
 *aria-controls*="collapse-@hint.Id">  
 Подсказка за @hint.Cost очков  
 @*if* (!isHintOpened)  
 {  
 <*form method*="post" *asp-controller*="Exercises" *asp-action*="OpenHint"   
 *asp-route-id*="@hint.Id" *asp-route-exerciseId*="@Model.Id">  
 <button *type*="submit" *style*="width: 180px;" *class*="btn btn-sm btn-primary rounded-pill align-self-center me-3">  
 Открыть подсказку  
 </button>  
 </*form*>  
 }  
 </button>  
 </h2>  
 <div *id*="collapse-@hint.Id"  
 *class*="accordion-collapse collapse"  
 *aria-labelledby*="heading-@hint.Id">  
 <div *class*="accordion-body">@*if* (isHintOpened){@hint.Text}  
 </div>  
 </div>  
 </div>  
 }  
 </div>  
 </div>  
   
 @*if* (TempData["ErrorMessage"] != *null*)  
 {  
 <div *class*="alert alert-danger alert-dismissible fade show" *role*="alert">  
 @TempData["ErrorMessage"]  
 <button *type*="button" *class*="btn-close" *data-bs-dismiss*="alert" *aria-label*="Close"></button>  
 </div>  
 }  
 </div>  
 </div>  
 <div *class*="card mt-1">  
 <div *class*="card-body d-flex justify-content-between flex-row">  
 <h4>Решение автора</h4>  
 <*a class*="btn btn-primary btn rounded-pill"  
 *asp-controller*="Solution"  
 *asp-action*="View"  
 *asp-route-id*="@Model.AuthorSolutionId">Посмотреть</*a*>  
 </div>  
 </div>  
 <div *class*="card mt-1">  
 <div *class*="card-header">  
 <h4 *class*="card-title text-center">Решения</h4>  
 </div>  
 <div *class*="card-body">  
 @*if* (Model.Solutions.Any())  
 {  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Автор</th>  
 <th *scope*="col" *class*="text-end">  
 @*if* (User.GetId() == Model.AuthorId)  
 {  
 <span *style*="white-space: nowrap;">Вы автор, вы не можете загрузить решение на своё же задание.</span>  
 }  
 *else* {  
 <*a class*="btn btn-primary btn-sm rounded-pill"  
 *asp-controller*="Solution"  
 *asp-action*="Create"  
 *asp-route-id*="@Model.Id">  
 Загрузить решение  
 </*a*>  
 }  
 </th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* solution *in* Model.Solutions)  
 {  
 <tr>  
 <td><*partial name*="\_AuthorLink" *model*="@solution.Author" /></td>  
 <td *class*="text-end">  
 <*form method*="post"  
 *asp-controller*="Solution"  
 *asp-action*="Delete"  
 *asp-route-id*="@solution.Id"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить этот упражнение?');">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="Solution"  
 *asp-action*="View"  
 *asp-route-id*="@solution.Id">Посмотреть</*a*>  
 @*if* (User.GetId() == solution.AuthorId || User.IsInRole("Администратор"))  
 {  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="Solution"  
 *asp-action*="Update"  
 *asp-route-id*="@solution.Id">Редактировать</*a*>  
 <button *type*="submit" *class*="btn btn-sm btn-danger">Удалить</button>  
 }  
 </div>  
 </*form*>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
 }  
 *else* {  
 <p *class*="text-center">Для этого задания еще нет решений</p>  
 @*if* (User.GetId() != Model.AuthorId)  
 {  
 <div *class*="text-center mt-3">  
 <*a class*="btn btn-primary rounded-pill"  
 *asp-controller*="Solution"  
 *asp-action*="Create"  
 *asp-route-id*="@Model.Id">  
 Загрузить решение  
 </*a*>  
 </div>  
 }  
 }  
 </div>  
 </div>  
 </div>  
 </div>  
</div>

**Auth/Login.cshtml**

@*model* Coursework.Models.DTOs.LoginDto  
  
@{  
 ViewData["Title"] = "Вход";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Вход в систему</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form asp-action*="Login" *method*="post">  
 <div *class*="mb-3">  
 <*label asp-for*="Email" *class*="form-label">Email</*label*>  
 <*input asp-for*="Email" *class*="form-control rounded-pill" *placeholder*="Введите email" />  
 <*span asp-validation-for*="Email" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <*label asp-for*="Password" *class*="form-label">Пароль</*label*>  
 <*input asp-for*="Password" *class*="form-control rounded-pill" *placeholder*="Введите пароль" />  
 <*span asp-validation-for*="Password" *class*="text-danger"></*span*>  
 </div>  
  
 <*div asp-validation-summary*="ModelOnly" *class*="text-danger"></*div*>  
  
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Войти</button>  
 </div>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**Auth/Register.cshtml**

@*model* Coursework.Models.DTOs.RegisterDto  
  
@{  
 ViewData["Title"] = "Регистрация";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Регистрация</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form asp-action*="Register" *method*="post">  
 <div *class*="mb-3">  
 <*label asp-for*="Name" *class*="form-label">Имя</*label*>  
 <*input asp-for*="Name" *class*="form-control rounded-pill" *placeholder*="Введите имя" />  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <*label asp-for*="Email" *class*="form-label">Email</*label*>  
 <*input asp-for*="Email" *class*="form-control rounded-pill" *placeholder*="Введите email" />  
 <*span asp-validation-for*="Email" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <*label asp-for*="Password" *class*="form-label">Пароль</*label*>  
 <*input asp-for*="Password" *class*="form-control rounded-pill" *placeholder*="Введите пароль" />  
 <*span asp-validation-for*="Password" *class*="text-danger"></*span*>  
 </div>  
  
 <div *class*="mb-3">  
 <*label asp-for*="ConfirmPassword" *class*="form-label">Подтверждение пароля</*label*>  
 <*input asp-for*="ConfirmPassword" *class*="form-control rounded-pill" *placeholder*="Повторите пароль" />  
 <*span asp-validation-for*="ConfirmPassword" *class*="text-danger"></*span*>  
 </div>  
  
 <*div asp-validation-summary*="ModelOnly" *class*="text-danger"></*div*>  
  
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Зарегистрироваться</button>  
 </div>  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**Solutions/Create.cshtml**

@*model* Coursework.Models.Entities.Solution  
  
@{  
 ViewData["Title"] = $"Решение задания \"{ViewBag.Exercise.Name}\"";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4" *style*="word-wrap: break-word; overflow-wrap: break-word;">Добавить решение для задания "@ViewBag.Exercise.Name"</h1>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="Solution" *asp-action*="Create">  
 <div *class*="mb-3">  
 <label *for*="S3Key" *class*="form-label">Ключ S3 решения</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3Key" *asp-for*="S3Key" *placeholder*="Введите ключ S3 решения">  
 <*span asp-validation-for*="S3Key" *class*="text-danger"></*span*>  
 </div>  
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Добавить</button>  
 </div>  
 <*input type*="hidden" *asp-for*="ExerciseId" *value*="@ViewBag.Exercise.Id">  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**Solutions /View.cshtml**

@*using* Coursework.Extensions  
@*using* Microsoft.AspNetCore.Mvc.TagHelpers  
@*model* Coursework.Models.Entities.Solution  
  
@{  
 ViewData["Title"] = $"Решение задания \"{Model.Exercise?.Name}\"";  
 *var* currentUserId = User.GetId();  
 *var* hasLiked = Model.LikedByUsers.Any(u => u.Id == currentUserId);  
}  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-header">  
 <h3 *class*="card-title text-center">Решение задания "@Model.Exercise?.Name"</h3>  
 </div>  
 <div *class*="card-body">  
 <ul *class*="list-group list-group-flush">  
 <li *class*="list-group-item">  
 <strong>S3 Ключ:</strong> @Model.S3Key  
 </li>  
 <li *class*="list-group-item">  
 <strong>Автор:</strong> <*partial name*="\_AuthorLink" *model*="@Model.Author"/>  
 </li>  
 </ul>  
 </div>  
 <div *class*="card-footer d-flex justify-content-between align-items-center">  
 <*a style*="width: 200px"  
 *class*="btn btn-sm btn-primary rounded-pill align-self-center"  
 *asp-controller*="Exercises"  
 *asp-action*="View"  
 *asp-route-id*="@Model.ExerciseId">  
 Перейти к заданию  
 </*a*>  
 <div *class*="d-flex justify-content-end align-items-center">  
 <span *class*="me-2">@Model.LikedByUsers.Count</span>  
 @*if* (User.GetId() == Model.AuthorId)  
 {  
 <button *type*="button" *class*="btn btn-link p-0 text-secondary" *disabled*>  
 <i *class*="bi bi-heart-fill" *style*="font-size: 1.5rem;"></i>  
 </button>  
 }  
 *else* {  
 @*if* (hasLiked)  
 {  
 <*form method*="post" *asp-controller*="Solution" *asp-action*="Unlike" *asp-route-id*="@Model.Id">  
 <button *type*="submit" *class*="btn btn-link p-0">  
 <i *class*="bi bi-heart-fill text-danger" *style*="font-size: 1.5rem;"></i>  
 </button>  
 </*form*>  
 }  
 *else* {  
 <*form method*="post" *asp-controller*="Solution" *asp-action*="Like" *asp-route-id*="@Model.Id">  
 <button *type*="submit" *class*="btn btn-link p-0">  
 <i *class*="bi bi-heart" *style*="font-size: 1.5rem;"></i>  
 </button>  
 </*form*>  
 }  
 }  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>

**Solutions /Update.cshtml**

@*model* Coursework.Models.Entities.Solution  
  
@{  
 ViewData["Title"] = $"Решение задания \"{Model.Exercise?.Name}\"";  
}  
  
<div *class*="container">  
 <div *class*="text-center">  
 <h1 *class*="display-4" *style*="word-wrap: break-word; overflow-wrap: break-word;">Редактировать решение для задания "@Model.Exercise?.Name"</h1>  
 </div>  
</div>  
  
<div *class*="container">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-6">  
 <div *class*="card mt-5">  
 <div *class*="card-body">  
 <*form method*="post" *asp-controller*="Solution" *asp-action*="Update">  
 <div *class*="mb-3">  
 <label *for*="S3Key" *class*="form-label">Ключ S3 решения</label>  
 <*input type*="text" *class*="form-control rounded-pill" *id*="S3Key" *asp-for*="S3Key" *placeholder*="Введите ключ S3 решения">  
 <*span asp-validation-for*="S3Key" *class*="text-danger"></*span*>  
 </div>  
 <div *class*="d-grid">  
 <button *type*="submit" *class*="btn btn-primary rounded-pill">Обновить</button>  
 </div>  
 <*input type*="hidden" *asp-for*="ExerciseId" *value*="@Model.Exercise?.Id">  
 </*form*>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <*partial name*="\_ValidationScriptsPartial" />  
}

**User /Index.cshtml**

@*using* Coursework.Extensions  
@*using* Microsoft.AspNetCore.Mvc.TagHelpers  
@*model* Coursework.Models.Entities.User  
  
@{  
 ViewData["Title"] = "Список пользователей";  
}  
  
<div *class*="text-center">  
 <h1 *class*="display-4">Пользователи</h1>  
</div>  
  
<div *class*="container">  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Id</th>  
 <th *scope*="col">Имя</th>  
 <th *scope*="col">Роль</th>  
 <th *scope*="col">Очки</th>  
 <th *scope*="col" *class*="text-end"></th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* user *in* ViewBag.Users)  
 {  
 <tr>  
 <th *scope*="row">@user.Id</th>  
 <td>@user.Name</td>  
 <td>@user.Role?.Name</td>  
 <td>@user.Score</td>  
 <td *class*="text-end">  
 <*form  
 method*="post"  
 *asp-controller*="User"  
 *asp-action*="Delete"  
 *asp-route-id*="@user.Id"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить этого пользователя? Это действие нельзя отменить.');">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="User"  
 *asp-action*="View"  
 *asp-route-id*="@user.Id">Просмотр</*a*>  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="User"  
 *asp-action*="Update"  
 *asp-route-id*="@user.Id">Редактировать</*a*>  
 @*if* (user.Id != User.GetId())  
 {  
 <button *type*="submit" *class*="btn btn-sm btn-danger">Удалить</button>  
 }  
 </div>  
 </*form*>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
</div>

**User /Update.cshtml**

@*using* Coursework.Extensions  
@*model* Coursework.Models.Entities.User  
  
@{  
 ViewData["Title"] = $"Редактирование профиля {Model.Name}";  
   
 *string* levelName;  
 *string* levelColor;  
 *var* currentScore = (*int*)Model.Score;  
 *int* nextLevelThreshold;  
 *var* previousLevelThreshold = 0;  
  
 *switch* (currentScore)  
 {  
 *case* < 5000:  
 levelName = "Уровень 1";  
 levelColor = "#198754";  
 nextLevelThreshold = 5000;  
 *break*;  
 *case* < 10000:  
 levelName = "Уровень 2";  
 levelColor = "#20c997";  
 previousLevelThreshold = 5000;  
 nextLevelThreshold = 10000;  
 *break*;  
 *case* < 20000:  
 levelName = "Уровень 3";  
 levelColor = "#ffc107";  
 previousLevelThreshold = 10000;  
 nextLevelThreshold = 20000;  
 *break*;  
 *case* < 40000:  
 levelName = "Уровень 4";  
 levelColor = "#fd7e14";  
 previousLevelThreshold = 20000;  
 nextLevelThreshold = 40000;  
 *break*;  
 *default*:  
 levelName = "Уровень 5";  
 levelColor = "#dc3545";  
 previousLevelThreshold = 40000;  
 nextLevelThreshold = 40000;  
 *break*;  
 }  
  
 // Расчет процента прогресса для текущего уровня  
 *var* levelRange = nextLevelThreshold - previousLevelThreshold;  
 *var* levelProgress = currentScore - previousLevelThreshold;  
 *var* progressPercentage = levelRange > 0 ? (levelProgress \* 100) / levelRange : 100;  
   
 *var* isAdmin = User.IsInRole("Администратор");  
 *var* isCurrentUser = User.GetId() == Model.Id;  
}  
  
<div *class*="container mt-5">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-8">  
 <*form method*="post" *asp-action*="Update" *asp-route-id*="@Model.Id">  
 <*input type*="hidden" *asp-for*="Id" />  
 <*input type*="hidden" *asp-for*="Email" />  
 <*input type*="hidden" *asp-for*="PasswordHash" />  
 <*input type*="hidden" *asp-for*="Score" *id*="userScore" />  
 <*input type*="hidden" *asp-for*="RoleId" *id*="userRoleId" />  
   
 <!-- Карточка профиля -->  
 <div *class*="card mb-4">  
 <div *class*="card-body text-center">  
 <div *class*="form-group mb-3">  
 <*input type*="text" *class*="form-control form-control-lg text-center" *asp-for*="Name" *required* />  
 <*span asp-validation-for*="Name" *class*="text-danger"></*span*>  
 </div>  
   
 <p *class*="text-muted">@Model.Role?.Name</p>  
   
 @*if* (isAdmin && Model.Id != User.GetId())  
 {  
 <div *class*="mb-3">  
 <button *type*="button" *class*="btn btn-sm btn-outline-secondary toggle-role-btn rounded-pill"   
 *data-current-role*="@Model.RoleId"   
 *data-admin-role-id*="1"   
 *data-user-role-id*="2">  
 @(Model.RoleId == 1 ? "Сделать пользователем" : "Сделать администратором")  
 </button>  
 </div>  
 }  
   
 <div *class*="mt-4">  
 <h3>  
 <span *class*="badge" *style*="background-color: @levelColor;">@levelName</span>  
 <span *class*="ms-2">@Model.Score очков</span>  
 </h3>  
   
 @*if* (levelName != "Уровень 5")  
 {  
 <div *class*="mt-3">  
 <div *class*="progress" *style*="height: 25px;">  
 <div *class*="progress-bar"   
 *role*="progressbar"   
 *style*="width: @progressPercentage%; background-color: @levelColor;"   
 *aria-valuenow*="@progressPercentage"   
 *aria-valuemin*="0"   
 *aria-valuemax*="100">  
 @progressPercentage%  
 </div>  
 </div>  
 <small *class*="text-muted mt-1 d-block">  
 До следующего уровня: @(nextLevelThreshold - currentScore) очков  
 </small>  
 </div>  
 }  
   
 @*if* (isAdmin)  
 {  
 <div *class*="mt-3 d-flex justify-content-center align-items-center">  
 <div *class*="input-group input-group-sm rounded-pill" *style*="max-width: 500px;">  
 <input *type*="number" *class*="form-control" *id*="scoreInput" *value*="100" *min*="1" />  
 <button *type*="button" *class*="btn ntn-sm btn-success score-btn" *data-operation*="add">  
 <i *class*="bi bi-plus-lg"></i> Добавить  
 </button>  
 <button *type*="button" *class*="btn btn-sm btn-danger score-btn" *data-operation*="subtract">  
 <i *class*="bi bi-dash-lg"></i> Вычесть  
 </button>  
 </div>  
 </div>  
 }  
 </div>  
   
 <div *class*="mt-4">  
 <h4>О себе</h4>  
 <div *class*="form-group">  
 <*textarea class*="form-control" *asp-for*="About" *rows*="5"   
 *placeholder*="Расскажите о себе..."></*textarea*>  
 </div>  
 </div>  
   
 <div *class*="d-flex justify-content-end mt-3">  
 <button *type*="submit" *class*="btn btn-secondary rounded-pill">Сохранить</button>  
 </div>  
 </div>  
 </div>  
 </*form*>  
   
 <!-- Упражнения пользователя -->  
 <div *class*="card mb-4">  
 <div *class*="card-header">  
 <h3>Упражнения</h3>  
 </div>  
 <div *class*="card-body">  
 @*if* (Model.Exercises.Any())  
 {  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Название</th>  
 <th *scope*="col">Сложность</th>  
 <th *scope*="col">Язык</th>  
 <th *scope*="col" *class*="text-end">Действия</th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* exercise *in* Model.Exercises)  
 {  
 <tr>  
 <td>@exercise.Name</td>  
 <td>@exercise.Difficulty?.Name</td>  
 <td>@exercise.Language?.Name</td>  
 <td *class*="text-end">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="Exercises"  
 *asp-action*="View"  
 *asp-route-id*="@exercise.Id">Посмотреть</*a*>  
 </div>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
 }  
 *else* {  
 <p *class*="text-center">Пользователь не создал ни одного упражнения</p>  
 }  
 </div>  
 </div>  
   
 <!-- Решения пользователя -->  
 <div *class*="card mb-4">  
 <div *class*="card-header">  
 <h3>Решения</h3>  
 </div>  
 <div *class*="card-body">  
 @*if* (Model.Solutions.Any())  
 {  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Упражнение</th>  
 <th *scope*="col" *class*="text-end">Действия</th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* solution *in* Model.Solutions)  
 {  
 <tr>  
 <td><*partial name*="\_ExerciseLink" *model*="@solution.Exercise" /></td>  
 <td *class*="text-end">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="Solution"  
 *asp-action*="View"  
 *asp-route-id*="@solution.Id">Посмотреть</*a*>  
 </div>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
 }  
 *else* {  
 <p *class*="text-center">Пользователь не создал ни одного решения</p>  
 }  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
  
@*section* Scripts {  
 <script>  
 $(*document*).ready(*function*() {  
 // Обработка изменения роли  
 $('.toggle-role-btn').click(*function*() {  
 *const* btn = $(*this*);  
 *const* currentRoleId = *parseInt*(btn.data('current-role'));  
 *const* adminRoleId = *parseInt*(btn.data('admin-role-id'));  
 *const* userRoleId = *parseInt*(btn.data('user-role-id'));  
   
 // Переключаем роль  
 *const* newRoleId = currentRoleId === adminRoleId ? userRoleId : adminRoleId;  
 $('#userRoleId').val(newRoleId);  
   
 // Обновляем текст кнопки  
 btn.text(newRoleId === adminRoleId ? 'Сделать пользователем' : 'Сделать администратором');  
 btn.data('current-role', newRoleId);  
 });  
   
 // Обработка изменения очков  
 $('.score-btn').click(*function*() {  
 *const* operation = $(*this*).data('operation');  
 *const* scoreInput = *parseInt*($('#scoreInput').val());  
 *const* currentScore = *parseInt*($('#userScore').val());  
   
 *if* (*isNaN*(scoreInput) || scoreInput <= 0) {  
 *alert*('Пожалуйста, введите положительное число очков');  
 *return*;  
 }  
   
 *let* newScore = currentScore;  
 *if* (operation === 'add') {  
 newScore = currentScore + scoreInput;  
 } *else if* (operation === 'subtract') {  
 newScore = *Math*.max(0, currentScore - scoreInput);  
 }  
   
 $('#userScore').val(newScore);  
   
 $('.badge').next().text(newScore + ' очков');  
 });  
 });  
 </script>  
}

**User /View.cshtml**

@*using* Coursework.Extensions  
@*model* Coursework.Models.Entities.User  
  
@{  
 ViewData["Title"] = $"Профиль пользователя {Model.Name}";  
   
 *string* levelName;  
 *string* levelColor;  
 *var* currentScore = (*int*)Model.Score;  
 *int* nextLevelThreshold;  
 *var* previousLevelThreshold = 0;  
  
 *switch* (currentScore)  
 {  
 *case* < 5000:  
 levelName = "Уровень 1";  
 levelColor = "#198754";  
 nextLevelThreshold = 5000;  
 *break*;  
 *case* < 10000:  
 levelName = "Уровень 2";  
 levelColor = "#20c997";  
 previousLevelThreshold = 5000;  
 nextLevelThreshold = 10000;  
 *break*;  
 *case* < 20000:  
 levelName = "Уровень 3";  
 levelColor = "#ffc107";  
 previousLevelThreshold = 10000;  
 nextLevelThreshold = 20000;  
 *break*;  
 *case* < 40000:  
 levelName = "Уровень 4";  
 levelColor = "#fd7e14";  
 previousLevelThreshold = 20000;  
 nextLevelThreshold = 40000;  
 *break*;  
 *default*:  
 levelName = "Уровень 5";  
 levelColor = "#dc3545";  
 previousLevelThreshold = 40000;  
 nextLevelThreshold = 40000;  
 *break*;  
 }  
  
 // Расчет процента прогресса для текущего уровня  
 *var* levelRange = nextLevelThreshold - previousLevelThreshold;  
 *var* levelProgress = currentScore - previousLevelThreshold;  
 *var* progressPercentage = levelRange > 0 ? (levelProgress \* 100) / levelRange : 100;  
}  
  
<div *class*="container mt-5">  
 <div *class*="row justify-content-center">  
 <div *class*="col-md-8">  
 <!-- Карточка профиля -->  
 <div *class*="card mb-4">  
 <div *class*="card-body text-center">  
 <h1 *class*="display-4">@Model.Name</h1>  
 <p *class*="text-muted">@Model.Role?.Name</p>  
   
 <div *class*="mt-4">  
 <h3>  
 <span *class*="badge" *style*="background-color: @levelColor;">@levelName</span>  
 <span *class*="ms-2">@Model.Score очков</span>  
 </h3>  
   
 @*if* (levelName != "Уровень 5")  
 {  
 <div *class*="mt-3">  
 <div *class*="progress" *style*="height: 25px;">  
 <div *class*="progress-bar"   
 *role*="progressbar"   
 *style*="width: @progressPercentage%; background-color: @levelColor;"   
 *aria-valuenow*="@progressPercentage"   
 *aria-valuemin*="0"   
 *aria-valuemax*="100">  
 @progressPercentage%  
 </div>  
 </div>  
 <small *class*="text-muted mt-1 d-block">  
 До следующего уровня: @(nextLevelThreshold - currentScore) очков  
 </small>  
 </div>  
 }  
 </div>  
   
 <div *class*="mt-4">  
 <h4>О себе</h4>  
 <p *class*="lead">  
 @*if* (*string*.*IsNullOrEmpty*(Model.About))  
 {  
 <span *class*="text-muted">Пользователь не указал информацию о себе</span>  
 }  
 *else* {  
 @Model.About  
 }  
 </p>  
 </div>  
  
 @*if* (User.GetId() == Model.Id || User.IsInRole("Администратор"))  
 {  
 <div *class*="d-flex justify-content-end mt-3">  
 <*a asp-controller*="User" *asp-action*="Update" *asp-route-id*="@Model.Id" *class*="btn btn-sm btn-secondary rounded-pill">  
 Редактировать  
 </*a*>  
 </div>  
 }  
 </div>  
 </div>  
   
 <!-- Упражнения пользователя -->  
 <div *class*="card mb-4">  
 <div *class*="card-header">  
 <h3>Упражнения</h3>  
 </div>  
 <div *class*="card-body">  
 @*if* (Model.Exercises.Any())  
 {  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Название</th>  
 <th *scope*="col">Сложность</th>  
 <th *scope*="col">Язык</th>  
 <th *scope*="col" *class*="text-end">Действия</th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* exercise *in* Model.Exercises)  
 {  
 <tr>  
 <td>@exercise.Name</td>  
 <td>@exercise.Difficulty?.Name</td>  
 <td>@exercise.Language?.Name</td>  
 <td *class*="text-end">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="Exercises"  
 *asp-action*="View"  
 *asp-route-id*="@exercise.Id">Посмотреть</*a*>  
 @*if* (User.GetId() == Model.Id || User.IsInRole("Администратор"))  
 {  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="Exercises"  
 *asp-action*="Update"  
 *asp-route-id*="@exercise.Id">Редактировать</*a*>  
 <button *type*="submit" *class*="btn btn-sm btn-danger"  
 *form*="delete-exercise-@exercise.Id">Удалить</button>  
 }  
 </div>  
   
 @*if* (User.GetId() == Model.Id || User.IsInRole("Администратор"))  
 {  
 <*form id*="delete-exercise-@exercise.Id"  
 *asp-controller*="Exercises"  
 *asp-action*="Delete"  
 *asp-route-id*="@exercise.Id"  
 *method*="post"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить это упражнение?');">  
 </*form*>  
 }  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
 }  
 *else* {  
 <p *class*="text-center">Пользователь не создал ни одного упражнения</p>  
 }  
 </div>  
 </div>  
   
 <!-- Решения пользователя -->  
 <div *class*="card mb-4">  
 <div *class*="card-header">  
 <h3>Решения</h3>  
 </div>  
 <div *class*="card-body">  
 @*if* (Model.Solutions.Any())  
 {  
 <table *class*="table">  
 <thead>  
 <tr>  
 <th *scope*="col">Упражнение</th>  
 <th *scope*="col" *class*="text-end">Действия</th>  
 </tr>  
 </thead>  
 <tbody *class*="table-group-divider">  
 @*foreach* (*var* solution *in* Model.Solutions)  
 {  
 <tr>  
 <td><*partial name*="\_ExerciseLink" *model*="@solution.Exercise" /></td>  
 <td *class*="text-end">  
 <div *class*="btn-group rounded-pill">  
 <*a class*="btn btn-sm btn-primary"  
 *asp-controller*="Solution"  
 *asp-action*="View"  
 *asp-route-id*="@solution.Id">Посмотреть</*a*>  
 @*if* (User.GetId() == Model.Id || User.IsInRole("Администратор"))  
 {  
 <*a class*="btn btn-sm btn-secondary"  
 *asp-controller*="Solution"  
 *asp-action*="Update"  
 *asp-route-id*="@solution.Id">Редактировать</*a*>  
 <button *type*="submit" *class*="btn btn-sm btn-danger"  
 *form*="delete-solution-@solution.Id">Удалить</button>  
 }  
 </div>  
 @*if* (User.GetId() == Model.Id || User.IsInRole("Администратор"))  
 {  
 <*form id*="delete-solution-@solution.Id"  
 *asp-controller*="Solution"  
 *asp-action*="Delete"  
 *asp-route-id*="@solution.Id"  
 *method*="post"  
 *onsubmit*="*return confirm*('Вы уверены, что хотите удалить это решение?');">  
 </*form*>  
 }  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
 }  
 *else* {  
 <p *class*="text-center">Пользователь не создал ни одного решения</p>  
 }  
 </div>  
 </div>  
 </div>  
 </div>  
</div>

**ErrorViewModel.cs**

*namespace* Coursework.Models.DTOs;  
  
*public class* ErrorViewModel  
{  
 *public string*? RequestId { get; set; }  
  
 *public bool* ShowRequestId => !*string*.*IsNullOrEmpty*(RequestId);  
}

**ExerciseDto.cs**

*using* System.ComponentModel.DataAnnotations;  
*using* Coursework.Models.Entities;  
*using* Coursework.Models.Interfaces;  
*using* Microsoft.AspNetCore.Mvc.ModelBinding.Validation;  
  
*namespace* Coursework.Models.DTOs;  
  
*public class* ExerciseDto : *IName*{  
 *public long* Id { get; set; }  
   
 [Required(ErrorMessage = "Название упражнения обязательно для заполнения.")]  
 [StringLength(50, ErrorMessage = "Название не должно превышать 50 символов.")]  
 *public string* Name { get; set; }  
  
 [Required(ErrorMessage = "Уровень сложности обязателен.")]  
 *public long* DifficultyId { get; set; }  
   
 [Required(ErrorMessage = "Язык программирования обязателен.")]  
 *public long* LanguageId { get; set; }  
   
   
 [Range(1, 25000, ErrorMessage = "Количество очков должно быть от 0 до 25000.")]  
 [Required(ErrorMessage = "Количество очков обязательно.")]  
 *public long* Score { get; set; }  
  
 [StringLength(150, ErrorMessage = "Краткое описание не должно превышать 150 символов.")]  
 *public string*? ShortDescription { get; set; }  
  
 [Required(ErrorMessage = "Текст упражнения обязателен.")]  
 *public string* FullDescription { get; set; }  
 *public bool* IsPublished { get; set; }  
  
 [StringLength(1024, ErrorMessage = "Ключ S3 для материалов упражнения не должен превышать 1024 символа.")]  
 *public string*? S3KeySource { get; set; }  
  
 [Required(ErrorMessage = "Ключ S3 для тестов обязателен.")]  
 [StringLength(1024, ErrorMessage = "Ключ S3 для тестов не должен превышать 1024 символа.")]  
 *public string*? S3KeyTests { get; set; }  
   
 [ValidateNever]  
 *public long* AuthorSolutionId { get; set; }  
   
 [Required(ErrorMessage = "Ключ S3 для тестов обязателен.")]  
 [StringLength(1024, ErrorMessage = "Ключ S3 для решения автора не должен превышать 1024 символа.")]  
 *public string* S3KeyAuthorSolution { get; set; }  
   
 [ValidateNever]  
 *public* DifficultyLevel? Difficulty { get; set; }  
   
 [ValidateNever]  
 *public* ProgrammingLanguage? Language { get; set; }  
  
 [ValidateNever]   
 *public IList*<Solution> Solutions { get; set; } = *new* List<Solution>();  
   
 [ValidateNever]  
 *public IList*<Framework> Frameworks { get; set; } = *new* List<Framework>();  
  
 [ValidateNever]  
 *public IList*<Hint> Hints { get; set; } = *new* List<Hint>();  
   
 [ValidateNever]  
 *public long* AuthorId { get; set; }  
   
 [ValidateNever]  
 *public* User? Author { get; set; }  
   
 [ValidateNever]  
 *public IList*<User> LikedByUsers { get; set; } = *new* List<User>();  
   
 [ValidateNever]  
 *public bool* IsLikedByCurrentUser { get; set; }  
}

**LoginDto.cs**

*using* System.ComponentModel.DataAnnotations;  
  
*namespace* Coursework.Models.DTOs;  
  
*public class* LoginDto  
{  
 [Required(ErrorMessage = "Email обязателен")]  
 [EmailAddress(ErrorMessage = "Некорректный формат email")]  
 [Display(Name = "Email")]  
 *public string* Email { get; set; } = *string*.*Empty*;  
   
 [Required(ErrorMessage = "Пароль обязателен")]  
 [DataType(DataType.Password)]  
 [Display(Name = "Пароль")]  
 *public string* Password { get; set; } = *string*.*Empty*;  
}

**RegisterDto.cs**

*using* System.ComponentModel.DataAnnotations;  
  
*namespace* Coursework.Models.DTOs;  
  
*public class* RegisterDto  
{  
 [Required(ErrorMessage = "Имя обязательно")]  
 [StringLength(100, ErrorMessage = "Имя должно содержать от {2} до {1} символов", MinimumLength = 2)]  
 [Display(Name = "Имя")]  
 *public string* Name { get; set; } = *string*.*Empty*;  
   
 [Required(ErrorMessage = "Email обязателен")]  
 [EmailAddress(ErrorMessage = "Некорректный формат email")]  
 [Display(Name = "Email")]  
 *public string* Email { get; set; } = *string*.*Empty*;  
   
 [Required(ErrorMessage = "Пароль обязателен")]  
 [StringLength(100, ErrorMessage = "Пароль должен содержать от {2} до {1} символов", MinimumLength = 6)]  
 [DataType(DataType.Password)]  
 [Display(Name = "Пароль")]  
 *public string* Password { get; set; } = *string*.*Empty*;  
   
 [DataType(DataType.Password)]  
 [Display(Name = "Подтверждение пароля")]  
 [Compare("Password", ErrorMessage = "Пароли не совпадают")]  
 *public string* ConfirmPassword { get; set; } = *string*.*Empty*;  
}

**DifficultyLevel.cs**

*using* Coursework.Models.Interfaces;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* DifficultyLevel : Entity, *IName*{  
 *public string* Name { get; set; }  
}

**Entity.cs**

*namespace* Coursework.Models.Entities;  
  
*public class* Entity  
{  
 *public long* Id { get; set; }  
}

**Exercise.cs**

*using* Coursework.Models.Interfaces;  
*namespace* Coursework.Models.Entities;  
  
*public class* Exercise : Entity, *IName*{  
 *public string* Name { get; set; }  
   
 *public long* DifficultyId { get; set; }  
 *public long* LanguageId { get; set; }  
   
 *public long* Score { get; set; }  
   
 *public string*? ShortDescription { get; set; }  
  
 *public string* FullDescription { get; set; }  
 *public bool* IsPublished { get; set; }  
   
 *public string*? S3KeySource { get; set; }  
   
 *public string* S3KeyTests { get; set; }  
   
 *public* DifficultyLevel? Difficulty { get; set; }  
 *public* ProgrammingLanguage? Language { get; set; }  
   
 *public long* AuthorId { get; set; }  
 *public* User? Author { get; set; }  
   
 *public* Solution? AuthorSolution { get; set; }  
 *public IList*<Framework> Frameworks { get; set; } = *new* List<Framework>();  
 *public IList*<Solution> Solutions { get; set; } = *new* List<Solution>();  
   
 *public IList*<Hint> Hints { get; set; } = *new* List<Hint>();  
   
 *public IList*<User> LikedByUsers { get; set; } = *new* List<User>();  
}

**Framework.cs**

*using* System.ComponentModel.DataAnnotations;  
*using* Coursework.Models.Interfaces;  
*using* Microsoft.AspNetCore.Mvc.ModelBinding.Validation;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* Framework : Entity, *IName*{  
 [Required(ErrorMessage = "Название фреймворка обязательно для заполнения.")]  
 [StringLength(50, ErrorMessage = "Название не должно превышать 50 символов.")]  
 *public string* Name { get; set; }  
   
 [Required(ErrorMessage = "Описание фреймворка обязательно для заполнения.")]  
 [StringLength(500, ErrorMessage = "Описание не должно превышать 500 символов.")]  
 *public string* Description { get; set; }  
   
 [Required(ErrorMessage = "Язык программирования обязателен.")]  
 *public long*? LanguageId { get; set; }  
  
 [ValidateNever]  
 *public* ProgrammingLanguage? Language { get; set; }  
}

**Hint.cs**

*using* System.ComponentModel.DataAnnotations;  
*using* Microsoft.AspNetCore.Mvc.ModelBinding.Validation;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* Hint : Entity  
{  
 [Required(ErrorMessage = "ID упражнения обязательно")]  
 *public long* ExerciseId { get; set; }  
   
 [Required(ErrorMessage = "Стоимость подсказки обязательна")]  
 [Range(1, *int*.MaxValue, ErrorMessage = "Стоимость должна быть положительной")]  
 *public int* Cost { get; set; }  
   
 [Required(ErrorMessage = "Текст подсказки обязателен")]  
 [StringLength(1000, ErrorMessage = "Текст подсказки не должен превышать 1000 символов")]  
 *public string* Text { get; set; }  
   
 [ValidateNever]  
 *public* Exercise? Exercise { get; set; }  
   
 [ValidateNever]  
 *public IList*<User> OpenedByUsers { get; set; } = *new* List<User>();  
}

**ProgrammingLanguage.cs**

*using* System.ComponentModel.DataAnnotations;  
*using* Coursework.Models.Interfaces;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* ProgrammingLanguage : Entity, *IName*{  
 [Required(ErrorMessage = "Название языка обязательно для заполнения.")]  
 [StringLength(50, ErrorMessage = "Название не должно превышать 50 символов.")]  
 *public string* Name { get; set; }  
   
 [Required(ErrorMessage = "Описание языка обязательно для заполнения.")]  
 [StringLength(500, ErrorMessage = "Описание не должно превышать 500 символов.")]  
 *public string* Description { get; set; }  
}

**Role.cs**

*using* Coursework.Models.Interfaces;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* Role : Entity, *IName*{  
 *public string* Name { get; set; }  
}

**Solution.cs**

*using* System.ComponentModel.DataAnnotations;  
*using* Microsoft.AspNetCore.Mvc.ModelBinding.Validation;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* Solution : Entity  
{  
 [Required(ErrorMessage = "Ключ S3 обязателен.")]  
 [StringLength(1024, ErrorMessage = "Ключ S3 не должен превышать 1024 символа.")]  
 *public string* S3Key { get; set; }  
   
 [Required(ErrorMessage = "Id упражнения обязателено.")]  
 *public long* ExerciseId { get; set; }  
   
 [Required(ErrorMessage = "Id владельца обязателен.")]  
 *public long* AuthorId { get; set; }  
   
 [ValidateNever]  
 *public* Exercise? Exercise { get; set; }  
   
 [ValidateNever]  
 *public* User? Author { get; set; }  
   
 [ValidateNever]  
 *public IList*<User> LikedByUsers { get; set; } = *new* List<User>();  
}

**User.cs**

*using* System.ComponentModel.DataAnnotations;  
*using* Microsoft.AspNetCore.Mvc.ModelBinding.Validation;  
  
*namespace* Coursework.Models.Entities;  
  
*public class* User : Entity  
{  
 [Required(ErrorMessage = "Имя пользователя обязательно для заполнения")]  
 [StringLength(100, ErrorMessage = "Имя пользователя не должно превышать 100 символов")]  
 *public string* Name { get; set; }  
   
 [Required(ErrorMessage = "Email обязателен для заполнения")]  
 [StringLength(255, ErrorMessage = "Email не должен превышать 255 символов")]  
 [EmailAddress(ErrorMessage = "Некорректный формат email")]  
 *public string* Email { get; set; }  
   
 [Required(ErrorMessage = "Пароль обязателен для заполнения")]  
 [StringLength(255, ErrorMessage = "Хеш пароля не должен превышать 255 символов")]  
 *public string* PasswordHash { get; set; }  
   
 *public string* About { get; set; } = *string*.*Empty*;  
   
 *public long* Score { get; set; } = 0;  
   
 [Required(ErrorMessage = "Роль пользователя обязательна")]  
 *public long* RoleId { get; set; }  
   
 [ValidateNever]  
 *public* Role? Role { get; set; }  
   
 [ValidateNever]  
 *public IList*<Exercise> Exercises { get; set; } = *new* List<Exercise>();  
   
 [ValidateNever]  
 *public IList*<Solution> Solutions { get; set; } = *new* List<Solution>();  
}

**IName.cs**

*namespace* Coursework.Models.Interfaces;  
  
*public interface IName*{  
 *public string* Name { get; set; }  
}

**JwtSettings.cs**

*namespace* Coursework.Models;  
  
*public class* JwtSettings  
{  
 *public string* SecretKey { get; set; } = *string*.*Empty*;  
 *public string* Issuer { get; set; } = *string*.*Empty*;  
 *public string* Audience { get; set; } = *string*.*Empty*;  
 *public int* ExpiryInMinutes { get; set; }  
}

**JwtService.cs**

*using* System.IdentityModel.Tokens.Jwt;  
*using* System.Security.Claims;  
*using* System.Text;  
*using* Coursework.Interfaces.Services;  
*using* Coursework.Models;  
*using* Microsoft.IdentityModel.Tokens;  
  
*namespace* Coursework.Services;  
  
*public class* JwtService(JwtSettings jwtSettings) : *IJwtService*{  
 *public string* GenerateToken(*long* userId, *string* userName, *string* roleName)  
 {  
 *var* securityKey = *new* SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings.SecretKey));  
 *var* credentials = *new* SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);  
   
 *var* claims = *new* List<Claim>  
 {  
 *new*(ClaimTypes.NameIdentifier, userId.ToString()),  
 *new*(ClaimTypes.Name, userName),  
 *new*(ClaimTypes.Role, roleName)  
 };  
   
 *var* token = *new* JwtSecurityToken(  
 issuer: jwtSettings.Issuer,  
 audience: jwtSettings.Audience,  
 claims: claims,  
 expires: DateTime.UtcNow.AddMinutes(jwtSettings.ExpiryInMinutes),  
 signingCredentials: credentials  
 );  
   
 *return new* JwtSecurityTokenHandler().WriteToken(token);  
 }  
   
 *public* ClaimsPrincipal? ValidateToken(*string* token)  
 {  
 *var* tokenHandler = *new* JwtSecurityTokenHandler();  
 *var* key = Encoding.UTF8.GetBytes(jwtSettings.SecretKey);  
   
 *var* validationParameters = *new* TokenValidationParameters  
 {  
 ValidateIssuer = *true*,  
 ValidateAudience = *true*,  
 ValidateLifetime = *true*,  
 ValidateIssuerSigningKey = *true*,  
 ValidIssuer = jwtSettings.Issuer,  
 ValidAudience = jwtSettings.Audience,  
 IssuerSigningKey = *new* SymmetricSecurityKey(key)  
 };  
   
 *try* {  
 *var* principal = tokenHandler.ValidateToken(token, validationParameters, *out* \_);  
 *return* principal;  
 }  
 *catch* {  
 *return null*;  
 }  
 }  
}

**PasswordService.cs**

*using* Coursework.Interfaces.Services;  
  
*namespace* Coursework.Services;  
  
*public class* PasswordService : *IPasswordService*{  
 *public string* HashPassword(*string* password)  
 {  
 *return* BCrypt.Net.BCrypt.*HashPassword*(password);  
 }  
   
 *public bool* VerifyPassword(*string* password, *string* hash)  
 {  
 *return* BCrypt.Net.BCrypt.*Verify*(password, hash);  
 }  
}

**ICrudRepository.cs**

*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface ICrudRepository*<*T*>  
{  
 Task<*long*> AddAsync(*T* entity);  
 Task UpdateAsync(*T* entity);  
 Task DeleteAsync(*long* id);  
 Task<*T*?> GetAsync(*long* id);  
 Task<List<*T*>> GetAllAsync();  
}

**IDifficultyLevelRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface IDifficultyLevelRepository*{  
 Task<List<DifficultyLevel>> GetAllAsync();  
}

**IExerciseRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface IExerciseRepository* : *ICrudRepository*<Exercise>  
{  
 Task<List<Exercise>> GetAllAsync(*string* search);  
 Task AddLikeAsync(*long* exerciseId, *long* userId);  
 Task RemoveLikeAsync(*long* exerciseId, *long* userId);  
 Task OpenHintAsync(*long* hintId, *long* userId);  
}

**IFrameworkRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface IFrameworkRepository* : *ICrudRepository*<Framework>  
{  
 Task<List<Framework>> GetAllWithLanguageAsync();  
}

**IProgrammingLanguageRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface IProgrammingLanguageRepository* : *ICrudRepository*<ProgrammingLanguage>  
{  
   
}

**IRoleRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface IRoleRepository*{  
 Task<Role?> GetAsync(*long* id);  
}

**ISolutionRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface ISolutionRepository* : *ICrudRepository*<Solution>  
{  
 Task AddLikeAsync(*long* solutionId, *long* userId);  
 Task RemoveLikeAsync(*long* solutionId, *long* userId);  
}

**IUserRepository.cs**

*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Interfaces.Database.Repositories;  
  
*public interface IUserRepository* : *ICrudRepository*<User>  
{  
 Task<User?> GetByEmailAsync(*string* email);  
}

**IUnitOfWork.cs**

*using* Coursework.Interfaces.Database.Repositories;  
  
*namespace* Coursework.Interfaces.Database;  
  
*public interface IUnitOfWork* : *IDisposable*, *IAsyncDisposable*{  
 *IProgrammingLanguageRepository* Languages { get; }  
 *IFrameworkRepository* Frameworks { get; }  
 *IDifficultyLevelRepository* DifficultyLevels { get; }  
 *IExerciseRepository* Exercises { get; }  
 *ISolutionRepository* Solutions { get; }  
 *IUserRepository* Users { get; }  
 *IRoleRepository* Roles { get; }  
   
 Task CommitAsync(CancellationToken cancellationToken);  
 Task RollBackAsync(CancellationToken cancellationToken);  
}

**IUnitOfWorkFactory.cs**

*namespace* Coursework.Interfaces.Database;  
  
*public interface IUnitOfWorkFactory*{  
 Task<*IUnitOfWork*> CreateAsync(CancellationToken token);  
}

**IJwtService.cs**

*using* System.Security.Claims;  
  
*namespace* Coursework.Interfaces.Services;  
  
*public interface IJwtService*{  
 *string* GenerateToken(*long* userId, *string* userName, *string* roleName);  
 ClaimsPrincipal? ValidateToken(*string* token);  
}

**IPasswordService.cs**

*namespace* Coursework.Interfaces.Services;  
  
*public interface IPasswordService*{  
 *string* HashPassword(*string* password);  
 *bool* VerifyPassword(*string* password, *string* hash);  
}

**ExerciseExtensions.cs**

*using* Coursework.Models.DTOs;  
*using* Coursework.Models.Entities;  
  
*namespace* Coursework.Extensions;  
  
*public static class* ExerciseExtensions  
{  
 *public static* Exercise Map(*this* ExerciseDto exerciseDto)  
 {  
 *return new* Exercise  
 {  
 Id = exerciseDto.Id,  
 Name = exerciseDto.Name,  
 DifficultyId = exerciseDto.DifficultyId,  
 LanguageId = exerciseDto.LanguageId,  
 Score = exerciseDto.Score,  
 ShortDescription = exerciseDto.ShortDescription,  
 FullDescription = exerciseDto.FullDescription,  
 IsPublished = exerciseDto.IsPublished,  
 S3KeySource = exerciseDto.S3KeySource,  
 S3KeyTests = exerciseDto.S3KeyTests!,  
 Frameworks = exerciseDto.Frameworks,  
 AuthorId = exerciseDto.AuthorId,  
 AuthorSolution = *new* Solution  
 {  
 Id = exerciseDto.AuthorSolutionId,  
 S3Key = exerciseDto.S3KeyAuthorSolution,  
 ExerciseId = exerciseDto.Id,  
 AuthorId = exerciseDto.AuthorId  
 },  
 Hints = exerciseDto.Hints.ToList(),  
 LikedByUsers = exerciseDto.LikedByUsers  
 };  
 }  
   
 *public static* ExerciseDto Map(*this* Exercise exercise)  
 {  
 *return new* ExerciseDto  
 {  
 Id = exercise.Id,  
 Name = exercise.Name,  
 DifficultyId = exercise.DifficultyId,  
 Difficulty = exercise.Difficulty,  
 LanguageId = exercise.LanguageId,  
 Language = exercise.Language,  
 Score = exercise.Score,  
 ShortDescription = exercise.ShortDescription,  
 FullDescription = exercise.FullDescription,  
 IsPublished = exercise.IsPublished,  
 S3KeySource = exercise.S3KeySource,  
 S3KeyTests = exercise.S3KeyTests,  
 Frameworks = exercise.Frameworks,  
 AuthorId = exercise.AuthorId,  
 Author = exercise.Author,  
 AuthorSolutionId = exercise.AuthorSolution?.Id ?? *default*,  
 S3KeyAuthorSolution = exercise.AuthorSolution?.S3Key!,  
 Solutions = exercise.Solutions,  
 Hints = exercise.Hints.ToList(),  
 LikedByUsers = exercise.LikedByUsers  
 };  
 }  
   
 *public static* ExerciseDto MapWithCurrentUser(*this* Exercise exercise, *long* currentUserId)  
 {  
 *var* dto = exercise.Map();  
 dto.IsLikedByCurrentUser = exercise.LikedByUsers.Any(u => u.Id == currentUserId);  
 *return* dto;  
 }  
}

**UserExtensions.cs**

*using* System.Security.Claims;  
  
*namespace* Coursework.Extensions;  
  
*public static class* UserExtensions  
{  
 *public static long* GetId(*this* ClaimsPrincipal user)  
 {  
 *var* userIdClaim = user.FindFirst(ClaimTypes.NameIdentifier);  
  
 *if* (userIdClaim == *null*)  
 {  
 *return default*;  
 }  
   
 *return long*.*TryParse*(userIdClaim.Value, *out var* id) ? id : *default*;  
 }  
}

**DbConnectionFactory.cs**

*using* Npgsql;  
  
*namespace* Coursework.Database.Infrastructure;  
  
*public class* DbConnectionFactory(*IConfiguration* configuration)  
{  
 *public async* Task<NpgsqlConnection> OpenAsync(CancellationToken cancellationToken)  
 {  
 cancellationToken.ThrowIfCancellationRequested();  
 *var* connection = *new* NpgsqlConnection(configuration.GetConnectionString("MainDbConnection"));  
 *await* connection.OpenAsync(cancellationToken);  
 *return* connection;  
 }  
}

**UnitOfWork.cs**

*using* Coursework.Database.Repositories;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Npgsql;  
  
*namespace* Coursework.Database.Infrastructure;  
  
*public class* UnitOfWork(NpgsqlConnection connection, NpgsqlTransaction transaction) : *IUnitOfWork*{  
 *private bool* \_commited;  
  
 *public IProgrammingLanguageRepository* Languages => *new* ProgrammingLanguageRepository(connection, transaction);  
 *public IFrameworkRepository* Frameworks => *new* FrameworkRepository(connection, transaction);  
 *public IDifficultyLevelRepository* DifficultyLevels => *new* DifficultyLevelRepository(connection, transaction);  
 *public IExerciseRepository* Exercises => *new* ExerciseRepository(connection, transaction);  
 *public ISolutionRepository* Solutions => *new* SolutionRepository(connection, transaction);  
 *public IUserRepository* Users => *new* UserRepository(connection, transaction);  
 *public IRoleRepository* Roles => *new* RoleRepository(connection, transaction);  
  
 *public async* Task CommitAsync(CancellationToken token)  
 {  
 *if* (\_commited)  
 {  
 *throw new* InvalidOperationException("Already committed");  
 }  
 \_commited = *true*;  
 *await* transaction.CommitAsync(token);  
 }  
  
 *public async* Task RollBackAsync(CancellationToken token)  
 {  
 *await* transaction.RollbackAsync(token);  
 transaction = *await* connection.BeginTransactionAsync(token);  
 }  
   
 *public async* ValueTask DisposeAsync()  
 {  
 *await* connection.DisposeAsync();  
 *await* transaction.DisposeAsync();  
 }  
  
 *public void* Dispose()  
 {  
 connection.Dispose();  
 transaction.Dispose();  
 }  
}

**UnitOfWorkFactory.cs**

*using* Coursework.Interfaces.Database;  
  
*namespace* Coursework.Database.Infrastructure;  
  
*public class* UnitOfWorkFactory(DbConnectionFactory connectionFactory) : *IUnitOfWorkFactory*{  
 *public async* Task<*IUnitOfWork*> CreateAsync(CancellationToken token)  
 {  
 *var* connection = *await* connectionFactory.OpenAsync(token);  
 *var* transaction = *await* connection.BeginTransactionAsync(token);  
 *return new* UnitOfWork(connection, transaction);  
 }  
}

**M0001Init.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_02\_1949)]   
*public class* M0001Init : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("programing\_languages")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("name").AsString(50).NotNullable()  
 .WithColumn("description").AsString(500).NotNullable();  
 }  
}

**M0002AddFramework.cs**

*using* System.Data;  
*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_03\_1136)]   
*public class* M0002AddFramework : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("frameworks")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("name").AsString(50).NotNullable()  
 .WithColumn("description").AsString(500).NotNullable()  
 .WithColumn("language\_id").AsInt64().NotNullable();  
  
 Create.ForeignKey("FK\_frameworks\_programing\_languages")  
 .FromTable("frameworks").ForeignColumn("language\_id")  
 .ToTable("programing\_languages").PrimaryColumn("id")  
 .OnDelete(Rule.Cascade);  
   
 Create.Index("IX\_frameworks\_language\_id")  
 .OnTable("frameworks")  
 .OnColumn("language\_id")  
 .Ascending();  
 }  
}

**M0003AddExercises.cs**

*using* System.Data;  
*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_03\_1220)]   
*public class* M0003AddExercises : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("difficulty\_levels")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("name").AsString(50).NotNullable();  
   
 Insert.IntoTable("difficulty\_levels").Row(*new* { name = "Очень легко" });  
 Insert.IntoTable("difficulty\_levels").Row(*new* { name = "Легко" });  
 Insert.IntoTable("difficulty\_levels").Row(*new* { name = "Средне" });  
 Insert.IntoTable("difficulty\_levels").Row(*new* { name = "Сложно" });  
 Insert.IntoTable("difficulty\_levels").Row(*new* { name = "Очень сложно" });  
   
 Create.Table("exercises")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("name").AsString(50).NotNullable()  
 .WithColumn("difficulty\_id").AsInt64().NotNullable()  
 .WithColumn("score").AsInt64().NotNullable()  
 .WithColumn("short\_description").AsString(250).Nullable()  
 .WithColumn("full\_description").AsCustom("TEXT").NotNullable()  
 .WithColumn("is\_published").AsBoolean().NotNullable()  
 .WithColumn("s3\_key\_source").AsString(1024).Nullable()  
 .WithColumn("s3\_key\_tests").AsString(1024).NotNullable();  
   
 Create.ForeignKey("FK\_exercises\_difficulty\_levels")  
 .FromTable("exercises").ForeignColumn("difficulty\_id")  
 .ToTable("difficulty\_levels").PrimaryColumn("id");  
   
 Create.Index("IX\_exercises\_difficulty\_id")  
 .OnTable("exercises")  
 .OnColumn("difficulty\_id")  
 .Ascending();  
   
 Create.Table("frameworks\_exercises")  
 .WithColumn("framework\_id").AsInt64().NotNullable()  
 .WithColumn("exercise\_id").AsInt64().NotNullable();  
   
 Create.PrimaryKey("PK\_frameworks\_exercises")  
 .OnTable("frameworks\_exercises")  
 .Columns(["framework\_id", "exercise\_id"]);  
   
 Create.ForeignKey("FK\_frameworks\_exercises\_frameworks")  
 .FromTable("frameworks\_exercises").ForeignColumn("framework\_id")  
 .ToTable("frameworks").PrimaryColumn("id")  
 .OnDelete(Rule.Cascade);  
  
 Create.ForeignKey("FK\_frameworks\_exercises\_exercises")  
 .FromTable("frameworks\_exercises").ForeignColumn("exercise\_id")  
 .ToTable("exercises").PrimaryColumn("id")  
 .OnDelete(Rule.Cascade);  
   
 Create.Index("IX\_frameworks\_exercises\_framework\_id")  
 .OnTable("frameworks\_exercises")  
 .OnColumn("framework\_id")  
 .Ascending();  
   
 Create.Index("IX\_frameworks\_exercises\_exercise\_id")  
 .OnTable("frameworks\_exercises")  
 .OnColumn("exercise\_id")  
 .Ascending();  
 }  
}

**M0004AddSolutions.cs**

*using* System.Data;  
*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_06\_0007)]   
*public class* M0004AddSolutions : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("solutions")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("s3\_key").AsString(1024).NotNullable()  
 .WithColumn("by\_exercise\_author").AsBoolean().NotNullable()  
 .WithColumn("exercise\_id").AsInt64().NotNullable();  
   
 Create.ForeignKey("FK\_solutions\_exercises")  
 .FromTable("solutions").ForeignColumn("exercise\_id")  
 .ToTable("exercises").PrimaryColumn("id")  
 .OnDelete(Rule.Cascade);;  
   
 Create.Index("IX\_solutions\_exercise\_id")  
 .OnTable("solutions")  
 .OnColumn("exercise\_id")  
 .Ascending();  
   
 Alter.Column("short\_description")  
 .OnTable("exercises")  
 .AsAnsiString(150)  
 .NotNullable();  
 }  
}

**M0005AddLanguageToExercise.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_08\_0054)]   
*public class* M0005AddLanguageToExercise : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Alter.Table("exercises")  
 .AddColumn("language\_id").AsInt32().NotNullable();  
   
 Create.Index("IX\_exercises\_language\_id")  
 .OnTable("exercises")  
 .OnColumn("language\_id")  
 .Ascending();  
   
 Create.ForeignKey("FK\_exercises\_programing\_languages")  
 .FromTable("exercises").ForeignColumn("language\_id")  
 .ToTable("programing\_languages").PrimaryColumn("id");  
 }  
}

**M0006AddHints.cs**

*using* System.Data;  
*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_10\_0008)]   
*public class* M0006AddHints : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("hints")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("exercise\_id").AsInt64().NotNullable()  
 .WithColumn("cost").AsInt32().NotNullable()  
 .WithColumn("text").AsString(1000).NotNullable();  
   
 Create.ForeignKey("FK\_hints\_exercises")  
 .FromTable("hints").ForeignColumn("exercise\_id")  
 .ToTable("exercises").PrimaryColumn("id")  
 .OnDelete(Rule.Cascade);  
   
 Create.Index("IX\_hints\_exercise\_id")  
 .OnTable("hints")  
 .OnColumn("exercise\_id")  
 .Ascending();  
 }  
}

**M0007AddUser.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_10\_1833)]   
*public class* M0007AddUser : Migration  
{  
 *public override void* Up()  
 {  
 Create.Table("roles")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("name").AsString(50).NotNullable();  
   
 Insert.IntoTable("roles").Row(*new* { name = "Администратор" });  
 Insert.IntoTable("roles").Row(*new* { name = "Пользователь" });  
   
 Create.Table("users")  
 .WithColumn("id").AsInt64().PrimaryKey().Identity()  
 .WithColumn("name").AsString(100).NotNullable()  
 .WithColumn("email").AsString(255).NotNullable().Unique()  
 .WithColumn("password\_hash").AsString(255).NotNullable()  
 .WithColumn("about").AsCustom("TEXT").NotNullable().WithDefaultValue("")  
 .WithColumn("score").AsInt64().NotNullable().WithDefaultValue(0)  
 .WithColumn("role\_id").AsInt64().NotNullable();  
   
 Create.ForeignKey("FK\_users\_roles")  
 .FromTable("users").ForeignColumn("role\_id")  
 .ToTable("roles").PrimaryColumn("id");  
   
 Create.Index("IX\_users\_role\_id")  
 .OnTable("users")  
 .OnColumn("role\_id")  
 .Ascending();  
   
 Alter.Table("exercises")  
 .AddColumn("author\_id").AsInt64().NotNullable();  
   
 Create.ForeignKey("FK\_exercises\_users")  
 .FromTable("exercises").ForeignColumn("author\_id")  
 .ToTable("users").PrimaryColumn("id");  
   
 Create.Index("IX\_exercises\_owner\_id")  
 .OnTable("exercises")  
 .OnColumn("author\_id")  
 .Ascending();  
   
 Delete.Column("by\_exercise\_author").FromTable("solutions");  
   
 Alter.Table("solutions")  
 .AddColumn("author\_id").AsInt64().NotNullable();  
   
 Create.ForeignKey("FK\_solutions\_users")  
 .FromTable("solutions").ForeignColumn("author\_id")  
 .ToTable("users").PrimaryColumn("id");  
   
 Create.Index("IX\_solutions\_owner\_id")  
 .OnTable("solutions")  
 .OnColumn("author\_id")  
 .Ascending();  
 }  
  
 *public override void* Down()  
 {  
 Delete.Index("IX\_solutions\_author\_id").OnTable("solutions");  
 Delete.ForeignKey("FK\_solutions\_users").OnTable("solutions");  
   
 Alter.Table("solutions")  
 .AddColumn("by\_exercise\_author").AsBoolean().NotNullable().WithDefaultValue(*false*);  
   
 Delete.Column("author\_id").FromTable("solutions");  
   
 Delete.Index("IX\_exercises\_author\_id").OnTable("exercises");  
 Delete.ForeignKey("FK\_exercises\_users").OnTable("exercises");  
   
 Delete.Column("author\_id").FromTable("exercises");  
   
 Delete.Index("IX\_users\_email").OnTable("users");  
 Delete.Index("IX\_users\_role\_id").OnTable("users");  
   
 Delete.ForeignKey("FK\_users\_roles").OnTable("users");  
   
 Delete.Table("users");  
   
 Delete.Table("roles");  
 }  
}

**M0008AddExerciseLikes.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_13\_1448)]   
*public class* M0008AddExerciseLikes : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("exercise\_likes")  
 .WithColumn("exercise\_id").AsInt64().NotNullable()  
 .WithColumn("user\_id").AsInt64().NotNullable();  
   
 Create.PrimaryKey("PK\_exercise\_likes")  
 .OnTable("exercise\_likes")  
 .Columns(["exercise\_id", "user\_id"]);  
   
 Create.ForeignKey("FK\_exercise\_likes\_exercises")  
 .FromTable("exercise\_likes").ForeignColumn("exercise\_id")  
 .ToTable("exercises").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
  
 Create.ForeignKey("FK\_exercise\_likes\_users")  
 .FromTable("exercise\_likes").ForeignColumn("user\_id")  
 .ToTable("users").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
   
 Create.Index("IX\_exercise\_likes\_exercise\_id")  
 .OnTable("exercise\_likes")  
 .OnColumn("exercise\_id")  
 .Ascending();  
   
 Create.Index("IX\_exercise\_likes\_user\_id")  
 .OnTable("exercise\_likes")  
 .OnColumn("user\_id")  
 .Ascending();  
 }  
}

**M0009AddSolutionLikes.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_13\_1757)]   
*public class* M0009AddSolutionLikes : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("solution\_likes")  
 .WithColumn("solution\_id").AsInt64().NotNullable()  
 .WithColumn("user\_id").AsInt64().NotNullable();  
   
 Create.PrimaryKey("PK\_solution\_likes")  
 .OnTable("solution\_likes")  
 .Columns(["solution\_id", "user\_id"]);  
   
 Create.ForeignKey("FK\_solution\_likes\_solutions")  
 .FromTable("solution\_likes").ForeignColumn("solution\_id")  
 .ToTable("solutions").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
  
 Create.ForeignKey("FK\_solution\_likes\_users")  
 .FromTable("solution\_likes").ForeignColumn("user\_id")  
 .ToTable("users").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
   
 Create.Index("IX\_solution\_likes\_solution\_id")  
 .OnTable("solution\_likes")  
 .OnColumn("solution\_id")  
 .Ascending();  
   
 Create.Index("IX\_solution\_likes\_user\_id")  
 .OnTable("solution\_likes")  
 .OnColumn("user\_id")  
 .Ascending();  
 }  
}

**M0010AddUserHints.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_13\_1853)]   
*public class* M0010AddUserHints : AutoReversingMigration  
{  
 *public override void* Up()  
 {  
 Create.Table("user\_hints")  
 .WithColumn("hint\_id").AsInt64().NotNullable()  
 .WithColumn("user\_id").AsInt64().NotNullable();  
   
 Create.PrimaryKey("PK\_user\_hints")  
 .OnTable("user\_hints")  
 .Columns(["hint\_id", "user\_id"]);  
   
 Create.ForeignKey("FK\_user\_hints\_hints")  
 .FromTable("user\_hints").ForeignColumn("hint\_id")  
 .ToTable("hints").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
  
 Create.ForeignKey("FK\_user\_hints\_users")  
 .FromTable("user\_hints").ForeignColumn("user\_id")  
 .ToTable("users").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
   
 Create.Index("IX\_user\_hints\_hint\_id")  
 .OnTable("user\_hints")  
 .OnColumn("hint\_id")  
 .Ascending();  
   
 Create.Index("IX\_user\_hints\_user\_id")  
 .OnTable("user\_hints")  
 .OnColumn("user\_id")  
 .Ascending();  
 }  
}

**M0011AddedOnDeleteForSolutions.cs**

*using* FluentMigrator;  
  
*namespace* Coursework.Database.Migrations;  
  
[Migration(2025\_04\_14\_1238)]   
*public class* M0011AddedOnDeleteRules : Migration  
{  
 *public override void* Up()  
 {  
 Delete.ForeignKey("FK\_solutions\_users").OnTable("solutions");  
 Delete.ForeignKey("FK\_exercises\_users").OnTable("exercises");  
 Delete.ForeignKey("FK\_exercises\_programing\_languages").OnTable("exercises");  
   
 Create.ForeignKey("FK\_solutions\_users")  
 .FromTable("solutions").ForeignColumn("author\_id")  
 .ToTable("users").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
  
 Create.ForeignKey("FK\_exercises\_users")  
 .FromTable("exercises").ForeignColumn("author\_id")  
 .ToTable("users").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
  
 Create.ForeignKey("FK\_exercises\_programing\_languages")  
 .FromTable("exercises").ForeignColumn("language\_id")  
 .ToTable("programing\_languages").PrimaryColumn("id")  
 .OnDelete(System.Data.Rule.Cascade);  
 }  
  
 *public override void* Down()  
 {  
 Delete.ForeignKey("FK\_solutions\_users").OnTable("solutions");  
 Delete.ForeignKey("FK\_exercises\_users").OnTable("exercises");  
 Delete.ForeignKey("FK\_exercises\_programing\_languages").OnTable("exercises");  
   
 Create.ForeignKey("FK\_solutions\_users")  
 .FromTable("solutions").ForeignColumn("author\_id")  
 .ToTable("users").PrimaryColumn("id");  
  
 Create.ForeignKey("FK\_exercises\_users")  
 .FromTable("exercises").ForeignColumn("author\_id")  
 .ToTable("users").PrimaryColumn("id");  
  
 Create.ForeignKey("FK\_exercises\_programing\_languages")  
 .FromTable("exercises").ForeignColumn("language\_id")  
 .ToTable("programing\_languages").PrimaryColumn("id");  
 }  
}

**MigrationRunner.cs**

*using* FluentMigrator.Runner;  
*using* Npgsql;  
  
*namespace* Coursework.Database;  
  
*public static class* MigrationRunner  
{  
 *public static void RunMigrations*(*string*? connectionString, *ILogger* logger)  
 {  
 *if*(*string*.*IsNullOrWhiteSpace*(connectionString))  
 {  
 logger.LogWarning("connectionString пуста, миграции не применены");  
 *return*;  
 };  
 *try* {  
 // Проверяем существование базы данных  
 *var* builder = *new* NpgsqlConnectionStringBuilder(connectionString);  
 *var* databaseName = builder.Database;  
  
 *using* (*var* connection = *new* NpgsqlConnection(builder.ConnectionString.Replace($"Database={databaseName}", "Database=postgres")))  
 {  
 connection.Open();  
 *var* command = *new* NpgsqlCommand($"SELECT 1 FROM pg\_database WHERE datname='{databaseName}'", connection);  
 *var* exists = command.ExecuteScalar() != *null*;  
  
 *if* (!exists)  
 {  
 logger.LogInformation("База данных {DatabaseName} не существует. Создаём её...", databaseName);  
 command.CommandText = $"CREATE DATABASE \"{databaseName}\"";  
 command.ExecuteNonQuery();  
 }  
 }  
   
 *var* serviceProvider = *new* ServiceCollection()  
 .AddFluentMigratorCore()  
 .ConfigureRunner(rb => rb  
 .AddPostgres()  
 .WithGlobalConnectionString(connectionString)  
 .ScanIn(*typeof*(MigrationRunner).Assembly).For.Migrations())  
 .AddLogging(lb => lb.AddFluentMigratorConsole())  
 .BuildServiceProvider();  
 *using var* scope = serviceProvider.CreateScope();  
 *var* runner = scope.ServiceProvider.GetRequiredService<*IMigrationRunner*>();  
 runner.MigrateUp();  
 }  
 *catch* {  
 logger.LogError("Ошибка применения миграций");  
 *throw*;  
 }  
   
 }  
}

**DifficultyLevelRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* DifficultyLevelRepository(*IDbConnection* connection, *IDbTransaction* transaction) : *IDifficultyLevelRepository*{  
 *public async* Task<List<DifficultyLevel>> GetAllAsync()  
 {  
 *const string* sql = """  
 SELECT id, name  
 FROM difficulty\_levels  
 ORDER BY id  
 """;  
  
 *var* result = *await* connection.QueryAsync<DifficultyLevel>(sql, transaction);  
 *return* result.ToList();  
 }  
}

**ExerciseRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* ExerciseRepository(*IDbConnection* connection, *IDbTransaction* transaction)  
 : *IExerciseRepository*{  
 *public async* Task<*long*> AddAsync(Exercise exercise)  
 {  
 ArgumentNullException.*ThrowIfNull*(exercise);  
  
 *const string* insertExerciseSql = """  
 INSERT INTO exercises (  
 name, difficulty\_id, language\_id, score, short\_description, full\_description,   
 is\_published, s3\_key\_source, s3\_key\_tests, author\_id  
 )  
 VALUES (  
 @Name, @DifficultyId, @LanguageId, @Score, @ShortDescription, @FullDescription,   
 @IsPublished, @S3KeySource, @S3KeyTests, @AuthorId  
 )  
 RETURNING id  
 """;  
  
 *var* exerciseId = *await* connection.QuerySingleOrDefaultAsync<*long*>(  
 insertExerciseSql, exercise, transaction);  
  
 *if* (exercise.Frameworks.Any())  
 {  
 *const string* insertFrameworksSql = """  
 INSERT INTO frameworks\_exercises (framework\_id, exercise\_id)  
 VALUES (@FrameworkId, @ExerciseId)  
 """;  
  
 *var* frameworkParams = exercise.Frameworks  
 .Where(f => f.Id > 0)  
 .Select(f => *new* { FrameworkId = f.Id, ExerciseId = exerciseId });  
  
 *await* connection.ExecuteAsync(insertFrameworksSql, frameworkParams, transaction);  
 }  
  
 *if* (exercise.Hints.Any())  
 {  
 *foreach* (*var* hint *in* exercise.Hints)  
 hint.ExerciseId = exerciseId;  
  
 *const string* insertHintsSql = """  
 INSERT INTO hints (exercise\_id, cost, text)  
 VALUES (@ExerciseId, @Cost, @Text)  
 """;  
  
 *await* connection.ExecuteAsync(insertHintsSql, exercise.Hints, transaction);  
 }  
  
 *return* exerciseId;  
 }  
  
 *public async* Task UpdateAsync(Exercise exercise)  
 {  
 ArgumentNullException.*ThrowIfNull*(exercise);  
  
 *const string* updateExerciseSql = """  
 UPDATE exercises  
 SET   
 name = @Name, difficulty\_id = @DifficultyId, language\_id = @LanguageId, score = @Score,   
 short\_description = @ShortDescription, full\_description = @FullDescription,   
 is\_published = @IsPublished, s3\_key\_source = @S3KeySource, s3\_key\_tests = @S3KeyTests  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(updateExerciseSql, exercise, transaction);  
  
 *const string* deleteFrameworksSql = """  
 DELETE FROM frameworks\_exercises  
 WHERE exercise\_id = @ExerciseId  
 """;  
  
 *const string* deleteHintsSql = """  
 DELETE FROM hints  
 WHERE exercise\_id = @ExerciseId  
 """;  
  
 *await* connection.ExecuteAsync(deleteFrameworksSql, *new* { ExerciseId = exercise.Id }, transaction);  
 *await* connection.ExecuteAsync(deleteHintsSql, *new* { ExerciseId = exercise.Id }, transaction);  
  
  
 *if* (exercise.Frameworks.Any())  
 {  
 *const string* insertFrameworksSql = """  
 INSERT INTO frameworks\_exercises (framework\_id, exercise\_id)  
 VALUES (@FrameworkId, @ExerciseId)  
 """;  
  
 *var* frameworkParams = exercise.Frameworks  
 .Where(f => f.Id > 0)  
 .Select(f => *new* { FrameworkId = f.Id, ExerciseId = exercise.Id });  
  
 *await* connection.ExecuteAsync(insertFrameworksSql, frameworkParams, transaction);  
 }  
  
 *if* (exercise.Hints.Any())  
 {  
 *foreach* (*var* hint *in* exercise.Hints)  
 hint.ExerciseId = exercise.Id;  
  
 *const string* insertHintsSql = """  
 INSERT INTO hints (exercise\_id, cost, text)  
 VALUES (@ExerciseId, @Cost, @Text)  
 """;  
  
 *await* connection.ExecuteAsync(insertHintsSql, exercise.Hints, transaction);  
 }  
 }  
  
 *public async* Task DeleteAsync(*long* id)  
 {  
 *const string* deleteExerciseSql = """  
 DELETE FROM exercises  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(deleteExerciseSql, *new* { Id = id }, transaction);  
 }  
  
 // Add this method to the ExerciseRepository class  
 *public async* Task OpenHintAsync(*long* hintId, *long* userId)  
 {  
 *const string* sql = """  
 INSERT INTO user\_hints (hint\_id, user\_id)  
 VALUES (@HintId, @UserId)  
 ON CONFLICT DO NOTHING  
 """;  
  
 *await* connection.ExecuteAsync(sql, *new* { HintId = hintId, UserId = userId }, transaction);  
 }  
  
 *public async* Task<Exercise?> GetAsync(*long* id)  
 {  
 *const string* sql = """  
 SELECT e.\*,  
 d.\*,  
 l.\*,  
 u.\*,  
 h.\*  
 FROM exercises e  
 LEFT JOIN difficulty\_levels d ON e.difficulty\_id = d.id  
 LEFT JOIN programing\_languages l ON e.language\_id = l.id  
 LEFT JOIN users u ON e.author\_id = u.id  
 LEFT JOIN hints h ON e.id = h.exercise\_id  
 WHERE e.id = @Id  
 ORDER BY h.cost  
 """;  
  
 *const string* frameworksSql = """  
 SELECT f.\*  
 FROM frameworks f  
 INNER JOIN frameworks\_exercises fe ON f.id = fe.framework\_id  
 WHERE fe.exercise\_id = @ExerciseId  
 """;  
  
 *const string* solutionsSql = """  
 SELECT s.\*,  
 u.\*  
 FROM solutions s  
 LEFT JOIN users u ON s.author\_id = u.id  
 WHERE s.exercise\_id = @ExerciseId  
 """;  
  
 *const string* likesSql = """  
 SELECT u.\*  
 FROM users u  
 INNER JOIN exercise\_likes el ON u.id = el.user\_id  
 WHERE el.exercise\_id = @ExerciseId  
 """;  
  
 *const string* hintUsersSql = """  
 SELECT h.id as hint\_id, u.\*  
 FROM users u  
 INNER JOIN user\_hints uh ON u.id = uh.user\_id  
 INNER JOIN hints h ON uh.hint\_id = h.id  
 WHERE h.exercise\_id = @ExerciseId  
 """;  
  
 *var* hintsDict = *new* Dictionary<*long*, Hint>();  
 Exercise? result = *null*;  
  
 *await* connection.QueryAsync<Exercise, DifficultyLevel, ProgrammingLanguage, User, Hint, Exercise>(  
 sql,  
 (exercise, difficulty, language, author, hint) =>  
 {  
 *if* (result == *null*)  
 {  
 result = exercise;  
 result.Difficulty = difficulty;  
 result.Language = language;  
 result.Author = author;  
 result.Hints = *new* List<Hint>();  
 }  
  
 *if* (hint *is not null* && hintsDict.TryAdd(hint.Id, hint))  
 {  
 hint.OpenedByUsers = *new* List<User>();  
 result.Hints.Add(hint);  
 }  
  
 *return* result;  
 },  
 *new* { Id = id },  
 transaction,  
 splitOn: "id"  
 );  
  
 *if* (result == *null*) *return null*;  
  
 result.Frameworks = (*await* connection.QueryAsync<Framework>(  
 frameworksSql,  
 *new* { ExerciseId = id },  
 transaction  
 )).ToList();  
  
 *var* solutions = *await* connection.QueryAsync<Solution, User, Solution>(  
 solutionsSql,  
 (solution, author) =>  
 {  
 solution.Author = author;  
 *return* solution;  
 },  
 *new* { ExerciseId = id },  
 transaction,  
 splitOn: "id"  
 );  
  
 *var* solutionsList = solutions.ToList();  
  
 result.AuthorSolution = solutionsList.FirstOrDefault(s => s.AuthorId == result.AuthorId);  
 result.Solutions = solutionsList.Where(s => s.AuthorId != result.AuthorId && s.Id > 0).ToList();  
  
 result.LikedByUsers = (*await* connection.QueryAsync<User>(  
 likesSql,  
 *new* { ExerciseId = id },  
 transaction  
 )).ToList();  
  
 *var* hintUsers = *await* connection.QueryAsync<*long*, User, (*long* HintId, User User)>(  
 hintUsersSql,  
 (hintId, user) => (hintId, user),  
 *new* { ExerciseId = id },  
 transaction,  
 splitOn: "id"  
 );  
  
 *foreach* (*var* (hintId, user) *in* hintUsers)  
 {  
 *var* hint = result.Hints.FirstOrDefault(h => h.Id == hintId);  
 hint?.OpenedByUsers.Add(user);  
 }  
  
 *return* result;  
 }  
  
 *public async* Task<List<Exercise>> GetAllAsync()  
 {  
 *return await* GetAllAsync(*null*);  
 }  
  
 *public async* Task<List<Exercise>> GetAllAsync(*string*? search)  
 {  
 *const string* sql = """  
 SELECT e.\*,  
 d.\*,  
 l.\*,  
 u.\*  
 FROM exercises e  
 LEFT JOIN difficulty\_levels d ON e.difficulty\_id = d.id  
 LEFT JOIN programing\_languages l ON e.language\_id = l.id  
 LEFT JOIN users u ON e.author\_id = u.id  
 WHERE @SearchPattern IS NULL   
 OR e.name LIKE @SearchPattern  
 OR e.short\_description LIKE @SearchPattern  
 OR l.name LIKE @SearchPattern  
 ORDER BY e.id  
 """;  
  
 *var* exercises = *await* connection.QueryAsync<Exercise, DifficultyLevel, ProgrammingLanguage, User, Exercise>(  
 sql,  
 (exercise, difficulty, language, author) =>  
 {  
 exercise.Difficulty = difficulty;  
 exercise.Language = language;  
 exercise.Author = author;  
 *return* exercise;  
 },  
 *new* {  
 SearchPattern = search != *null* ? $"%{search}%" : *null* },  
 transaction: transaction,  
 splitOn: "id"  
 );  
  
 *var* result = exercises.ToList();  
  
 *const string* likesSql = """  
 SELECT el.exercise\_id, u.\*  
 FROM exercise\_likes el  
 JOIN users u ON el.user\_id = u.id  
 WHERE el.exercise\_id = ANY(@ExerciseIds)  
 """;  
  
 *var* exerciseIds = result.Select(e => e.Id).ToArray();  
 *if* (exerciseIds.Length == 0) *return* result;  
  
 *var* likesDict = *new* Dictionary<*long*, List<User>>();  
  
 *var* likes = *await* connection.QueryAsync<*long*, User, (*long* ExerciseId, User User)>(  
 likesSql,  
 (exerciseId, user) => (exerciseId, user),  
 *new* { ExerciseIds = exerciseIds },  
 transaction,  
 splitOn: "id"  
 );  
  
 *foreach* (*var* (exerciseId, user) *in* likes)  
 {  
 *if* (!likesDict.TryGetValue(exerciseId, *out var* users))  
 {  
 users = *new* List<User>();  
 likesDict[exerciseId] = users;  
 }  
  
 users.Add(user);  
 }  
  
 *foreach* (*var* exercise *in* result)  
 {  
 *if* (likesDict.TryGetValue(exercise.Id, *out var* users))  
 {  
 exercise.LikedByUsers = users;  
 }  
 }  
  
 *return* result;  
 }  
  
 *public async* Task AddLikeAsync(*long* exerciseId, *long* userId)  
 {  
 *const string* sql = """  
 INSERT INTO exercise\_likes (exercise\_id, user\_id)  
 VALUES (@ExerciseId, @UserId)  
 ON CONFLICT DO NOTHING  
 """;  
  
 *await* connection.ExecuteAsync(sql, *new* { ExerciseId = exerciseId, UserId = userId }, transaction);  
 }  
  
 *public async* Task RemoveLikeAsync(*long* exerciseId, *long* userId)  
 {  
 *const string* sql = """  
 DELETE FROM exercise\_likes  
 WHERE exercise\_id = @ExerciseId AND user\_id = @UserId  
 """;  
  
 *await* connection.ExecuteAsync(sql, *new* { ExerciseId = exerciseId, UserId = userId }, transaction);  
 }  
}

**FrameworkRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* FrameworkRepository(*IDbConnection* connection, *IDbTransaction* transaction)  
 : *IFrameworkRepository*{  
 *public async* Task<*long*> AddAsync(Framework language)  
 {  
 ArgumentNullException.*ThrowIfNull*(language);  
  
 *const string* sql = """  
 INSERT INTO frameworks (name, description, language\_id)  
 VALUES (@Name, @Description, @LanguageId)  
 RETURNING id  
 """;  
  
 *var* result = *await* connection.QuerySingleOrDefaultAsync<*long*>(  
 sql, language, transaction);  
  
 *return* result;  
 }  
  
 *public async* Task UpdateAsync(Framework language)  
 {  
 ArgumentNullException.*ThrowIfNull*(language);  
  
 *const string* sql = """  
 UPDATE frameworks  
 SET name = @Name, description = @Description, language\_id = @LanguageId  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(sql, language, transaction);  
 }  
  
 *public async* Task DeleteAsync(*long* id)  
 {  
 *const string* sql = """  
 DELETE FROM frameworks  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(sql, *new* { Id = id }, transaction);  
 }  
  
 *public async* Task<Framework?> GetAsync(*long* id)  
 {  
 *const string* sql = """  
 SELECT \*  
 FROM frameworks  
 WHERE id = @Id  
 LIMIT 1  
 """;  
  
 *var* result = *await* connection.QuerySingleOrDefaultAsync<Framework?>(  
 sql, *new* { Id = id }, transaction);  
  
 *return* result;  
 }  
  
 *public async* Task<List<Framework>> GetAllAsync()  
 {  
 *const string* sql = """  
 SELECT \*  
 FROM frameworks  
 ORDER BY id  
 """;  
  
 *var* result = *await* connection.QueryAsync<Framework>(sql, transaction);  
 *return* result.ToList();  
 }  
  
 *public async* Task<List<Framework>> GetAllWithLanguageAsync()  
 {  
 *const string* sql = """  
 SELECT   
 f.\*,   
 pl.\*  
 FROM frameworks f  
 LEFT JOIN programing\_languages pl ON f.language\_id = pl.id  
 ORDER BY f.id  
 """;  
  
 *var* result = *await* connection.QueryAsync<Framework, ProgrammingLanguage, Framework>(  
 sql,  
 (framework, language) =>  
 {  
 framework.Language = language;  
 *return* framework;  
 },  
 splitOn: "id",  
 transaction: transaction  
 );  
  
 *return* result.ToList();  
 }  
}

**ProgrammingLanguageRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* ProgrammingLanguageRepository(*IDbConnection* connection, *IDbTransaction* transaction)  
 : *IProgrammingLanguageRepository*{  
 *public async* Task<*long*> AddAsync(ProgrammingLanguage language)  
 {  
 ArgumentNullException.*ThrowIfNull*(language);  
  
 *const string* sql = """  
 INSERT INTO programing\_languages (name, description)  
 VALUES (@Name, @Description)  
 RETURNING id  
 """;  
  
 *var* result = *await* connection.QuerySingleOrDefaultAsync<*long*>(  
 sql, language, transaction);  
  
 *return* result;  
 }  
  
 *public async* Task UpdateAsync(ProgrammingLanguage language)  
 {  
 ArgumentNullException.*ThrowIfNull*(language);  
  
 *const string* sql = """  
 UPDATE programing\_languages  
 SET name = @Name, description = @Description  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(sql, language, transaction);  
 }  
  
 *public async* Task DeleteAsync(*long* id)  
 {  
 *const string* sql = """  
 DELETE FROM programing\_languages  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(sql, *new* { Id = id }, transaction);  
 }  
  
 *public async* Task<ProgrammingLanguage?> GetAsync(*long* id)  
 {  
 *const string* sql = """  
 SELECT id, name, description  
 FROM programing\_languages  
 WHERE id = @Id  
 LIMIT 1  
 """;  
  
 *var* result = *await* connection.QuerySingleOrDefaultAsync<ProgrammingLanguage?>(  
 sql, *new* { Id = id }, transaction);  
  
 *return* result;  
 }  
  
 *public async* Task<List<ProgrammingLanguage>> GetAllAsync()  
 {  
 *const string* sql = """  
 SELECT id, name, description  
 FROM programing\_languages  
 ORDER BY id  
 """;  
  
 *var* result = *await* connection.QueryAsync<ProgrammingLanguage>(sql, transaction);  
 *return* result.ToList();  
 }  
}

**RoleRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* RoleRepository(*IDbConnection* connection, *IDbTransaction* transaction) : *IRoleRepository*{  
 *public async* Task<Role?> GetAsync(*long* id)  
 {  
 *const string* sql = """  
 SELECT id, name  
 FROM roles  
 WHERE id = @Id  
 """;  
  
 *return await* connection.QueryFirstOrDefaultAsync<Role>(sql, *new* { Id = id }, transaction);  
 }  
}

**SolutionRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* SolutionRepository(*IDbConnection* connection, *IDbTransaction* transaction) : *ISolutionRepository*{  
 *public async* Task<*long*> AddAsync(Solution solution)  
 {  
 ArgumentNullException.*ThrowIfNull*(solution);  
  
 *const string* sql = """  
 INSERT INTO solutions (s3\_key, exercise\_id, author\_id)  
 VALUES (@S3Key, @ExerciseId, @AuthorId)  
 RETURNING id  
 """;  
  
 *var* result = *await* connection.QuerySingleOrDefaultAsync<*long*>(  
 sql, solution, transaction);  
  
 *return* result;  
 }  
  
 *public async* Task UpdateAsync(Solution solution)  
 {  
 ArgumentNullException.*ThrowIfNull*(solution);  
  
 *const string* sql = """  
 UPDATE solutions  
 SET s3\_key = @S3Key, exercise\_id = @ExerciseId  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(sql, solution, transaction);  
 }  
  
 *public async* Task DeleteAsync(*long* id)  
 {  
 *const string* sql = """  
 DELETE FROM solutions  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(sql, *new* { Id = id }, transaction);  
 }  
  
  
 *public async* Task AddLikeAsync(*long* solutionId, *long* userId)  
 {  
 *const string* sql = """  
 INSERT INTO solution\_likes (solution\_id, user\_id)  
 VALUES (@SolutionId, @UserId)  
 ON CONFLICT DO NOTHING  
 """;  
   
 *await* connection.ExecuteAsync(sql, *new* { SolutionId = solutionId, UserId = userId }, transaction);  
 }  
  
 *public async* Task RemoveLikeAsync(*long* solutionId, *long* userId)  
 {  
 *const string* sql = """  
 DELETE FROM solution\_likes  
 WHERE solution\_id = @SolutionId AND user\_id = @UserId  
 """;  
   
 *await* connection.ExecuteAsync(sql, *new* { SolutionId = solutionId, UserId = userId }, transaction);  
 }  
   
 *public async* Task<Solution?> GetAsync(*long* id)  
 {  
 *const string* sql = """  
 SELECT s.\*,   
 e.\*,  
 u.\*  
 FROM solutions s  
 LEFT JOIN exercises e ON s.exercise\_id = e.id  
 LEFT JOIN users u ON s.author\_id = u.id  
 WHERE s.id = @Id  
 LIMIT 1  
 """;  
   
 *const string* likesSql = """  
 SELECT u.\*  
 FROM users u  
 INNER JOIN solution\_likes sl ON u.id = sl.user\_id  
 WHERE sl.solution\_id = @SolutionId  
 """;  
   
 *var* result = *await* connection.QueryAsync<Solution, Exercise, User, Solution>(  
 sql,  
 (solution, exercise, author) =>  
 {  
 solution.Exercise = exercise;  
 solution.Author = author;  
 *return* solution;  
 },  
 *new* { Id = id },  
 transaction,  
 splitOn: "id"  
 );  
   
 *var* solution = result.FirstOrDefault();  
 *if* (solution != *null*)  
 {  
 solution.LikedByUsers = (*await* connection.QueryAsync<User>(  
 likesSql,  
 *new* { SolutionId = id },  
 transaction  
 )).ToList();  
 }  
   
 *return* solution;  
 }  
  
 *public async* Task<List<Solution>> GetAllAsync()  
 {  
 *const string* sql = """  
 SELECT s.\*,  
 u.\*  
 FROM solutions s  
 LEFT JOIN users u ON s.author\_id = u.id  
 ORDER BY s.id  
 """;  
  
 *var* result = *await* connection.QueryAsync<Solution, User, Solution>(  
 sql,  
 (solution, author) =>  
 {  
 solution.Author = author;  
 *return* solution;  
 },  
 transaction,  
 splitOn: "id"  
 );  
   
 *return* result.ToList();  
 }  
  
 *public async* Task<List<Solution>> GetAllByExerciseAsync(*long* exerciseId)  
 {  
 *const string* sql = """  
 SELECT s.\*,  
 u.\*  
 FROM solutions s  
 LEFT JOIN users u ON s.author\_id = u.id  
 WHERE s.exercise\_id = @ExerciseId  
 ORDER BY s.id  
 """;  
  
 *var* result = *await* connection.QueryAsync<Solution, User, Solution>(  
 sql,   
 (solution, author) =>  
 {  
 solution.Author = author;  
 *return* solution;  
 },  
 *new* { ExerciseId = exerciseId },   
 transaction,  
 splitOn: "id"  
 );  
   
 *return* result.ToList();  
 }  
}

**UserRepository.cs**

*using* System.Data;  
*using* Coursework.Interfaces.Database.Repositories;  
*using* Coursework.Models.Entities;  
*using* Dapper;  
  
*namespace* Coursework.Database.Repositories;  
  
*public class* UserRepository(*IDbConnection* connection, *IDbTransaction* transaction)  
 : *IUserRepository*{  
 *public async* Task<*long*> AddAsync(User user)  
 {  
 ArgumentNullException.*ThrowIfNull*(user);  
   
 *const string* insertUserSql = """  
 INSERT INTO users (  
 name, email, password\_hash, about, score, role\_id  
 )  
 VALUES (  
 @Name, @Email, @PasswordHash, @About, @Score, @RoleId  
 )  
 RETURNING id  
 """;  
  
 *var* userId = *await* connection.QuerySingleOrDefaultAsync<*long*>(  
 insertUserSql, user, transaction);  
  
 *return* userId;  
 }  
  
 *public async* Task UpdateAsync(User user)  
 {  
 ArgumentNullException.*ThrowIfNull*(user);  
   
 *const string* updateUserSql = """  
 UPDATE users  
 SET   
 name = @Name,   
 email = @Email,   
 password\_hash = @PasswordHash,   
 about = @About,   
 score = @Score,   
 role\_id = @RoleId  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(updateUserSql, user, transaction);  
 }  
  
 *public async* Task DeleteAsync(*long* id)  
 {  
 *const string* deleteUserSql = """  
 DELETE FROM users  
 WHERE id = @Id  
 """;  
  
 *await* connection.ExecuteAsync(deleteUserSql, *new* { Id = id }, transaction);  
 }  
  
 *public async* Task<User?> GetAsync(*long* id)  
 {  
 *const string* sql = """  
 SELECT u.\*,  
 r.\*  
 FROM users u  
 LEFT JOIN roles r ON u.role\_id = r.id  
 WHERE u.id = @Id  
 """;  
   
 *const string* exercisesSql = """  
 SELECT e.\*,  
 d.\*,  
 l.\*  
 FROM exercises e  
 LEFT JOIN difficulty\_levels d ON e.difficulty\_id = d.id  
 LEFT JOIN programing\_languages l ON e.language\_id = l.id  
 WHERE e.author\_id = @UserId  
 """;  
   
 *const string* solutionsSql = """  
 SELECT s.\*,  
 e.\*  
 FROM solutions s  
 LEFT JOIN exercises e ON s.exercise\_id = e.id  
 WHERE s.author\_id = @UserId  
 """;  
   
 *var* userResult = *await* connection.QueryAsync<User, Role, User>(  
 sql,  
 (user, role) =>  
 {  
 user.Role = role;  
 *return* user;  
 },  
 *new* { Id = id },  
 transaction,  
 splitOn: "id"  
 );  
   
 *var* user = userResult.FirstOrDefault();  
 *if* (user == *null*) *return null*;  
   
 *var* exercises = *await* connection.QueryAsync<Exercise, DifficultyLevel, ProgrammingLanguage, Exercise>(  
 exercisesSql,  
 (exercise, difficulty, language) =>  
 {  
 exercise.Difficulty = difficulty;  
 exercise.Language = language;  
 *return* exercise;  
 },  
 *new* { UserId = id },  
 transaction,  
 splitOn: "id"  
 );  
   
 user.Exercises = exercises.ToList();  
   
 *var* solutions = *await* connection.QueryAsync<Solution, Exercise, Solution>(  
 solutionsSql,  
 (solution, exercise) =>  
 {  
 solution.Exercise = exercise;  
 *return* solution;  
 },  
 *new* { UserId = id },  
 transaction,  
 splitOn: "id"  
 );  
   
 user.Solutions = solutions.ToList();  
  
 *return* user;  
 }  
  
 *public async* Task<List<User>> GetAllAsync()  
 {  
 *const string* sql = """  
 SELECT u.\*,  
 r.\*  
 FROM users u  
 LEFT JOIN roles r ON u.role\_id = r.id  
 ORDER BY u.id  
 """;  
  
 *var* users = *await* connection.QueryAsync<User, Role, User>(  
 sql,  
 (user, role) =>  
 {  
 user.Role = role;  
 *return* user;  
 },  
 transaction: transaction,  
 splitOn: "id"  
 );  
  
 *return* users.ToList();  
 }  
   
 *public async* Task<User?> GetByEmailAsync(*string* email)  
 {  
 *const string* sql = """  
 SELECT u.\*,  
 r.\*  
 FROM users u  
 LEFT JOIN roles r ON u.role\_id = r.id  
 WHERE u.email = @Email  
 """;  
   
 *var* userResult = *await* connection.QueryAsync<User, Role, User>(  
 sql,  
 (user, role) =>  
 {  
 user.Role = role;  
 *return* user;  
 },  
 *new* { Email = email },  
 transaction,  
 splitOn: "id"  
 );  
   
 *var* user = userResult.FirstOrDefault();  
 *if* (user == *null*) *return null*;  
   
 *const string* exercisesSql = """  
 SELECT e.\*,  
 d.\*,  
 l.\*  
 FROM exercises e  
 LEFT JOIN difficulty\_levels d ON e.difficulty\_id = d.id  
 LEFT JOIN programing\_languages l ON e.language\_id = l.id  
 WHERE e.author\_id = @UserId  
 """;  
   
 *var* exercises = *await* connection.QueryAsync<Exercise, DifficultyLevel, ProgrammingLanguage, Exercise>(  
 exercisesSql,  
 (exercise, difficulty, language) =>  
 {  
 exercise.Difficulty = difficulty;  
 exercise.Language = language;  
 *return* exercise;  
 },  
 *new* { UserId = user.Id },  
 transaction,  
 splitOn: "id"  
 );  
   
 user.Exercises = exercises.ToList();  
   
 *const string* solutionsSql = """  
 SELECT s.\*,  
 e.\*  
 FROM solutions s  
 LEFT JOIN exercises e ON s.exercise\_id = e.id  
 WHERE s.author\_id = @UserId  
 """;  
   
 *var* solutions = *await* connection.QueryAsync<Solution, Exercise, Solution>(  
 solutionsSql,  
 (solution, exercise) =>  
 {  
 solution.Exercise = exercise;  
 *return* solution;  
 },  
 *new* { UserId = user.Id },  
 transaction,  
 splitOn: "id"  
 );  
   
 user.Solutions = solutions.ToList();  
  
 *return* user;  
 }  
}

**AuthController.cs**

*using* Coursework.Interfaces.Database;  
*using* Coursework.Interfaces.Services;  
*using* Coursework.Models.DTOs;  
*using* Coursework.Models.Entities;  
*using* Microsoft.AspNetCore.Authentication;  
*using* Microsoft.AspNetCore.Authentication.JwtBearer;  
*using* Microsoft.AspNetCore.Authorization;  
*using* Microsoft.AspNetCore.Mvc;  
  
*namespace* Coursework.Controllers;  
  
*public class* AuthController(  
 *IUnitOfWorkFactory* uowFactory,   
 *IJwtService* jwtService,   
 *IPasswordService* passwordService,  
 *ILogger*<AuthController> logger) : Controller  
{  
 [HttpGet]  
 *public IActionResult* Login()  
 {  
 *if* (User.Identity?.IsAuthenticated == *true*)  
 {  
 *return* RedirectToAction("Index", "Home");  
 }  
   
 *return* View();  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Login(LoginDto model, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid)  
 {  
 *return* View(model);  
 }  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* user = *await* uow.Users.GetByEmailAsync(model.Email);  
   
 *if* (user == *null* || !passwordService.VerifyPassword(model.Password, user.PasswordHash))  
 {  
 ModelState.AddModelError(*string*.*Empty*, "Неверный email или пароль");  
 *return* View(model);  
 }  
   
 *var* token = jwtService.GenerateToken(user.Id, user.Name, user.Role?.Name ?? "Пользователь");  
   
 Response.Cookies.Append("jwt", token, *new* CookieOptions  
 {  
 HttpOnly = *true*,  
 Secure = *true*,  
 SameSite = SameSiteMode.Strict,  
 Expires = DateTime.UtcNow.AddHours(1)  
 });  
   
 logger.LogInformation("Пользователь {Email} успешно вошел в систему", user.Email);  
   
 *return* RedirectToAction("Index", "Home");  
 }  
   
 [HttpGet]  
 *public IActionResult* Register()  
 {  
 *if* (User.Identity?.IsAuthenticated == *true*)  
 {  
 *return* RedirectToAction("Index", "Home");  
 }  
   
 *return* View();  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Register(RegisterDto model, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid)  
 {  
 *return* View(model);  
 }  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
   
 *var* existingUser = *await* uow.Users.GetByEmailAsync(model.Email);  
 *if* (existingUser != *null*)  
 {  
 ModelState.AddModelError("Email", "Пользователь с таким email уже существует");  
 *return* View(model);  
 }  
   
 // Получаем роль "Пользователь" (ID = 2)  
 *const long* userRoleId = 2;  
 *var* role = *await* uow.Roles.GetAsync(userRoleId);  
 *if* (role *is null*)  
 *throw new* InvalidOperationException("Роль \"Пользователь\" отсутствует или задана корректно");  
   
 *var* user = *new* User  
 {  
 Name = model.Name,  
 Email = model.Email,  
 PasswordHash = passwordService.HashPassword(model.Password),  
 RoleId = role.Id,  
 About = "",  
 Score = 0  
 };  
   
 *var* userId = *await* uow.Users.AddAsync(user);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Зарегистрирован новый пользователь: {Email}", user.Email);  
  
 *var* token = jwtService.GenerateToken(userId, user.Name, role.Name);  
   
 Response.Cookies.Append("jwt", token, *new* CookieOptions  
 {  
 HttpOnly = *true*,  
 Secure = *true*,  
 SameSite = SameSiteMode.Strict,  
 Expires = DateTime.UtcNow.AddHours(1)  
 });  
   
 *return* RedirectToAction("Index", "Home");  
 }  
   
 [Authorize]  
 [HttpPost]  
 *public IActionResult* Logout()  
 {  
 Response.Cookies.Delete("jwt");  
   
 *return* RedirectToAction("Index", "Home");  
 }  
}

**ExercisesController.cs**

*using* Coursework.Extensions;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Models.DTOs;  
*using* Microsoft.AspNetCore.Mvc;  
*using* Microsoft.AspNetCore.Authorization;  
  
*namespace* Coursework.Controllers;  
  
[Authorize(Policy = "UserAccess")]  
*public class* ExercisesController(*IUnitOfWorkFactory* uowFactory, *ILogger*<HomeController> logger) : Controller  
{  
 *public async* Task<*IActionResult*> Index(*string* search, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* exercises = *await* uow.Exercises.GetAllAsync(search);  
  
 *var* currentUserId = User.GetId();  
 *var* exerciseDtos = exercises.Select(e => e.MapWithCurrentUser(currentUserId)).ToList();  
  
 ViewBag.Exercises = exerciseDtos.Where(x => x.IsPublished   
 || x.AuthorId == User.GetId()   
 || User.IsInRole("Администратор"));  
 ViewBag.Search = search;  
  
 *return* View();  
 }  
  
 *public async* Task<*IActionResult*> Create(CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
  
 *var* difficultyLevels = *await* uow.DifficultyLevels.GetAllAsync();  
 *var* languages = *await* uow.Languages.GetAllAsync();  
 *var* frameworks = *await* uow.Frameworks.GetAllAsync();  
  
 ViewBag.DifficultyLevels = difficultyLevels;  
 ViewBag.Languages = languages;  
 ViewBag.Frameworks = frameworks;  
  
 *return* View();  
 }  
  
 *public async* Task<*IActionResult*> Update(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* exercise = *await* uow.Exercises.GetAsync(id);  
 *if* (exercise *is null*)  
 *return* NotFound();  
 *if* (exercise.AuthorId != User.GetId() && !User.IsInRole("Администратор"))  
 *return* Forbid();  
  
 *var* difficultyLevels = *await* uow.DifficultyLevels.GetAllAsync();  
 *var* languages = *await* uow.Languages.GetAllAsync();  
 *var* frameworks = *await* uow.Frameworks.GetAllAsync();  
  
 ViewBag.DifficultyLevels = difficultyLevels;  
 ViewBag.Languages = languages;  
 ViewBag.Frameworks = frameworks;  
  
 *return* View(exercise.Map());  
 }  
  
 *public async* Task<*IActionResult*> View(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* exercise = *await* uow.Exercises.GetAsync(id);  
 *if* (exercise *is null*)  
 *return* NotFound();  
  
 *var* currentUserId = User.GetId();  
 *return* View(exercise.MapWithCurrentUser(currentUserId));  
 }  
  
 [HttpPost]  
 *public async* Task<*IActionResult*> Create(ExerciseDto exercise, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
  
 *if* (!ModelState.IsValid)  
 {  
 *var* difficultyLevels = *await* uow.DifficultyLevels.GetAllAsync();  
 *var* languages = *await* uow.Languages.GetAllAsync();  
 *var* frameworks = *await* uow.Frameworks.GetAllAsync();  
  
 ViewBag.DifficultyLevels = difficultyLevels;  
 ViewBag.Languages = languages;  
 ViewBag.Frameworks = frameworks;  
  
 *return* View(exercise);  
 }  
  
 exercise.AuthorId = User.GetId();  
 *var* exerciseEntity = exercise.Map();  
 *var* id = *await* uow.Exercises.AddAsync(exerciseEntity);  
 *var* solution = exerciseEntity.AuthorSolution;  
 solution!.ExerciseId = id;  
 *await* uow.Solutions.AddAsync(solution);  
 *await* uow.CommitAsync(ct);  
  
 logger.LogInformation(  
 "Добавлено упражнение. Id:{Id}, Название:{Name}, Баллы:{Score}, Автор: {AuthorId}",  
 id, exercise.Name, exercise.Score, exerciseEntity.AuthorId);  
  
 *return* RedirectToAction("Index");  
 }  
  
 [HttpPost]  
 *public async* Task<*IActionResult*> Update(ExerciseDto exercise, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
  
 *if* (!ModelState.IsValid)  
 {  
 *var* difficultyLevels = *await* uow.DifficultyLevels.GetAllAsync();  
 *var* languages = *await* uow.Languages.GetAllAsync();  
 *var* frameworks = *await* uow.Frameworks.GetAllAsync();  
  
 ViewBag.DifficultyLevels = difficultyLevels;  
 ViewBag.Languages = languages;  
 ViewBag.Frameworks = frameworks;  
  
 *return* View(exercise);  
 }  
  
 *var* prev = *await* uow.Exercises.GetAsync(exercise.Id);  
 *if* (prev *is null*)  
 *return* NotFound();  
 *if* (prev.AuthorId != User.GetId() && !User.IsInRole("Администратор"))  
 *return* Forbid();  
  
 *var* solution = prev.AuthorSolution;  
 solution!.S3Key = exercise.S3KeyAuthorSolution;  
 *await* uow.Solutions.UpdateAsync(solution);  
  
 *if* (prev.IsPublished)  
 exercise.IsPublished = *true*;  
  
 *await* uow.Exercises.UpdateAsync(exercise.Map());  
 *await* uow.CommitAsync(ct);  
  
 logger.LogInformation(  
 "Обновлено упражнение. Id:{Id}, Название:{LanguageName}, Баллы:{Score}",  
 exercise.Id, exercise.Name, exercise.Score);  
  
 *return* RedirectToAction("Index");  
 }  
  
 [HttpPost]  
 *public async* Task<*IActionResult*> Delete(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* exercise = *await* uow.Exercises.GetAsync(id);  
 *if* (exercise *is null*)  
 *return* NotFound();  
 *if* (exercise.AuthorId != User.GetId() && !User.IsInRole("Администратор"))  
 *return* Forbid();  
 *await* uow.Exercises.DeleteAsync(id);  
 *await* uow.CommitAsync(ct);  
  
 logger.LogInformation("Удалено упражнение. Id:{Id}", id);  
  
 *return* RedirectToAction("Index");  
 }  
  
 [HttpPost]  
 *public async* Task<*IActionResult*> Like(*long* id, *string* returnUrl, CancellationToken ct)  
 {  
 *var* userId = User.GetId();  
  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *await* uow.Exercises.AddLikeAsync(id, userId);  
 *await* uow.CommitAsync(ct);  
  
 logger.LogInformation("Пользователь {UserId} лайкнул упражнение {ExerciseId}", userId, id);  
  
 *if* (*string*.*IsNullOrEmpty*(returnUrl))  
 *return* RedirectToAction("Index");  
  
 *return* Redirect(returnUrl);  
 }  
  
 [HttpPost]  
 *public async* Task<*IActionResult*> Unlike(*long* id, *string* returnUrl, CancellationToken ct)  
 {  
 *var* userId = User.GetId();  
  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *await* uow.Exercises.RemoveLikeAsync(id, userId);  
 *await* uow.CommitAsync(ct);  
  
 logger.LogInformation("Пользователь {UserId} убрал лайк с упражнения {ExerciseId}", userId, id);  
  
 *if* (*string*.*IsNullOrEmpty*(returnUrl))  
 *return* RedirectToAction("Index");  
  
 *return* Redirect(returnUrl);  
 }  
  
 [HttpPost]  
 *public async* Task<*IActionResult*> OpenHint(*long* id, *long* exerciseId, CancellationToken ct)  
 {  
 *var* userId = User.GetId();  
  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
   
 *var* exercise = *await* uow.Exercises.GetAsync(exerciseId);  
 *if* (exercise == *null*)  
 *return* NotFound();  
  
 *var* hint = exercise.Hints.FirstOrDefault(h => h.Id == id);  
 *if* (hint == *null*)  
 *return* NotFound();  
   
 *var* user = *await* uow.Users.GetAsync(userId);  
 *if* (user == *null*)  
 *return* NotFound();  
  
 *if* (!User.IsInRole("Администратор"))  
 {  
 *if* (user.Score < hint.Cost)  
 {  
 TempData["ErrorMessage"] =  
 $"У вас недостаточно очков для открытия этой подсказки. Требуется: {hint.Cost}, у вас: {user.Score}";  
 *return* RedirectToAction("View", *new* { id = exerciseId });  
 }  
   
 user.Score -= hint.Cost;  
 *await* uow.Users.UpdateAsync(user);  
 }  
   
 *await* uow.Exercises.OpenHintAsync(id, userId);  
 *await* uow.CommitAsync(ct);  
  
 logger.LogInformation("Пользователь {UserId} открыл подсказку {HintId} за {Cost} очков", userId, id, hint.Cost);  
  
 *return* RedirectToAction("View", *new* { id = exerciseId });  
 }  
}

**FrameworksController.cs**

*using* Coursework.Extensions;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Models;  
*using* Coursework.Models.Entities;  
*using* Microsoft.AspNetCore.Mvc;  
*using* Microsoft.AspNetCore.Authorization;  
  
*namespace* Coursework.Controllers;  
  
[Authorize(Roles = "Администратор")]  
*public class* FrameworksController(*IUnitOfWorkFactory* uowFactory, *ILogger*<HomeController> logger) : Controller  
{  
 *public async* Task<*IActionResult*> Index(CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* frameworks = *await* uow.Frameworks.GetAllWithLanguageAsync();  
   
 ViewBag.Frameworks = frameworks;  
   
 *return* View();  
 }  
   
 *public async* Task<*IActionResult*> Create(CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* languages = *await* uow.Languages.GetAllAsync();  
   
 ViewBag.Languages = languages;  
  
 *return* View();  
 }  
   
 *public async* Task<*IActionResult*> Update(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* framework = *await* uow.Frameworks.GetAsync(id);  
 *if* (framework *is null*)  
 *return* NotFound();  
   
 *var* languages = *await* uow.Languages.GetAllAsync();  
 ViewBag.Languages = languages;  
   
 *return* View(framework);  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Create(Framework framework, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
   
 *if* (!ModelState.IsValid)  
 {  
 *var* languages = *await* uow.Languages.GetAllAsync();  
 ViewBag.Languages = languages;  
 *return* View(framework);  
 }  
   
 *var* id = *await* uow.Frameworks.AddAsync(framework);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Добавлен фреймворк. Id:{Id}, Название:{LanguageName}, Описание:{LanguageDescription}, Автор: {AuthorId}",  
 id, framework.Name, framework.Description, User.GetId());  
   
 *return* RedirectToAction("Index");  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Update(Framework framework, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid) *return* View(framework);  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *await* uow.Frameworks.UpdateAsync(framework);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation(  
 "Обновлен фреймворк. Id:{Id}, Название:{Name}, Описание:{Description}, Id языка программирования:{LanguageId}",   
 framework.Id, framework.Name, framework.Description, framework.LanguageId);  
   
 *return* RedirectToAction("Index");  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Delete(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *await* uow.Frameworks.DeleteAsync(id);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Удален фреймворк. Id:{Id}", id);  
   
 *return* RedirectToAction("Index");  
 }  
}

**HomeController.cs**

*using* System.Diagnostics;  
*using* Microsoft.AspNetCore.Mvc;  
*using* Coursework.Models.DTOs;  
*using* Microsoft.AspNetCore.Authorization;  
  
*namespace* Coursework.Controllers;  
  
[Authorize(Policy = "UserAccess")]  
*public class* HomeController(*ILogger*<HomeController> logger) : Controller  
{  
 [AllowAnonymous]  
 *public IActionResult* Index()  
 {  
 *return* View();  
 }  
  
 [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = *true*)]  
 [AllowAnonymous]  
 *public IActionResult* Error()  
 {  
 *return* View(*new* ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });  
 }  
}

**ProgrammingLanguagesController.cs**

*using* Coursework.Extensions;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Models.Entities;  
*using* Microsoft.AspNetCore.Mvc;  
*using* Microsoft.AspNetCore.Authorization;  
  
*namespace* Coursework.Controllers;  
  
[Authorize(Roles = "Администратор")]  
*public class* ProgrammingLanguagesController(*IUnitOfWorkFactory* uowFactory, *ILogger*<HomeController> logger) : Controller  
{  
 *public async* Task<*IActionResult*> Index(CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* languages = *await* uow.Languages.GetAllAsync();  
   
 ViewBag.Languages = languages;  
   
 *return* View();  
 }  
   
 *public IActionResult* Create()  
 {  
 *return* View();  
 }  
   
 *public async* Task<*IActionResult*> Update(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* language = *await* uow.Languages.GetAsync(id);  
 *if* (language *is null*)  
 *return* NotFound();  
   
 *return* View(language);  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Create(ProgrammingLanguage language, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid) *return* View(language);  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* id = *await* uow.Languages.AddAsync(language);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation(  
 "Добавлен язык программирования. Id:{Id}, Название:{Name}, Описание:{Description}, Автор:{AuthorId}",  
 id, language.Name, language.Description, User.GetId());  
   
 *return* RedirectToAction("Index");  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Update(ProgrammingLanguage language, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid) *return* View(language);  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *await* uow.Languages.UpdateAsync(language);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation(  
 "Обновлен язык программирования. Id:{Id}, Название:{Name}, Описание:{Description}",  
 language.Id, language.Name, language.Description);  
   
 *return* RedirectToAction("Index");  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Delete(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *await* uow.Languages.DeleteAsync(id);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Удален язык программирования. Id:{Id}", id);  
   
 *return* RedirectToAction("Index");  
 }  
}

**SolutionController.cs**

*using* Coursework.Extensions;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Models.Entities;  
*using* Microsoft.AspNetCore.Mvc;  
*using* Microsoft.AspNetCore.Authorization;  
  
*namespace* Coursework.Controllers;  
  
[Authorize(Policy = "UserAccess")]  
*public class* SolutionController(*IUnitOfWorkFactory* uowFactory, *ILogger*<HomeController> logger) : Controller  
{  
 *public async* Task<*IActionResult*> Create(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* exercise = *await* uow.Exercises.GetAsync(id);  
 *if* (exercise *is null*)  
 *return* NotFound();  
  
 ViewBag.Exercise = exercise;  
  
 *return* View();  
 }  
   
 *public async* Task<*IActionResult*> Update(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* solution = *await* uow.Solutions.GetAsync(id);  
 *if* (solution *is null*)  
 *return* NotFound();  
 *if* (solution.AuthorId != User.GetId() && !User.IsInRole("Администратор"))  
 *return* Forbid();  
   
 *return* View(solution);  
 }  
   
 *public async* Task<*IActionResult*> View(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* solution = *await* uow.Solutions.GetAsync(id);  
 *if* (solution *is null*)  
 *return* NotFound();  
   
 *return* View(solution);  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Create(Solution solution, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid) *return* View(solution);  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 solution.AuthorId = User.GetId();  
 *await* uow.Solutions.AddAsync(solution);  
 *var* exercise = *await* uow.Exercises.GetAsync(solution.ExerciseId);  
 *var* user = *await* uow.Users.GetAsync(solution.AuthorId);  
 *if* (user *is not null* && exercise *is not null*)  
 {  
 user.Score += exercise.Score;  
 *await* uow.Users.UpdateAsync(user);  
 }  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation(  
 "Добавлено решение. Id:{Id}, Id упражнения:{ExerciseId}",   
 solution.Id, solution.ExerciseId);  
   
 *return* RedirectToAction("View", "Exercises", *new* { Id = solution.ExerciseId });  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Update(Solution solution, CancellationToken ct)  
 {  
 *if* (!ModelState.IsValid) *return* View(solution);  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
   
 *if* (solution.AuthorId != User.GetId() && !User.IsInRole("Администратор"))  
 *return* Forbid();  
 *await* uow.Solutions.UpdateAsync(solution);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation(  
 "Обновлено решение. Id:{Id}, Id упражнения:{ExerciseId}",   
 solution.Id, solution.ExerciseId);  
   
 *return* RedirectToAction("View", "Exercises", *new* { Id = solution.ExerciseId });  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Delete(*long* id, CancellationToken ct)  
 {  
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* solution = *await* uow.Solutions.GetAsync(id);  
 *if* (solution *is null*)  
 *return* NotFound();  
 *if* (solution.AuthorId != User.GetId() && !User.IsInRole("Администратор"))  
 *return* Forbid();  
   
 *await* uow.Solutions.DeleteAsync(id);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Удалено решение. Id:{Id}", id);  
   
 *return* RedirectToAction("View", "Exercises", *new* { Id = solution.ExerciseId });  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Like(*long* id, CancellationToken ct)  
 {  
 *var* userId = User.GetId();  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* solution = *await* uow.Solutions.GetAsync(id);  
 *if* (solution *is null*)  
 *return* NotFound();  
   
 *await* uow.Solutions.AddLikeAsync(id, userId);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Пользователь {UserId} лайкнул решение {SolutionId}", userId, id);  
   
 *return* RedirectToAction("View", *new* { id });  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Unlike(*long* id, CancellationToken ct)  
 {  
 *var* userId = User.GetId();  
   
 *await using var* uow = *await* uowFactory.CreateAsync(ct);  
 *var* solution = *await* uow.Solutions.GetAsync(id);  
 *if* (solution *is null*)  
 *return* NotFound();  
   
 *await* uow.Solutions.RemoveLikeAsync(id, userId);  
 *await* uow.CommitAsync(ct);  
   
 logger.LogInformation("Пользователь {UserId} убрал лайк с решения {SolutionId}", userId, id);  
   
 *return* RedirectToAction("View", *new* { id });  
 }  
}

**UserController.cs**

*using* Coursework.Extensions;  
*using* Coursework.Interfaces.Database;  
*using* Coursework.Models.Entities;  
*using* Microsoft.AspNetCore.Authorization;  
*using* Microsoft.AspNetCore.Mvc;  
  
*namespace* Coursework.Controllers;  
  
[Authorize(Policy = "UserAccess")]  
*public class* UserController(*IUnitOfWorkFactory* unitOfWorkFactory, *ILogger*<UserController> logger) : Controller  
{  
 *public async* Task<*IActionResult*> View(*long* id, CancellationToken ct)  
 {  
 *await using var* unitOfWork = *await* unitOfWorkFactory.CreateAsync(ct);  
 *var* user = *await* unitOfWork.Users.GetAsync(id);  
  
 *if* (user == *null*)  
 {  
 *return* NotFound();  
 }  
  
 *return* View(user);  
 }  
   
 [Authorize(Roles = "Администратор")]  
 *public async* Task<*IActionResult*> Index(CancellationToken ct)  
 {  
 *await using var* unitOfWork = *await* unitOfWorkFactory.CreateAsync(ct);  
 *var* users = *await* unitOfWork.Users.GetAllAsync();  
   
 ViewBag.Users = users;  
   
 *return* View();  
 }  
   
 *public async* Task<*IActionResult*> Update(*long* id, CancellationToken ct)  
 {  
 *var* currentUserId = User.GetId();  
 *var* isAdmin = User.IsInRole("Администратор");  
   
 *if* (currentUserId != id && !isAdmin)  
 {  
 *return* Forbid();  
 }  
   
 *await using var* unitOfWork = *await* unitOfWorkFactory.CreateAsync(ct);  
 *var* user = *await* unitOfWork.Users.GetAsync(id);  
  
 *if* (user == *null*)  
 {  
 *return* NotFound();  
 }  
  
 *return* View(user);  
 }  
   
 [HttpPost]  
 *public async* Task<*IActionResult*> Update(User model, CancellationToken ct)  
 {  
 *var* currentUserId = User.GetId();  
 *var* isAdmin = User.IsInRole("Администратор");  
   
 *if* (currentUserId != model.Id && !isAdmin)  
 {  
 *return* Forbid();  
 }  
   
 *if* (*string*.*IsNullOrWhiteSpace*(model.Name))  
 {  
 ModelState.AddModelError("Name", "Имя пользователя не может быть пустым");  
 *return* View("Update", model);  
 }  
   
 *await using var* unitOfWork = *await* unitOfWorkFactory.CreateAsync(ct);  
 *var* user = *await* unitOfWork.Users.GetAsync(model.Id);  
  
 *if* (user == *null*)  
 {  
 *return* NotFound();  
 }  
   
 user.Name = model.Name;  
 user.About = model.About ?? *string*.Empty;  
   
 *if* (isAdmin)  
 {  
 user.RoleId = model.RoleId;  
 user.Score = model.Score;  
 }  
   
 *await* unitOfWork.Users.UpdateAsync(user);  
 *await* unitOfWork.CommitAsync(ct);  
   
 *if*(!isAdmin)   
 logger.LogInformation("Пользователь {UserId} обновил профиль", model.Id);  
 *else* logger.LogInformation("Администратор {AdminId} изменил пользователя {UserId}",   
 currentUserId, model.Id);  
   
 *return* RedirectToAction("View", *new* { model.Id });  
 }  
   
 [HttpPost]  
 [Authorize(Roles = "Администратор")]  
 *public async* Task<*IActionResult*> Delete(*long* id, CancellationToken ct)  
 {  
 *await using var* unitOfWork = *await* unitOfWorkFactory.CreateAsync(ct);  
 *var* user = *await* unitOfWork.Users.GetAsync(id);  
   
 *if* (user == *null*)  
 {  
 *return* NotFound();  
 }  
   
 *if* (user.Id == User.GetId())  
 {  
 ModelState.AddModelError("", "Вы не можете удалить свой собственный аккаунт");  
 *return* RedirectToAction("Index");  
 }  
   
 *await* unitOfWork.Users.DeleteAsync(id);  
 *await* unitOfWork.CommitAsync(ct);  
   
 logger.LogInformation("Администратор {AdminId} удалил пользователя {UserId}", User.GetId(), id);  
   
 *return* RedirectToAction("Index");  
 }  
}