

СОДЕРЖАНИЕ

| | |
|---|-----|
| Введение..... | 5 |
| 1 Предметная область | 6 |
| 1.1 Описание предметной области | 6 |
| 1.2 Обзор аналогов..... | 9 |
| 1.3 Выводы и постановка задачи | 14 |
| 2 Инструменты разработки..... | 15 |
| 2.1 Обзор языков технологий и языков программирования | 15 |
| 2.2 Обзор среды разработки..... | 23 |
| 3 Разработка приложения | 28 |
| 3.1 Разработка серверной части..... | 28 |
| 3.2 Разработка панели администрирования | 45 |
| 3.3 Разработка приложения для ПК..... | 47 |
| 4 Руководство пользователя | 49 |
| 4.1 Административная панель | 49 |
| 4.2 Приложение для ПК | 58 |
| 5 Технико-экономическое обоснование разработки web-приложения для профилактики профессиональных заболеваний | 67 |
| 5.1 Краткая характеристика разрабатываемого web-приложения | 67 |
| 5.2 Расчет себестоимости и разрабатываемого web-приложения..... | 67 |
| 6 Охрана труда при эксплуатации и обслуживании ПЭВМ и ВДТ, офисной техники | 74 |
| 6.1 Общие требования охраны труда | 74 |
| 6.2 Требования охраны труда перед началом работы | 75 |
| 6.3 Требования охраны труда во время работы | 75 |
| 6.4 Требования охраны труда в аварийных ситуациях | 78 |
| 6.5 Требования охраны труда по окончании работы..... | 79 |
| Заключение..... | 80 |
| Список использованных источников..... | 81 |
| Приложение А. UML диаграмма базы данных..... | 82 |
| Приложение Б. UML диаграмма классов дипломного проекта..... | 83 |
| Приложение В. UML диаграмма вариантов использования административной панели..... | 84 |
| Приложение Г. UML диаграмма деятельности добавления упражнения..... | 85 |
| Приложение Д. UML диаграмма вариантов использования приложения для ПК..... | 86 |
| Приложение Е. UML диаграмма деятельности профилактической тренировки..... | 87 |
| Приложение Ж. Исходный код программы..... | 88 |
| Приложение И. Презентация..... | 123 |

ВВЕДЕНИЕ

Профилактика заболеваний является одной из важнейших задач современного здравоохранения, она заложена в ряде государственных программ и системе ОМС. К тому же даже привычные гигиенические навыки и правильный образ жизни способны оказать профилактическое действие.

Человек с самого раннего возраста может быть подвержен различным заболеваниям. Они влияют на продолжительность жизни и ее качество, снижают трудоспособность и даже становятся причиной инвалидности и социально-бытовой беспомощности. Некоторые болезни характеризуются высокой смертностью, другие повышают риск рождения потомства с различными отклонениями, третьи делают заболевшего человека опасным для окружающих и могут привести к эпидемиям. Во многих случаях профилактические меры способны предупредить развитие заболеваний или сделать их прогноз более благоприятным.

Целью данной дипломной работы является разработка web-приложения для профилактики профессиональных заболеваний.

Разработанное приложение должно будет помочь людям предотвращать появление у заболеваний и более того, увеличить качество жизни людей.

При разборе аналогов были выявлены такие недостатки, как низкий уровень персонализации, узкая специализация, навязчивый интерфейс.

1 ПРЕДМЕТНАЯ ОБЛАСТЬ

1.1 Описание предметной области

Целью данного дипломного проекта является разработка автоматизированной системы создания web-приложения для профилактики профессиональных заболеваний.

Прикладная программа или приложение – программа, предназначенная для выполнения определённых задач и рассчитанная на непосредственное взаимодействие с пользователем. В большинстве операционных систем прикладные программы не могут обращаться к ресурсам компьютера напрямую, а взаимодействуют с оборудованием и другими программами посредством операционной системы. Также на простом языке – вспомогательные программы.

К прикладному программному обеспечению относятся компьютерные программы, написанные для пользователей или самими пользователями для задания компьютеру конкретной работы. Программы обработки заказов или создания списков рассылки – пример прикладного программного обеспечения. Программистов, которые пишут прикладное программное обеспечение, называют прикладными программистами.

Web-приложение – клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает – веб-сервер. Логика web-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому web-приложения являются межплатформенными службами.

Web-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиент-сервер – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер – это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной

почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Само web-приложение может выступать в качестве клиента других служб, например, базы данных или другого web-приложения, расположенного на другом сервере. Ярким примером web-приложения является система управления содержимым статей Википедии: множество её участников могут принимать участие в создании сетевой энциклопедии, используя для этого браузеры своих операционных систем (будь то Microsoft Windows, GNU/Linux или любая другая операционная система) и не загружая дополнительных исполняемых модулей для работы с базой данных статей.

В настоящее время набирает популярность новый подход к разработке web-приложений, называемый Ajax. При использовании Ajax страницы web-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Также в последнее время набирает большую популярность технология WebSocket, которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания web-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль.

База данных (БД) – это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для динамических сайтов со значительными объемами данных – часто это интернет-магазины, порталы, корпоративные сайты. Такие

сайты обычно разработаны с помощью серверного языка программирования (как пример, PHP) или на основе CMS (как пример, WordPress), и не имеют готовых страничек с данными по аналогии с HTML-сайтами. Странички динамических сайтов формируются «на лету» в результате взаимодействия скриптов и баз данных после соответствующего запроса клиента к веб-серверу.

Система управления базами данных (СУБД) – это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Болезнь, заболевание (лат. morbus) – это возникающие в ответ на действие патогенных факторов нарушения нормальной жизнедеятельности, работоспособности, социально полезной деятельности, продолжительности жизни организма и его способности адаптироваться к постоянно изменяющимся условиям внешней и внутренней сред при одновременной активизации защитно-компенсаторно-приспособительных реакций и механизмов.

Болезнь – это состояние организма, выраженное в нарушении его нормальной жизнедеятельности, продолжительности жизни, и его способности поддерживать свой гомеостаз. Является следствием ограниченных энергетических и функциональных возможностей живой системы в противопоставлении патогенным факторам

Профессиональные заболевания – заболевания, спровоцированные либо усугубляемые специфическими условиями определенной профессиональной деятельности. К наиболее известным относятся силикозы и силикатозы (цементоз, асбестоз) (поражение легких каменной пылью), антракоз ("болезнь шахтеров", вызывается угольной пылью) и многие другие разновидности пневмокониозов, литейная лихорадка (поражение легких парами металлов), вибрационная болезнь (поражение опорно-двигательного аппарата длительным воздействием вибрации), синдром запястного канала (разновидность туннельного синдрома), нарушения слуха воздействием промышленного шума.

Многие профессиональные заболевания излечиваются улучшением условий труда либо сменой профессии.

К профессиональным заболеваниям не относятся производственные травмы.

Профилактика профессиональных и профессионально обусловленных заболеваний – система мер медицинского (санитарно-эпидемиологического, санитарно-гигиенического, лечебно-профилактического и т. д.) и немедицинского (государственного, общественного, экономического, правового, экологического и др.) характера, направленных на предупреждение несчастных случаев на производстве, снижение риска развития отклонений

в состоянии здоровья работников, предотвращение или замедление прогрессирования заболеваний, уменьшение неблагоприятных последствий. Развитие многих профессиональных заболеваний и профессионально обусловленных заболеваний зависит от комплексного взаимодействия повреждающих факторов и от качества трудовой жизни. Все работники должны приобретать гигиенические знания и навыки, выполнять нормы и требования, обеспечивающие безопасность труда.

1.2 Обзор аналогов

1.2.1 Web-приложение, которое размещено по адресу <http://blimb.su>. Предоставляет полноценную программу для разминки глаз.

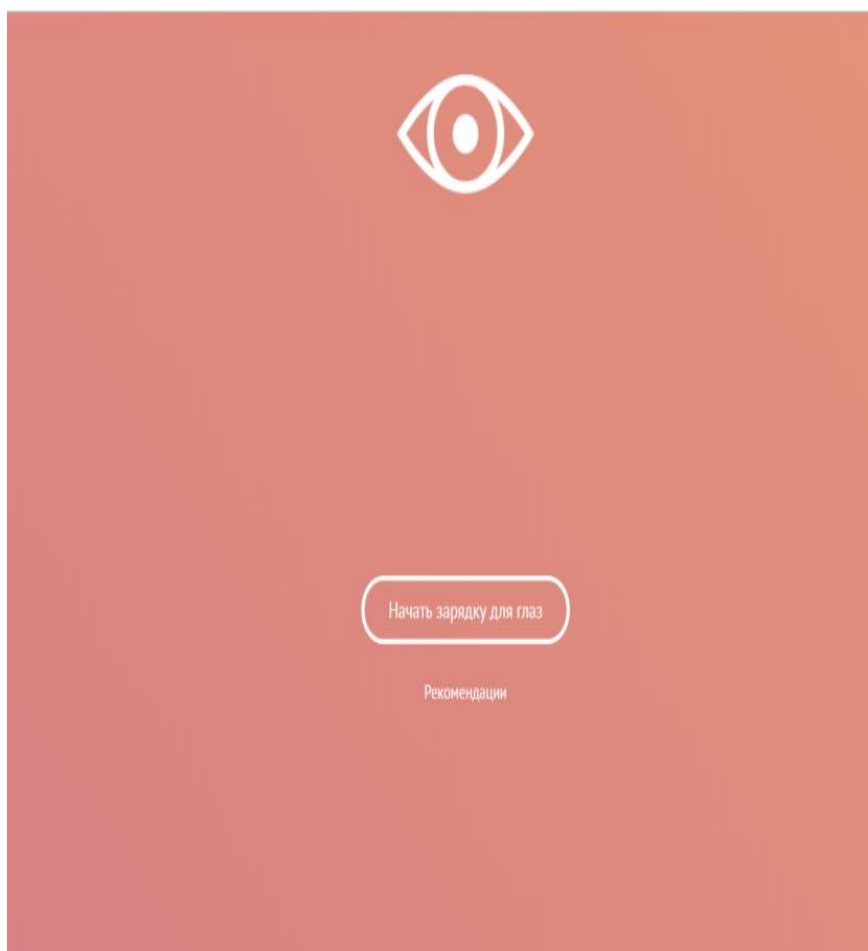


Рисунок 1.1 – Страница приветствия пользователя

После того, как была нажата кнопка «Начать зарядку для глаз», пользователь увидит страницу, на которой отображены рекомендации к пользованию (рисунок 1.2).

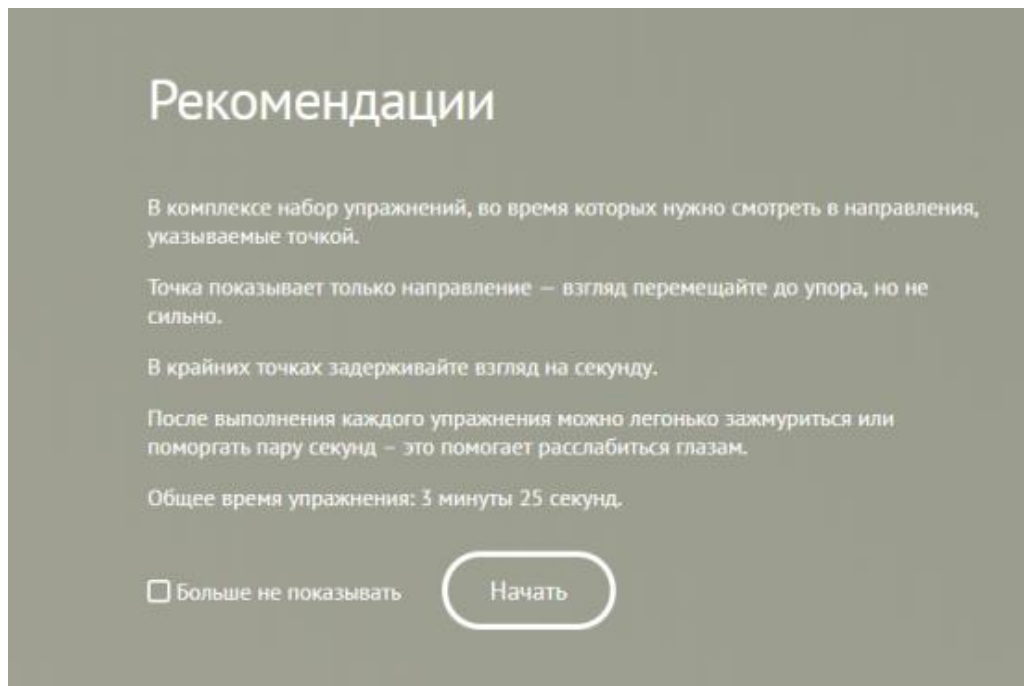


Рисунок 1.2 – Страница с рекомендациями

Далее пользователь должен последовательно выполнять движения глазами в соответствии с инструкциями на экране (рисунок 1.3).

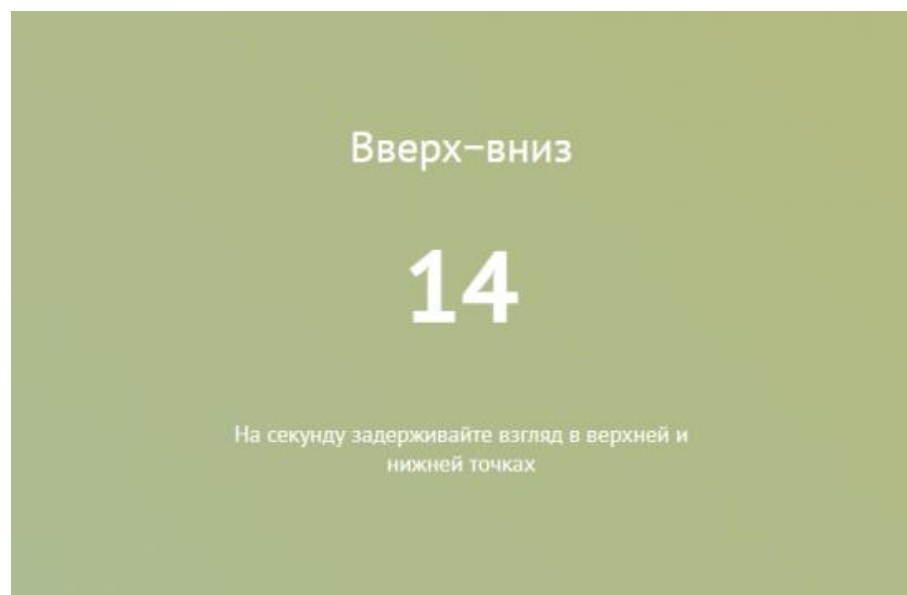


Рисунок 1.3 – Пример упражнения

Основные достоинства приложения:

– Хороший дизайн;

- Широкий спектр упражнений для разминки глаз;
- Невысокая общая длительность разминки.

Основные недостатки приложения:

- Имеет очень узкую специализацию;

1.2.2 Workrave – ещё одна удобная бесплатная программа для перерывов на русском языке. Предлагая отдохнуть, Workwave сначала действует ненавязчиво, потом – настойчиво требует сделать перерыв. В итоге программа для зрения Workrave вообще блокирует клавиатуру с мышью на заданное в настройках время.

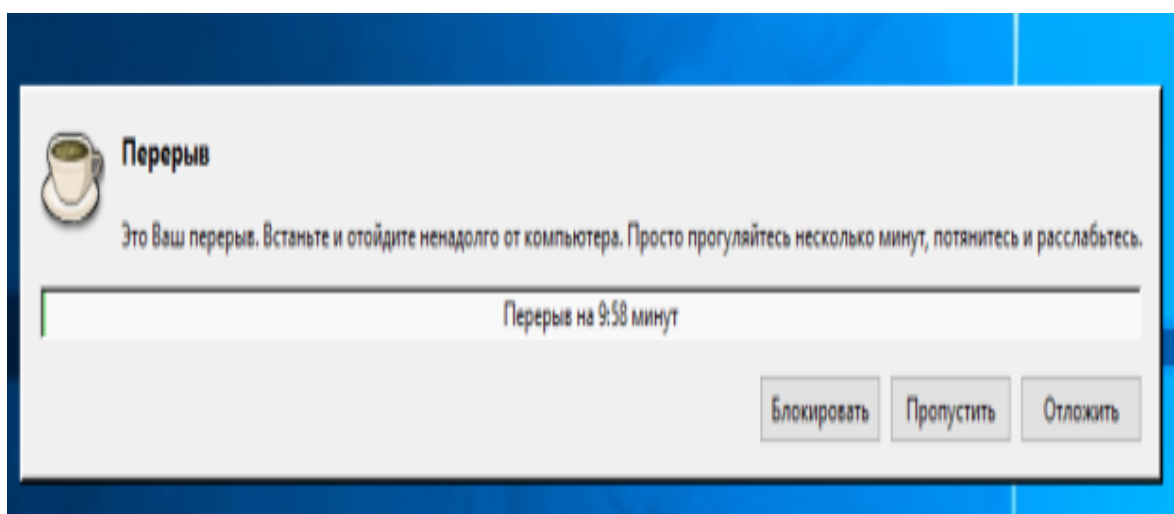


Рисунок 1.4 – Окно блокировки экрана на время перерыва

Помимо функций блокировки, программа предлагает специальные упражнения для отдыха от работы за компьютером.

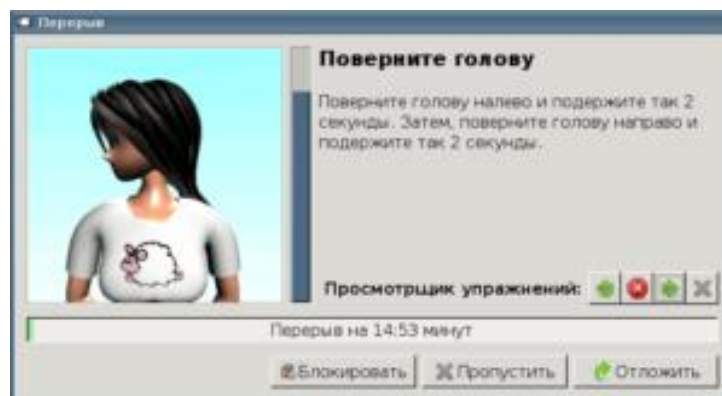


Рисунок 1.5 – Специальные упражнения для отдыха от работы за компьютером.

Основные достоинства приложения:

- Хороший функционал;
- Отличный набор упражнений и их комбинации;
- Возможность запуска при старте операционной системы;
- Легкая установка.

Основные недостатки приложения:

- Устаревший интерфейс;
- Плохие возможности настройки;
- Высокая навязчивость.

1.2.3 ChronoControl. Помогает следить за режимом работы пользователя и заставляет пользователя своевременно делать перерывы, необходимые для здоровья. При этом ChronoControl поступает наиболее радикальным и эффективным методом - на время перерыва программа блокирует монитор, клавиатуру и мышь.

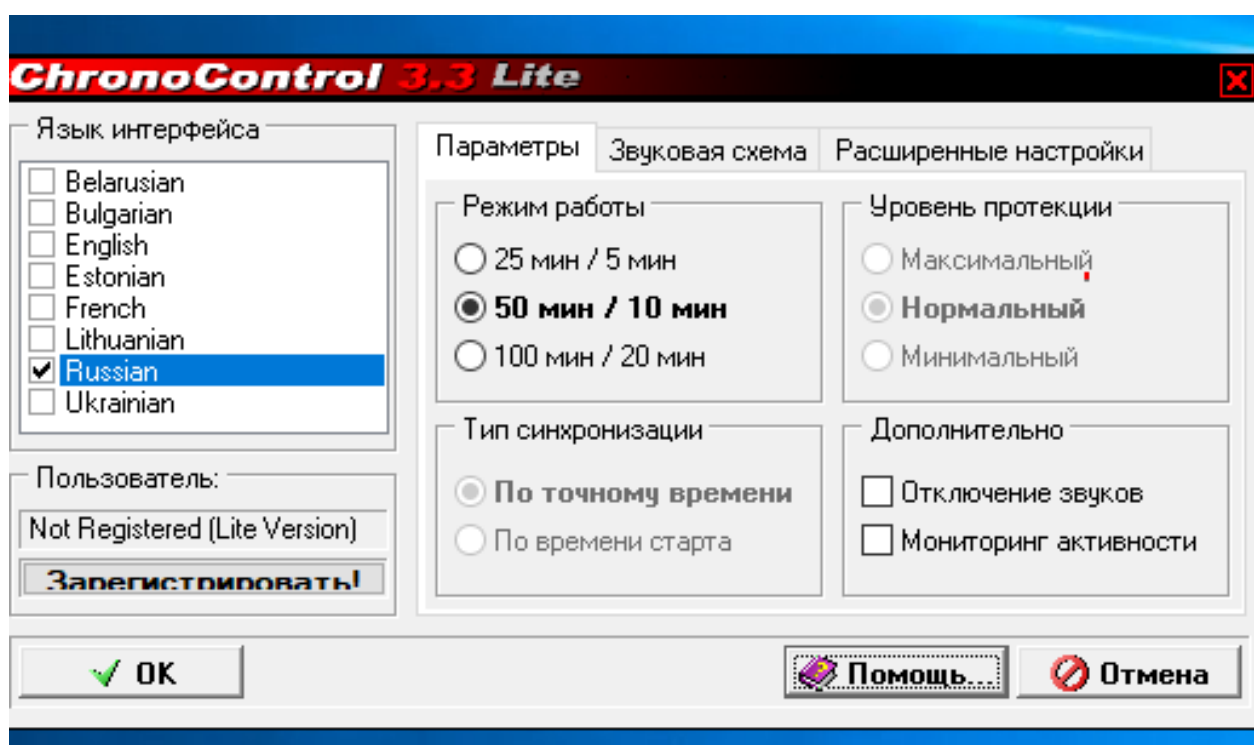


Рисунок 1.6 – Страница настроек ChronoControl

После того, как была нажата кнопка «Начать зарядку для глаз», пользователь увидит страницу, на которой отображены рекомендации к пользованию.

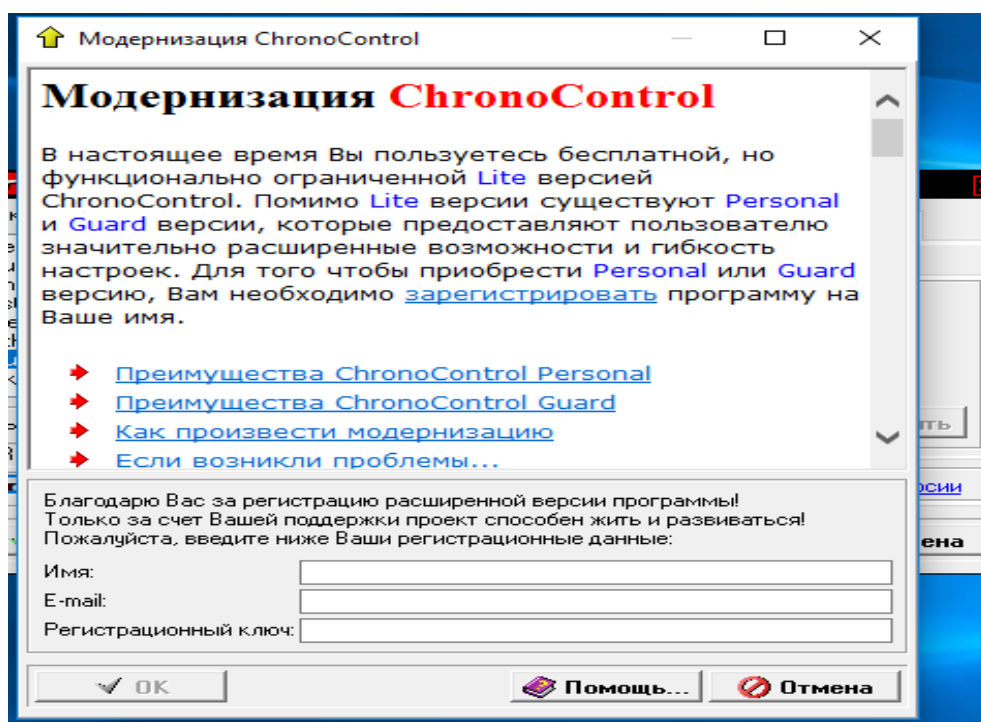


Рисунок 1.7 – Страница расширенных настроек

Есть возможность установить пароль. Большинство дополнительных возможностей недоступно в бесплатной версии.

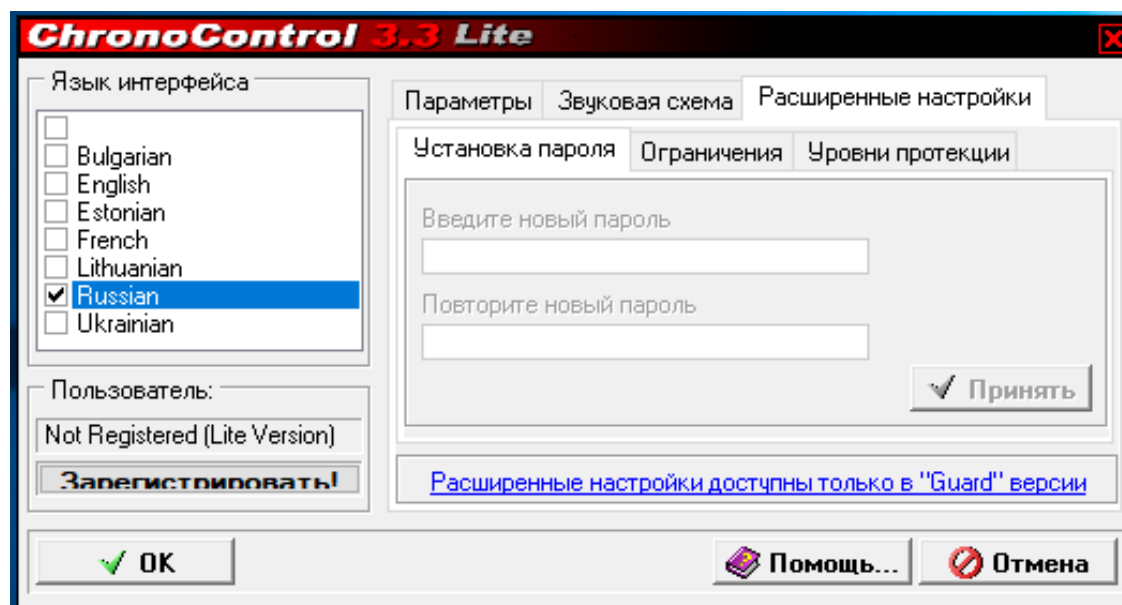


Рисунок 1.8 – Страница установки пароля

Основные достоинства приложения:

- Хороший функционал;
- Широкий спектр настроек.

Основные недостатки приложения:

- Радикальные меры ухудшают пользовательский опыт;
- Большинство возможностей заблокировано в бесплатной версии;
- Устаревший дизайн;
- Отсутствие программ упражнений.

1.3 Выводы и постановка задачи

В данном дипломном проекте необходимо разработать web-приложение для профилактики профессиональных заболеваний. Пользователь должен иметь возможность осуществлять поиск по упражнениям. Специально для пользователя с учетом его профессии, индивидуальных показателей и критерий должна подбираться оптимальная программа для профилактики заболеваний.

Необходимо написать веб-сервер, который будет обрабатывать запросы пользователя. Необходимо создать веб-клиент, в котором администратор сможет управлять работой программы и настройкой приложения.

Из исследований первого раздела можно сделать вывод, что оптимальным будет наличие так же клиента для настольных ПК.

2 ИНСТРУМЕНТЫ РАЗРАБОТКИ

2.1 Обзор языков технологий и языков программирования

2.1.1 C#. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла в 7 марта 2017 года вместе с Visual Studio 2017.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. C# является объектно-ориентированным и в этом плане много перенял у Java и C++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (WPF, ASP.NET). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общезыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А благодаря проекту Mono можно создавать приложения, которые будут работать и на других ОС семейства Linux, в том числе на мобильных платформах Android и iOS.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от C++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

Нередко приложение, созданное на C#, называют управляемым кодом (managed code). Что это значит? А это значит, что данное приложение создано на основе платформы .NET и поэтому управляется общеязыковой средой CLR, которая загружает приложение и при необходимости очищает память. Но есть также приложения, например, созданные на языке C++, которые компилируются не в общий язык CIL, как C# или VB.NET, а в обычный машинный код. В этом случае .NET не управляет приложением.

В то же время платформа .NET предоставляет возможности для взаимодействия с неуправляемым кодом. Мы наряду со стандартными классами библиотеки .NET можем также использовать сборки COM.

Код на C# компилируется в приложения или сборки с расширениями exe или dll на языке CIL. Далее при запуске на выполнение подобного приложения происходит JIT-компиляция (Just-In-Time) в машинный код, который затем выполняется. При этом, поскольку наше приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет обращение. Если мы обратимся к другой части кода, то она будет

скомпилирована из CIL в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.

Еще одним весом преимуществом платформы .NET является наличие LINQ. LINQ (Language-Integrated Query) представляет простой и удобный язык запросов к источнику данных. В качестве источника данных может выступать объект, реализующий интерфейс IEnumerable (например, стандартные коллекции, массивы), набор данных DataSet, документ XML. Но вне зависимости от типа источника LINQ позволяет применить ко всем один и тот же подход для выборки данных.

Существует несколько разновидностей LINQ:

- LINQ to Objects: применяется для работы с массивами и коллекциями;
- LINQ to Entities: используется при обращении к базам данных через технологию Entity Framework;
- LINQ to Sql: технология доступа к данным в MS SQL Server;
- LINQ to XML: применяется при работе с файлами XML;
- LINQ to DataSet: применяется при работе с объектом DataSet;
- Parallel LINQ (PLINQ): используется для выполнения параллельной запросов.

Язык C# был выбран для реализации данной дипломной работы, т.к. он является основным языком платформы .NET и технологии ASP.NET в частности. Так же весомую роль играет наличие такой технологии, как LINQ.

2.1.2 ASP.NET – технология компании Microsoft для создания веб-сайтов, веб-сервисов и приложений. Благодаря надежности, гибкости и безопасности активно используется крупными компаниями.

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и web-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

Общую схему взаимодействия этих компонентов можно представить следующим образом:

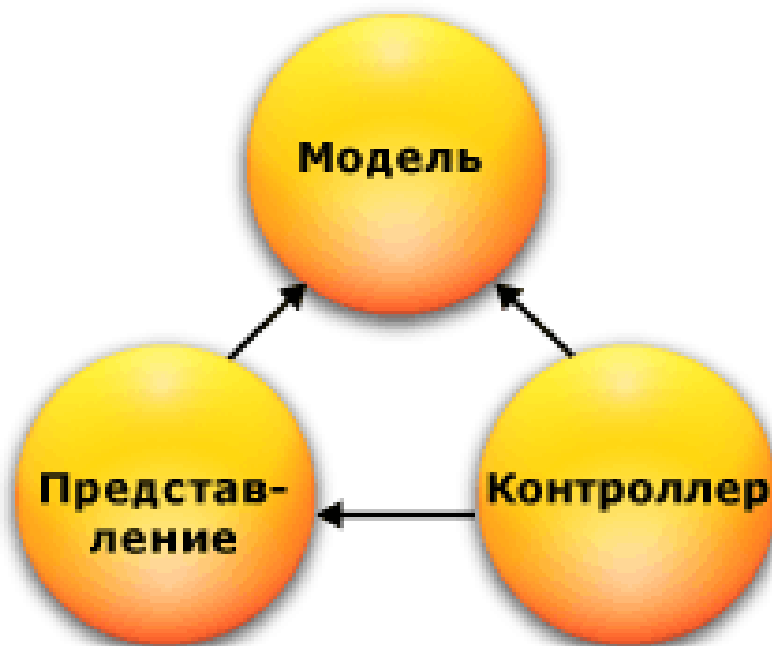


Рисунок 2.1 – Модель MVC

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга.

Благодаря этому реализуется концепция разделение ответственности, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если нам, допустим, важна визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на бэкэнде и тестировать контроллер.

Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

В MVC 5 используется система ASP.NET Identity, которая использует компоненты OWIN и Katana. Для создания адаптивного и расширяемого интерфейса в MVC 5 используется css-фреймворк Bootstrap, но можно использовать и любой другой. Добавлены

фильтры аутентификации, а также появилась функциональность переопределения фильтров. В MVC 5 также добавлены атрибуты маршрутизации.

2.1.3 JavaScript – это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript невероятно универсален. JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

Программные интерфейсы приложения (API), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов.

Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook.

Есть возможность применять к HTML сторонние фреймворки и библиотеки, что позволит ускорить создание сайтов и приложений.

2.1.4 Vue – это прогрессивный фреймворк для создания пользовательских интерфейсов. Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками.

Vue был выбран, как фреймворк для разработки интерфейса, потому что он очень легковесный и предоставляет отличные API.

2.1.5 Electron (ранее известен как `atom shell`) – фреймворк, разработанный GitHub. Позволяет разрабатывать нативные графические приложения для настольных операционных систем с помощью веб-технологий. Фреймворк включает в себя Node.js, для работы с back-end, и библиотеку рендеринга из Chromium.

Electron был выбран как платформа для разработки приложения для настольных ПК, т.к сформировался определенный набор технологий основанный на веб-технологиях. Это позволит использовать одни и те же компоненты и структуры в разработке системы.

2.1.6 HTML(HyperTextMarkupLanguage) – стандартный язык разметки гипертекстовых страниц. Есть и другие языки разметки гипертекста, но большая часть страниц сайтов Интернета размечена именно на языке HTML. Такие страницы успешно интерпретируются браузерами, которые отображают их на экранах различных электронных устройств в удобном для человека виде.

HTML является теговым языком разметки гипертекста: чтобы превратить текст в гипертекст, используют разделители (дескрипторы), для краткости названные тегами. Вот пример тега: `` – этот открывающий тег обеспечивает вывод текста жирным шрифтом до тех пор, пока не встретится закрывающий тег ``.

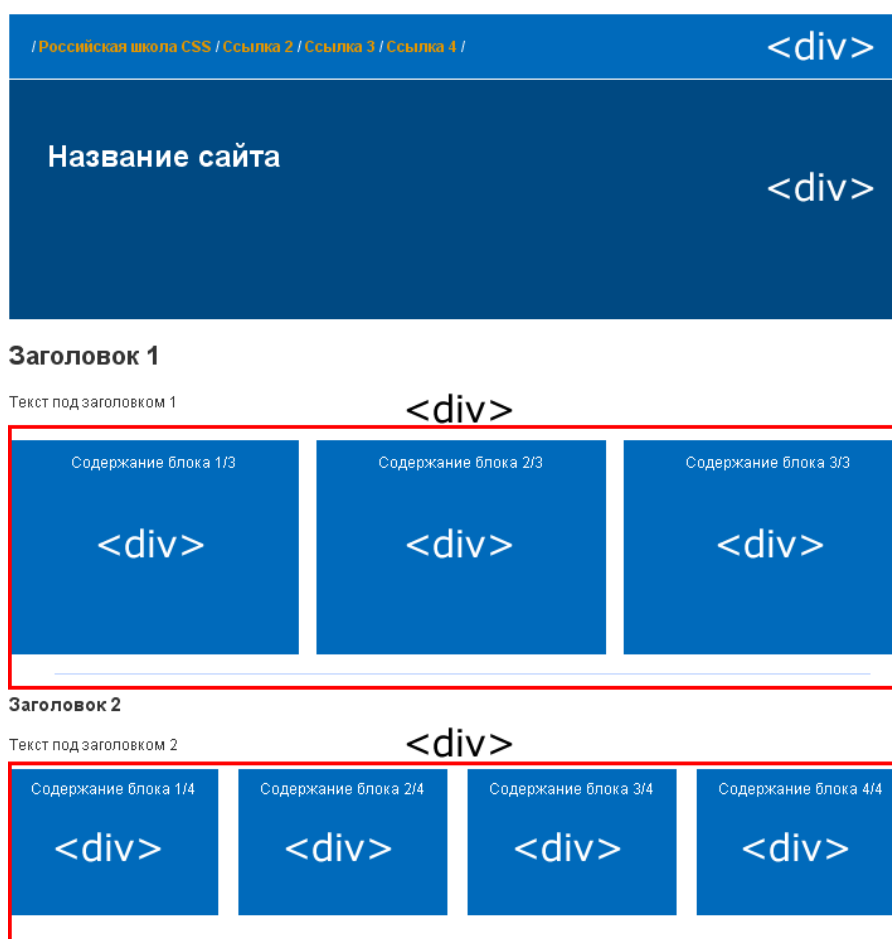


Рисунок 2.2 – Макет описанный на языке HTML

2.1.7 CSS(Cascading Style Sheets) – каскадные таблицы стилей. Стиль – набор параметров, задающий внешнее представление объекта.

Например, пусть мы хотим, чтобы все заголовки первого уровня (теги <h1>) на одной странице имели красный цвет, размер - 24 и были написаны курсивом, а на другой странице были бы синего цвета, размера - 12. Наш заголовок - это объект, а цвет, размер и начертание - это параметры. Каждый элемент на странице может иметь свой стиль (параграфы, заголовки, линии, текст...). Набор стилей всех элементов называют таблицей стилей. Если для одного элемента задано несколько стилей (как в примере с заголовками), то применяется каскадирование, которое определяет приоритет того или иного стиля.

Преимущества CSS:

- CSS позволяет значительно сократить размер кода и сделать его читабельным;
 - CSS позволяет задавать такие параметры, которые нельзя задать только языком HTML. Например, убрать подчеркивание у ссылок;
 - CSS позволяет легко изменять внешний вид страниц;
- С CSS связана так называемая блочная верстка сайта.

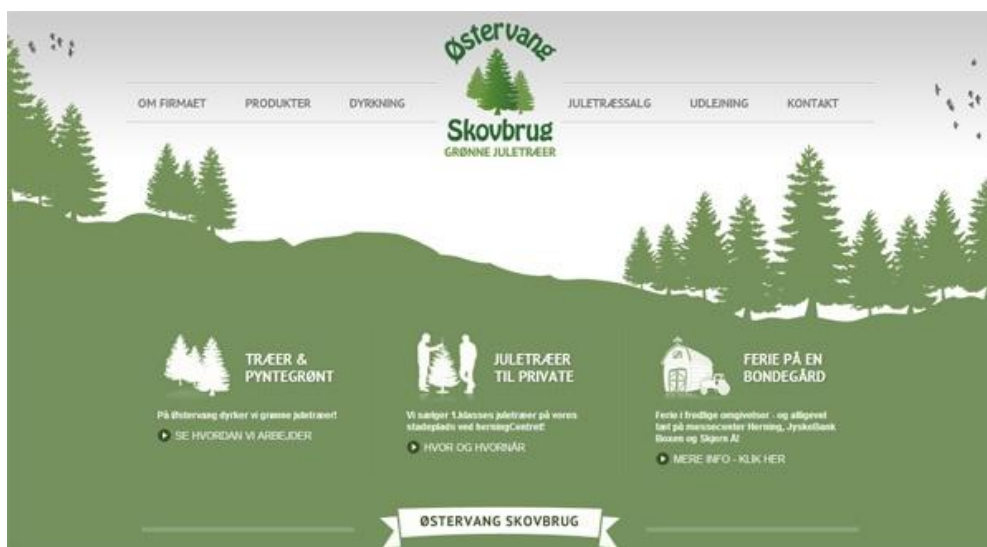


Рисунок 2.3 – Пример верстки, сделанной с помощью CSS

2.1.8 MSSQL. SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году. SQL Server долгое время

был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

- Производительность. SQL Server работает очень быстро;
- Надежность и безопасность. SQL Server предоставляет шифрование данных;
- Простота. С данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных.

База данных представляет хранилище данных, организованных определенным способом.

Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно.

Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language).

Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API.

СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

2.2 Обзор среды разработки

Интегрированная среда разработки программного обеспечения – система программных средств, используемая программистами для разработки программного обеспечения.

Обычно среда разработки включает в себя текстовый редактор, компилятор и/или интерпретатор, средства автоматизации сборки и отладчик. Иногда также содержит средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов – для использования при объектно-ориентированной разработке ПО. Ранее были широко распространены среды разработки только для одного языка, например, Visual Basic, но сейчас популярны такие среды разработки, которые поддерживают мультиязычность, такие как SublimeText, Visual Studio Code, WebStorm, IDEA, Microsoft Visual Studio.

Примеры сред разработки :

- Eclipse;
- Sun Studio;
- Turbo Pascal;
- Borland C++;
- GNU toolchain;
- DrPython;
- Borland Delphi;
- Dev-C++;
- KDevelop;
- XCode;
- Visual Studio Code;
- Microsoft Visual Studio.

Частный случай ИРС – среды визуальной разработки, которые включают в себя возможность визуального редактирования интерфейса программы.

В ходе разработки данного дипломного проекта применялись следующие среды разработки:

- Microsoft Visual Studio 2017 Community Edition;
- Visual Studio Code;
- Microsoft SQL Server Manager Studio.

2.2.1 Microsoft Visual Studio.

Интерактивная интегрированная среда разработки Visual Studio – это оригинальная среда запуска, которая позволяет просматривать и изменять практически любой код, а также отлаживать, выполнять сборку и публиковать приложения для устройств с Android, iOS, Windows, а также в Интернете и облаке. Доступны версии как для компьютеров Mac, так и для компьютеров с Windows.

Visual Studio предлагает набор инструментов, позволяющих с легкостью создавать облачные приложения на базе Microsoft Azure. Она упрощает настройку, сборку, отладку, упаковку и развертывание приложений и служб в Microsoft Azure прямо из IDE. Чтобы получить инструменты Azure для .NET, при установке Visual Studio необходимо рабочую нагрузку разработки для Azure.

Есть возможность использовать службы Azure для приложений с помощью подключенных служб, таких как:

- Мобильные службы Azure;
- Хранилище Azure.

HockeyApp позволяет распространять бета-версии, собирать динамические отчеты о сбоях и получать отзывы от реальных пользователей.

Application Insights позволяет выявлять и диагностировать проблемы с качеством в приложениях и веб-службах. Служба Application Insights также помогает понять, что именно пользователи делают с разработанным приложением. Это помогает оптимизировать работу с ним.

С помощью среды разработки Visual Studio есть возможность создавать web-приложения с помощью технологий ASP.NET, Node.js, Python, JavaScript и TypeScript. Visual Studio распознает такие платформы для создания web-приложений, как Angular, jQuery, Express и другие. Платформы ASP.NET Core и .NET Core поддерживаются на компьютерах с Windows и Linux, а также на компьютерах Mac. ASP.NET Core представляет собой существенное обновление для MVC, WebAPI и SignalR, которое работает на платформах Windows, Mac и Linux.

Платформа ASP.NET Core была разработана с нуля и предоставляет компактный и изменяемый стек .NET для разработки современных облачных web-приложений и служб.

Когда приложение будет готово к развертыванию для пользователей или клиентов, доступна возможность выполнить его с помощью инструментов Visual Studio, будь это развертывание в Microsoft Store, на сайте SharePoint или с помощью технологии InstallShield или установщика Windows. Все эти возможности доступны в среде IDE.

При написании кода требуется запустить его и проверить на ошибки и производительность. Современная система отладки Visual Studio позволяет выполнять отладку кода в локальном проекте, на удаленном устройстве или в эмуляторе, например в эмуляторе для устройств Android.

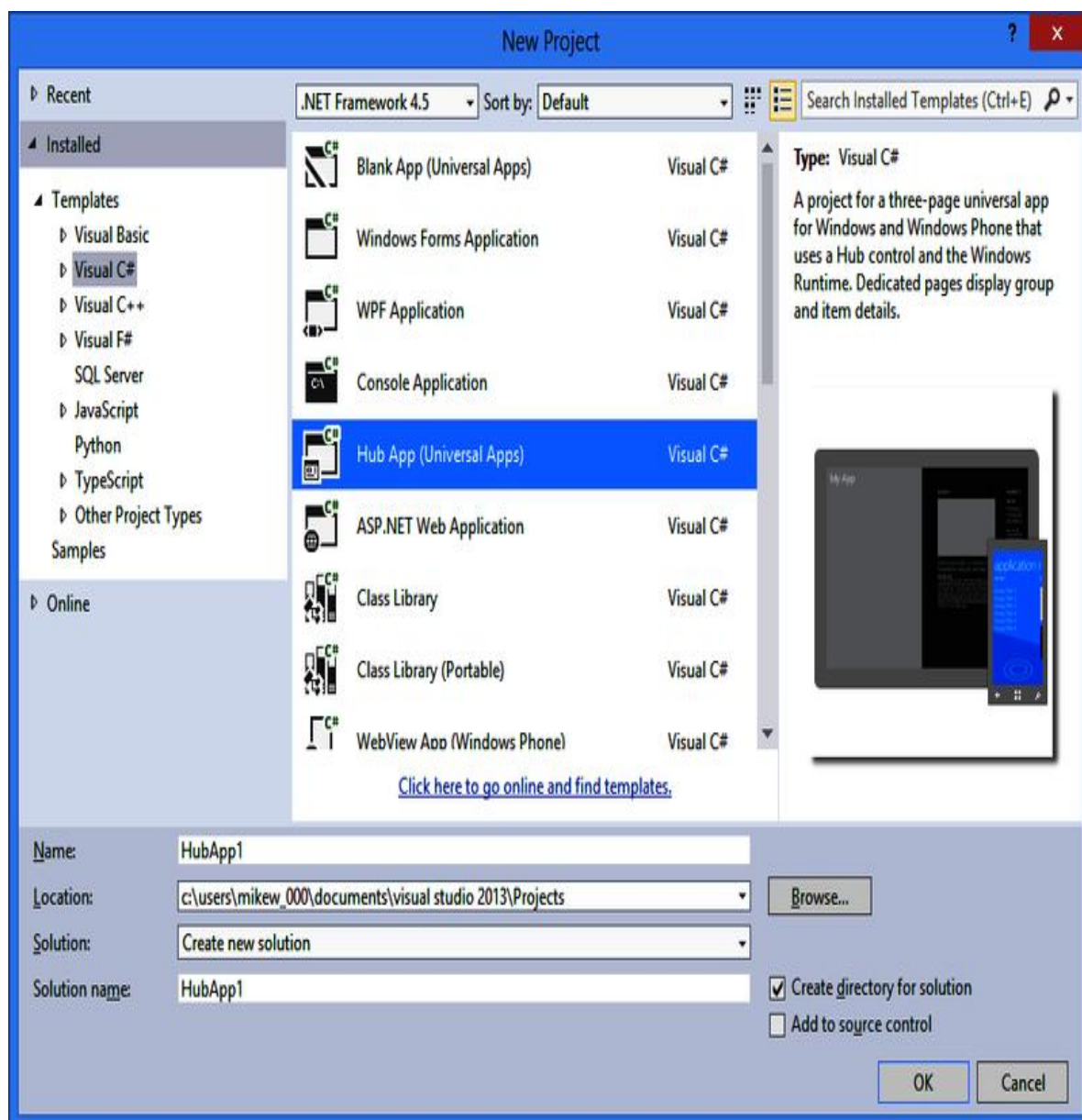


Рисунок 2.4 – Среда разработки Microsoft Visual Studio

Можно просматривать код с шагом в один оператор, проверяя значения переменных. Можно задать точки останова, которые срабатывают только при выполнении указанного условия. Есть возможность отслеживать значения переменных во время выполнения кода, а также многое другое. Все это можно контролировать в самом редакторе кода, не покидая окно с кодом.

Для выполнения проверки Visual Studio предлагает такие возможности, как модульное тестирование, IntelliTest, тестирование производительности, нагрузочное тестирование и прочее.

Данная среда была выбрана в связи с тем, что основным языком для данного дипломного проекта был выбран C#, а основной технологией ASP.NET MVC 5.

2.2.2 Visual Studio Code. Visual Studio Code – редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «легкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Visual Studio Code основан на Electron – фреймворк, позволяющий с использованием Node.js разрабатывать настольные приложения, которые работают на движке Blink. Несмотря на то, что редактор основан на Electron, он не использует редактор Atom. Вместо него реализуется веб-редактор Monaco, разработанный для Visual Studio Online.

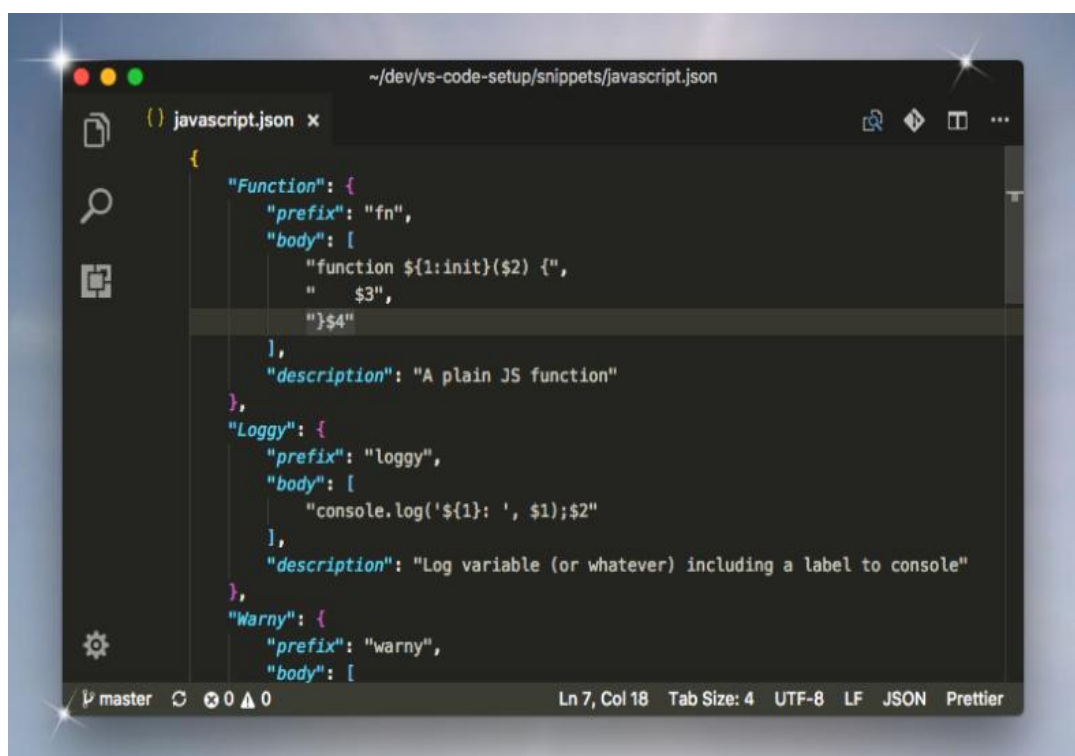


Рисунок 2.5 – Среда разработки Visual Studio Code

Visual Studio Code – это редактор исходного кода. Он поддерживает ряд языков программирования, подсветку синтаксиса, IntelliSense, рефакторинг, отладку, навигацию по коду, поддержку Git и другие возможности.

Многие возможности Visual Studio Code не доступны через графический интерфейс, зачастую они используются через палитру команд или JSON файлы (например, пользовательские настройки). Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

Visual Studio так же позволяет заменять кодовую страницу при сохранении документа, символы перевода строки и язык программирования текущего документа.

Данная среда разработки была выбрана второй основной средой, т.к. имеет широкий инструментарий и базу расширений для работы с одним из основных языков программирования для данного дипломного проекта – JavaScript. Так же она имеет готовую поддержку и хорошую интеграцию со сборщиком приложений, написанных на JavaScript, Html и CSS – webpack.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Разработка серверной части

Разработка серверной части предусматривает несколько этапов:

1. Разработка базы данных;
2. Разработка архитектуры серверного приложения;
3. Разработка внешнего программного интерфейса приложения.

Серверная часть должна предоставлять следующие возможности:

- внешний пользовательский интерфейс на основе RESTful архитектуры;
- интерфейсы для работы с базой данных;
- системы аутентификации и авторизации;
- CRUD для сущностей модели.

3.1.1 Разработка базы данных. В качестве базы данных была выбрана SQL база данных от компании Microsoft – MSSQL. Диаграмма базы данных представлена в Приложении А.

Таблица 3.1 – Сводка о таблицах в базе данных

| Название | Схема | Описание | К-во столбцов |
|--------------------|-------|--|---------------|
| 1 | 2 | 3 | 4 |
| __MigrationHistory | dbo | | 4 |
| Criteria | dbo | Таблица, в которой хранятся критерии | 2 |
| ExerciseCriteria | dbo | Таблица, которая определяет связь упражнения с критерием | 4 |
| Exercises | dbo | Таблица, в которой хранятся упражнения | 5 |
| ProfessionCriteria | dbo | | 4 |
| Professions | dbo | Таблица, в которой хранятся профессии | 3 |
| Roles | dbo | Таблица, в которой определены существующие роли | 3 |

Продолжение таблицы 3.1

| 1 | 2 | 3 | 4 |
|---------------|-----|--|----|
| Settings | dbo | Таблица с персональными настройками пользователя для приложения | 2 |
| TrainingTimes | dbo | Таблица, в которой определено предпочтительное время перерыва для пользователя | 3 |
| UserClaims | dbo | Таблица с характеристиками пользователя | 4 |
| UserExercises | dbo | Упражнения, которые сгенерированы для определенного пользователя и определенной его тренировки | 4 |
| UserLogins | dbo | Таблица, которая определяет связи пользователя с определенным внешним провайдером аутентификации | 3 |
| UserRoles | dbo | Роли пользователя | 2 |
| Users | dbo | Таблица пользователей | 13 |
| UserTrainings | dbo | Таблица тренировок пользователя | 4 |

Таблица 3.2 – Описание таблицы __MigrationHistory

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|----------------|---------------------------------|------------|--------------------|-----------------|
| MigrationId | Идентификатор миграции | nvarchar | 150 | Нет |
| ContextKey | Ключ контекста Entity Framework | nvarchar | 300 | Нет |
| Model | Hash-код модели | varbinary | 2147483647 | Нет |
| ProductVersion | Версия продукта | nvarchar | 32 | Нет |

Таблица 3.3 – Описание таблицы Criteria

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|----------|------------------------|------------|--------------------|-----------------|
| Id | Идентификатор критерия | int | 4 | |
| Name | Название критерия | nvarchar | 1073741823 | Да |

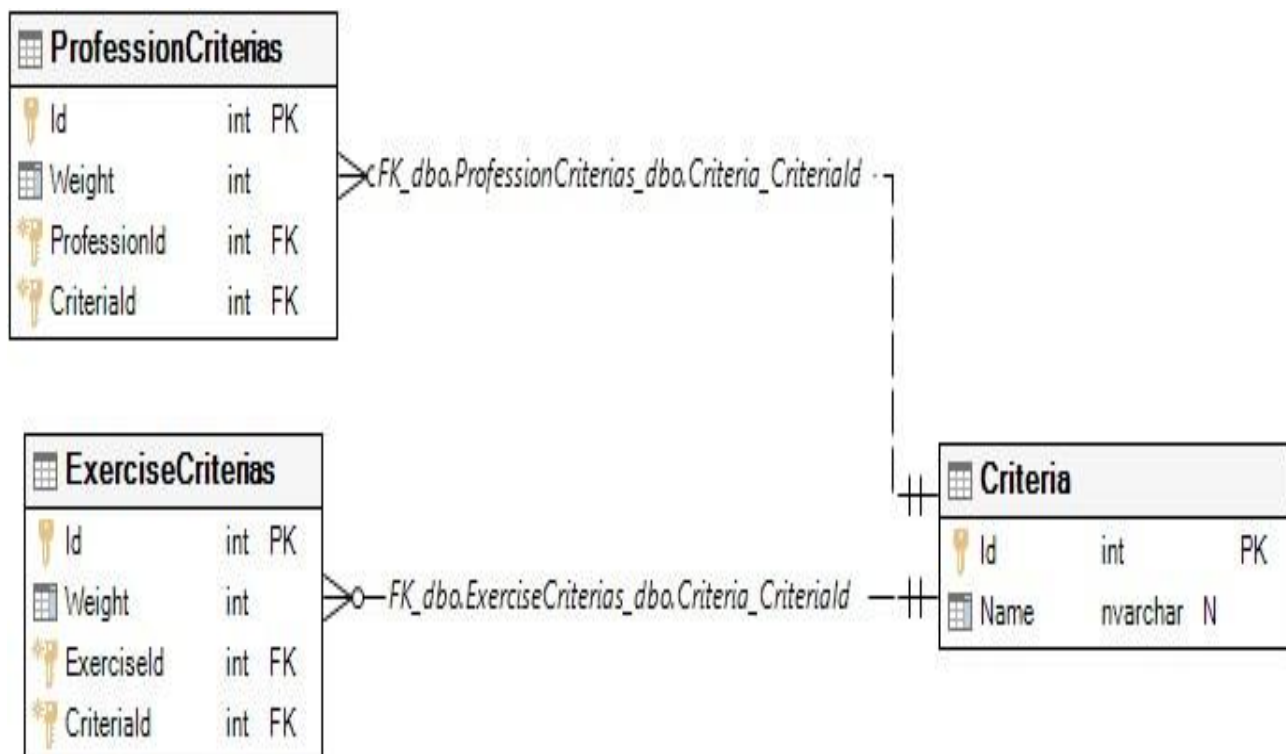


Рисунок 3.1 – Модель данных в отношении таблицы Criteria

Таблица3.3 – Описание таблицы ExerciseCriteria

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|------------|--------------------------|------------|--------------------|-----------------|
| Id | Идентификатор | int | 4 | Нет |
| Weight | Вес отношения | int | 4 | Нет |
| ExerciseId | Идентификатор упражнения | int | 4 | Нет |
| CriteriaId | Идентификатор критерия | int | 4 | Нет |

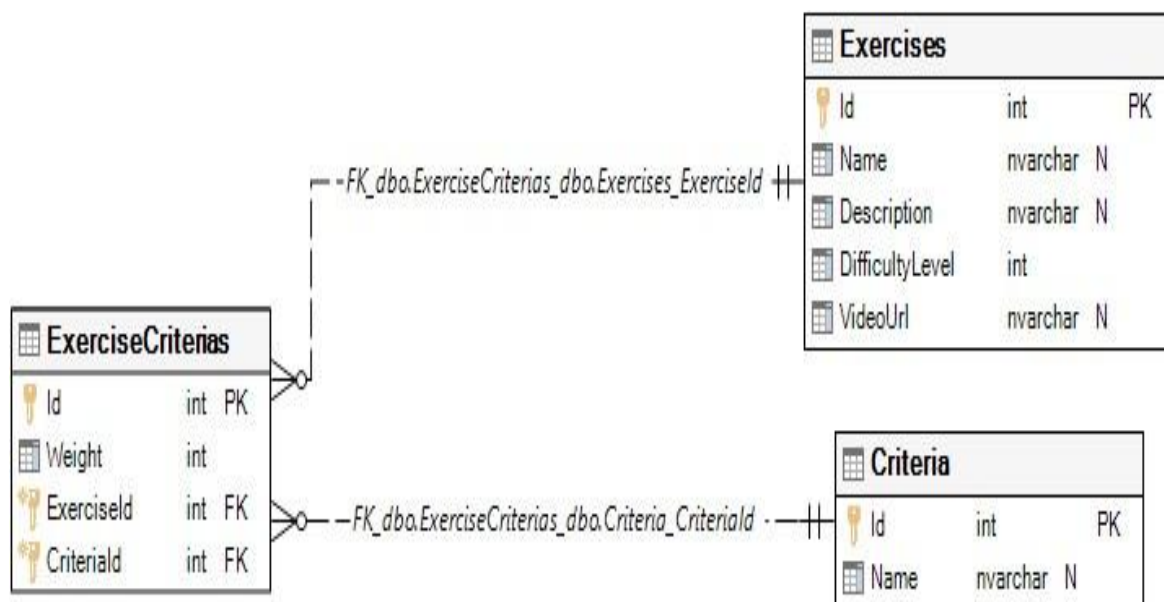


Рисунок 3.2 – Модель данных в отношении таблицы ExerciseCriteria

Таблица 3.4 – Описание таблицы Exercises

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|-----------------|--|------------|--------------------|-----------------|
| Id | Идентификатор | int | 4 | Нет |
| Name | Название упражнения | nvarchar | 1073741823 | Да |
| Description | Описание упражнения | nvarchar | 1073741823 | Да |
| DifficultyLevel | Сложность упражнения. Значения: 0 - Легкое. 1 - Средней сложности. 2 - Тяжелое | int | 4 | Нет |
| VideoUrl | Ссылка на видео | nvarchar | 1073741823 | Да |

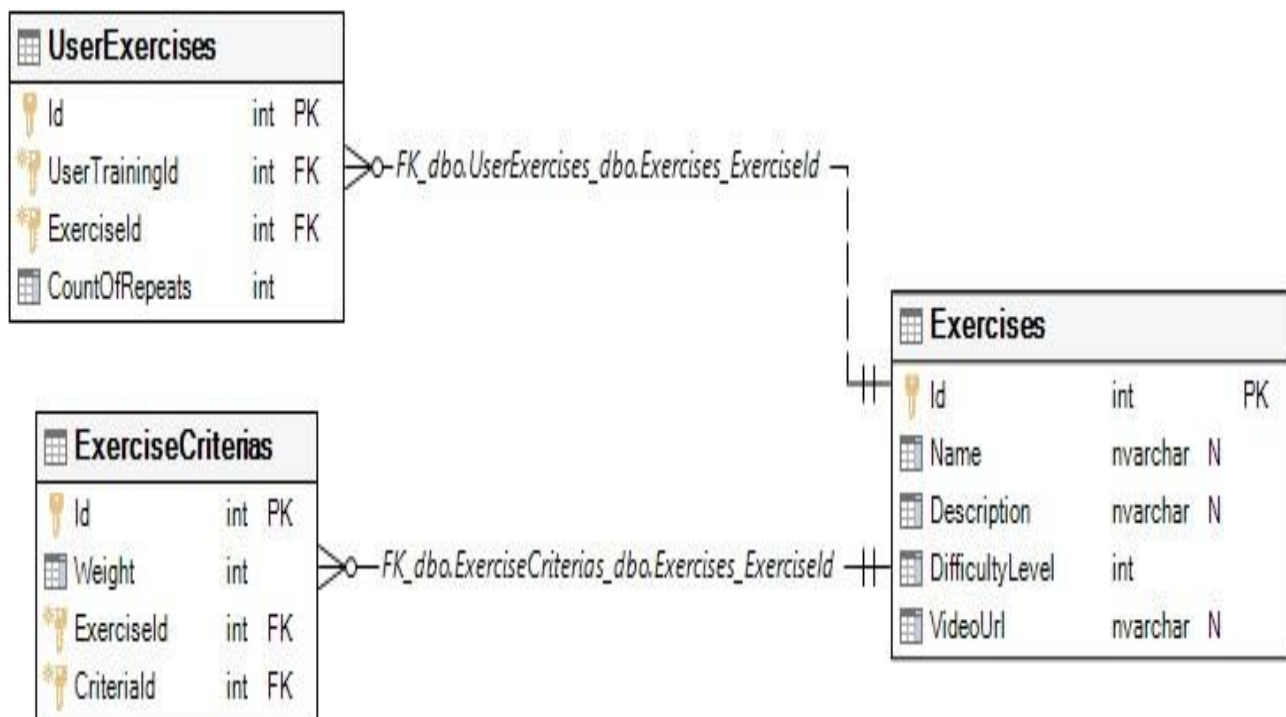


Рисунок 3.3 – Модель данных таблицы Exercises

Таблица 3.5 – Описание таблицы ProfessionCriteria

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|--------------|-------------------------|------------|--------------------|-----------------|
| Id | Идентификатор отношения | int | 4 | Нет |
| Weight | Вес отношения | int | 4 | Нет |
| ProfessionId | Идентификатор профессии | int | 4 | Нет |
| Criteriald | Идентификатор критерия | int | 4 | Нет |

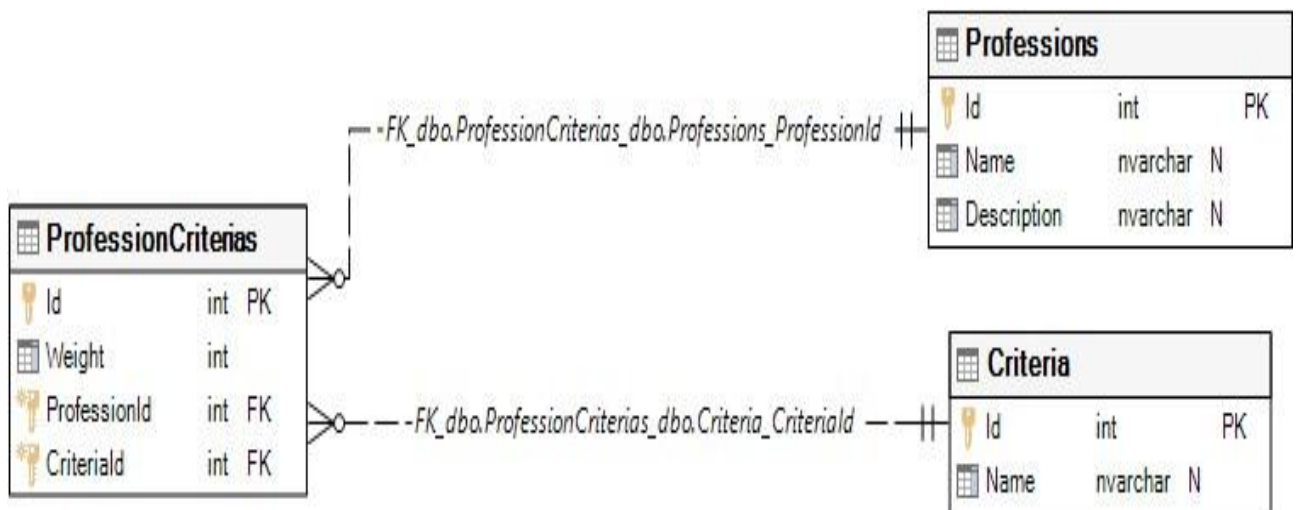


Рисунок 3.4 – Модель данных таблицы **ProfessionCriteria**

Таблица 3.6 – Описание таблицы **Professions**

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|-------------|--------------------|------------|--------------------|-----------------|
| Id | | int | 4 | Нет |
| Name | Название профессии | nvarchar | 1073741823 | Да |
| Description | Описание профессии | nvarchar | 1073741823 | Да |

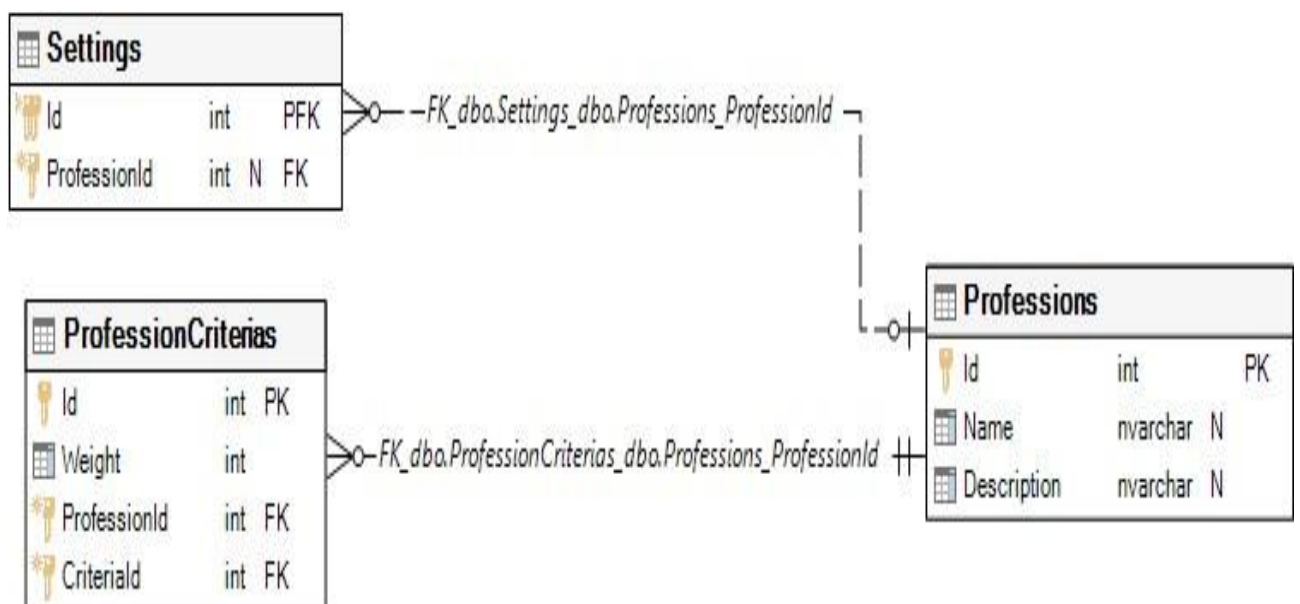


Рисунок 3.5 – Модель данных таблицы **Professions**

Таблица 3.7 – Описание таблицы Roles

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|------------|----------------------------|------------|--------------------|-----------------|
| Id | Идентификатор роли | int | 4 | Нет |
| CreatedUtc | Дата и время создания роли | datetime | 8 | Нет |
| Name | Название роли | nvarchar | 256 | Нет |



Рисунок 3.6 – Модель данных таблицы Roles

Таблица 3.8 – Описание таблицы Settings

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|--------------|---|------------|--------------------|-----------------|
| Id | Идентификатор настройки. Совпадает с идентификатором пользователя | int | 4 | Нет |
| ProfessionId | Идентификатор профессии. Настройка может существовать без профессии. Предполагается программное управление наличием профессии | int | 4 | Да |

Таблица 3.9 – описание таблицы TrainingTimes

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|------------|--|------------|--------------------|-----------------|
| Id | Идентификатор предустановленного времени | int | 4 | Нет |
| Value | Значение времени | time | 5 | Нет |
| SettingsId | Идентификатор настройки, к которой относится это время | int | 4 | Нет |



Рисунок 3.7 – Модель данных таблицы TrainingTimes

Таблица 3.10 – Описание таблицы UserClaims

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|------------|---|------------|--------------------|-----------------|
| Id | Идентификатор характеристики | int | 4 | Нет |
| UserId | Идентификатор пользователя | int | 4 | Нет |
| ClaimType | Тип характеристики. Например, роль, возраст и тд. | nvarchar | 1073741823 | Да |
| ClaimValue | Значение характеристики | nvarchar | 1073741823 | Да |

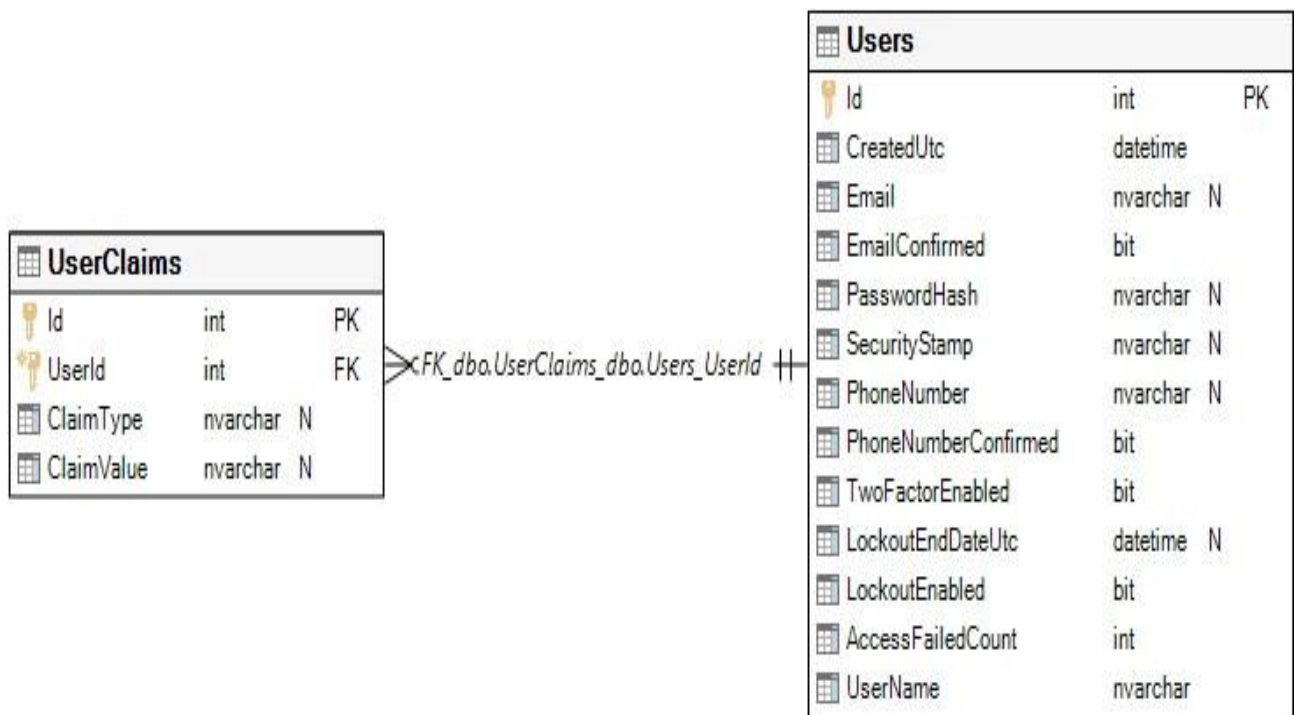


Рисунок 3.8 – Модель данных таблицы UserClaims

Таблица 3.11 – Описание таблицы UserExercises

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|----------------|--|------------|--------------------|-----------------|
| Id | Идентификатор пользовательского упражнения | int | 4 | Нет |
| UserTrainingId | Идентификатор тренировки | int | 4 | Нет |
| ExerciseId | Идентификатор упражнения | int | 4 | Нет |
| CountOfRepeats | Число повторений упражнения | int | 4 | Нет |

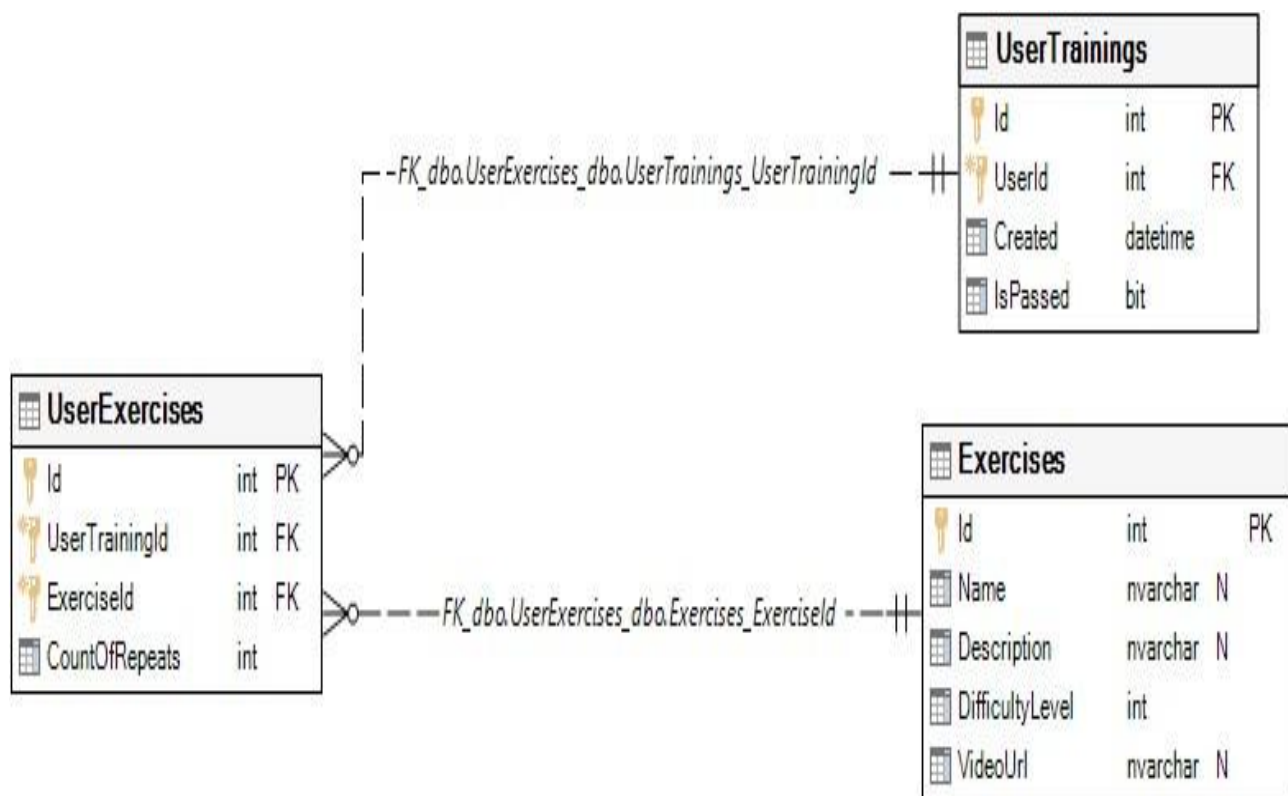


Рисунок 3.9 – Модель данных таблицы UserExercises

Таблица 3.12 – Описание таблицы UserLogins

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|---------------|-------------------------------------|------------|--------------------|-----------------|
| LoginProvider | Определяет логин во внешнем сервисе | nvarchar | 128 | Нет |
| ProviderKey | Определяет ключ внешнего сервиса | nvarchar | 128 | Нет |
| UserId | Идентификатор пользователя | int | 4 | Нет |

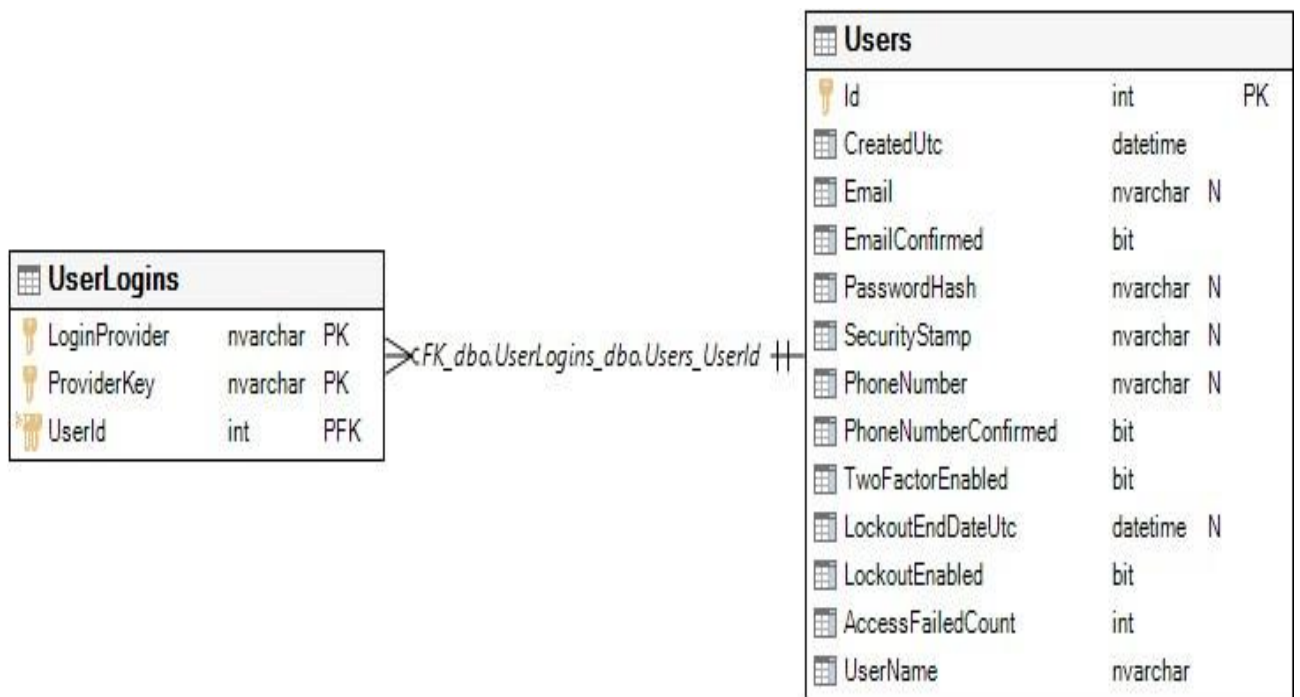


Рисунок 3.10 – Модель данных таблицы UserLogins

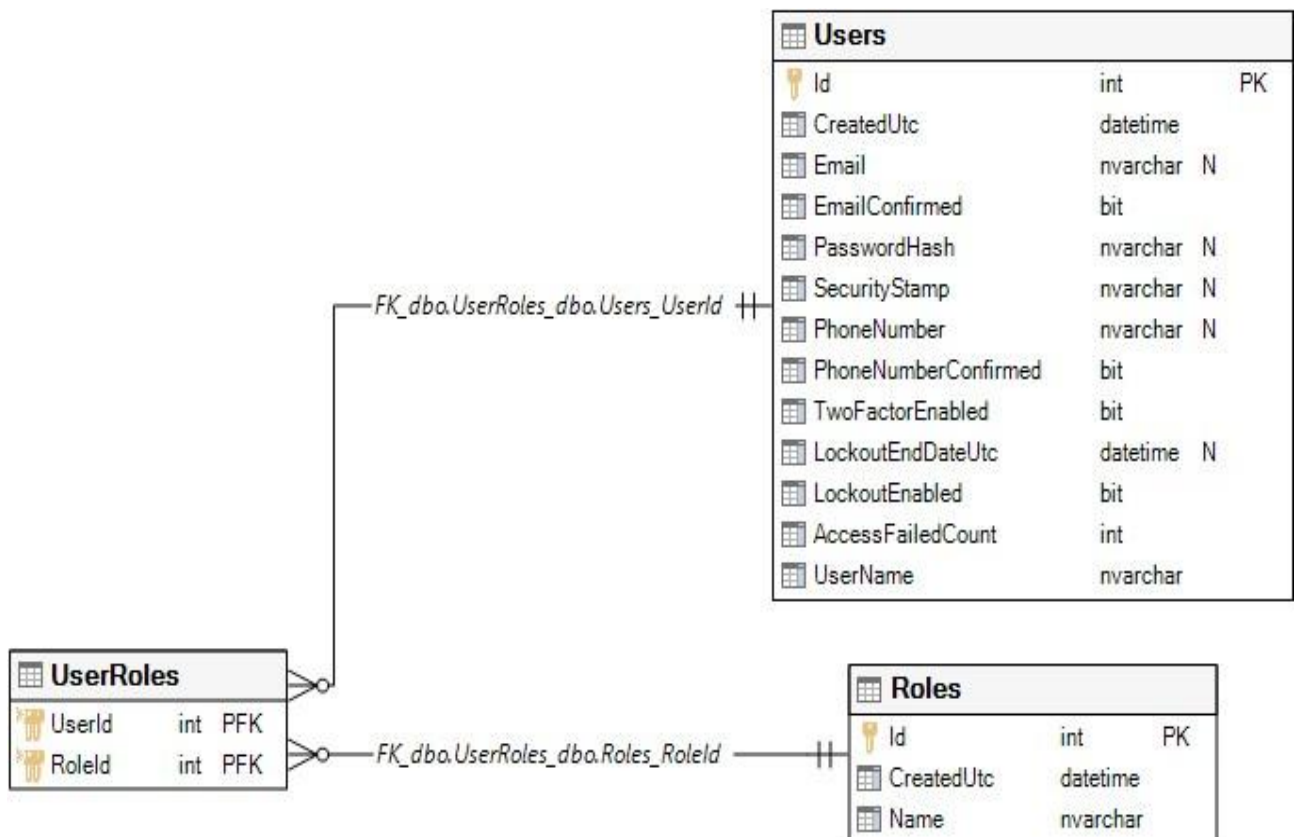


Рисунок 3.11 – Модель данных таблицы UserRoles

Таблица 3.13 – Описание таблицы UserRoles

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|----------|----------------------------|------------|--------------------|-----------------|
| UserId | Идентификатор пользователя | int | 4 | Нет |
| RoleId | Идентификатор роли | int | 4 | Нет |

Таблица 3.14 – Описание таблицы Users

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|----------------------|----------------------------|------------|--------------------|-----------------|
| Id | Идентификатор пользователя | int | 4 | Нет |
| CreatedUtc | | datetime | 8 | Нет |
| Email | | nvarchar | 256 | Да |
| EmailConfirmed | | bit | 1 | Нет |
| PasswordHash | Хеш пароля пользователя | nvarchar | 1073741823 | Да |
| SecurityStamp | | nvarchar | 1073741823 | Да |
| PhoneNumber | | nvarchar | 1073741823 | Да |
| PhoneNumberConfirmed | | bit | 1 | Нет |
| TwoFactorEnabled | | bit | 1 | Нет |
| LockoutEndDateUtc | | datetime | 8 | Да |
| LockoutEnabled | | bit | 1 | Нет |
| AccessFailedCount | | int | 4 | Нет |
| UserName | Имя пользователя | nvarchar | 256 | Нет |

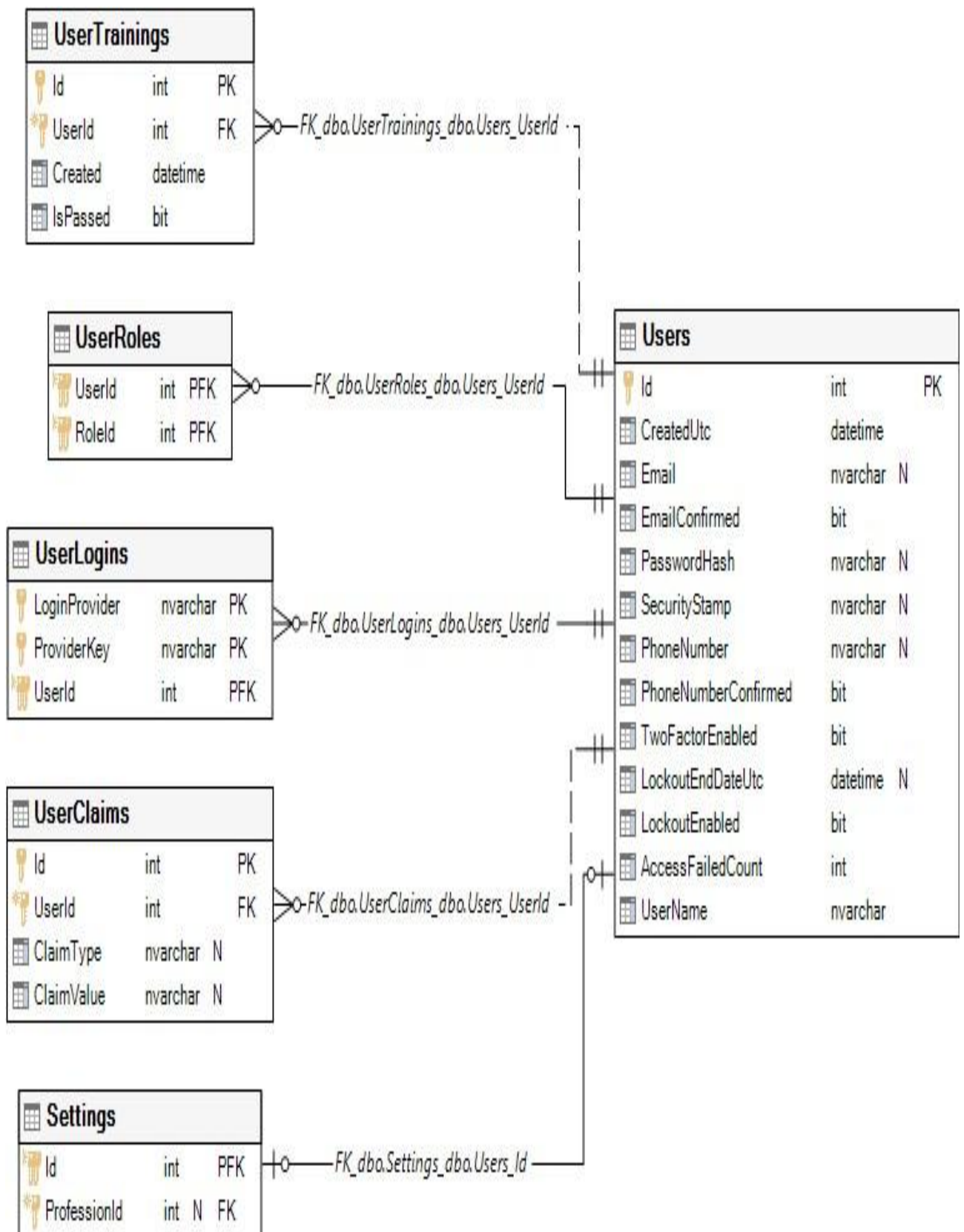


Рисунок 3.12 – Модель данных таблицы Users

Таблица 3.15 – Описание таблицы UserTrainings

| Название | Описание | Тип данных | Максимальная длина | Может быть null |
|----------|---|------------|--------------------|-----------------|
| Id | Идентификатор тренировки | int | 4 | Нет |
| UserId | Идентификатор пользователя | int | 4 | Нет |
| Created | Время создания тренировки | datetime | 8 | Нет |
| IsPassed | Определяет, пройдена тренировка или нет | bit | 1 | Нет |

3.1.2 Разработка архитектуры серверного приложения. Языком разработки дипломного проекта является C#, а основной технологией является ASP.NET MVC. Одним из наиболее подходящих шаблонов архитектурного проектирования для данного набора технологий и типа приложений является «Многоуровневая архитектура». Каждый уровень обеспечивает необходимый уровень абстракции, обычно, таких уровня 3.

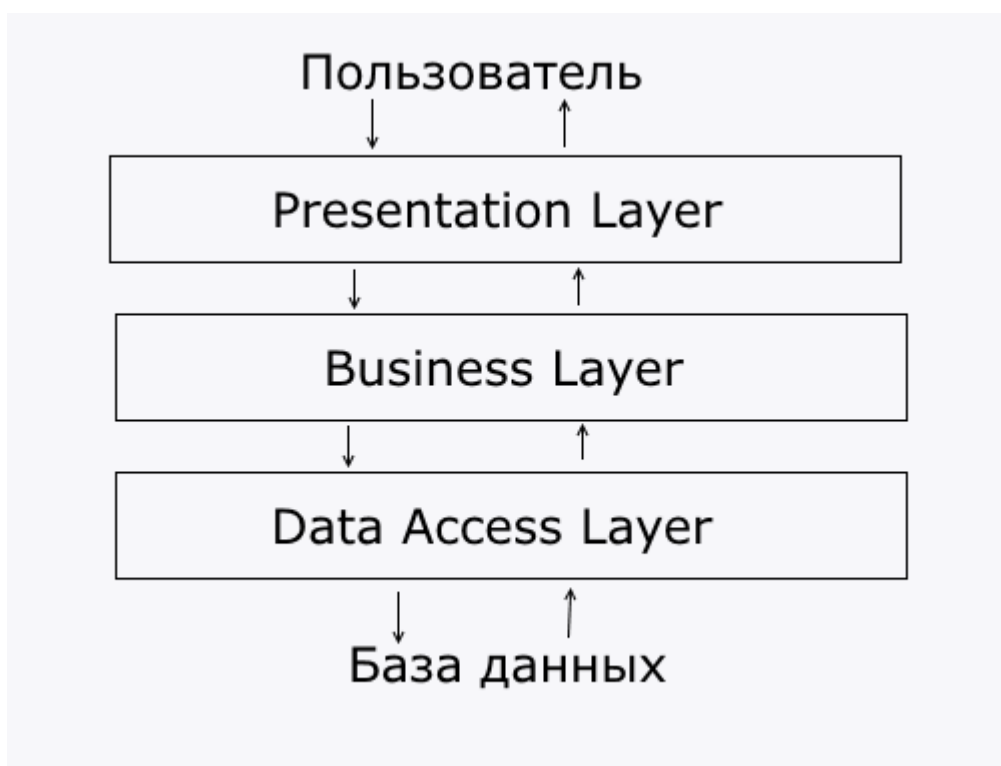


Рисунок 3.13 – Пример трехуровневой архитектуры

Presentation layer (или уровень представления) – это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя. Применительно к asp.net mvc на данном уровне расположены представления и все те компоненты, который составляют пользовательский интерфейс (стили, статичные страницы html, javascript), а также модели представлений, контроллеры, объекты контекста запроса. Так же на этом слое располагается внешний программный интерфейс приложения.

Business layer (или уровень бизнес-логики) содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

Data Access layer (или уровень доступа к данным) хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

Для обеспечения максимально высокого уровня расширяемости и поддержки хорошим выбором будут паттерны проектирования «Репозиторий» и «Единица работы».

Паттерн «Репозиторий» позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Паттерн «Единица работы» позволяет упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же контекст данных.

Для предоставления внешнего программного интерфейса приложения в данном дипломном проекте используется технология ASP.NET Web API 2.

Web API представляет собой веб-службу, которая может взаимодействовать с различными приложениями. При этом приложение может быть web-приложением ASP.NET, либо может быть мобильным или обычным десктопным приложением.

Также надо отметить, что платформа Web API 2 не является частью фреймворка ASP.NET MVC и может быть задействована как в связке с MVC, так и в соединении с Web Forms. Поэтому в Web API имеется своя система версий. Так, первая версия появилась с .net 4.5. А вместе с .NET 4.5.1 и MVC 5 вышла Web API 2.0.

Для связи с базой данных используется ORM (объектно-реляционное отображение) EntityFramework 6.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

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

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

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

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

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

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

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

- Database First. Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
- Model First. Сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
- Code First. Разработчик создает класс модели данных, которые будут храниться в БД, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В данной дипломной работе использовался подход Code First. Тем не менее, имея схему базы данных, описанную выше, имеется возможность восстановить модель, используя подход Database First.

В результате серверная часть приложения состоит из следующих наборов библиотек:

- Services.DTO;
- Services.Implementations;
- Services.Interfaces;
- Data.Extensions;
- Data.Implementations;
- Data.Interfaces;
- Contracts;
- Entity;
- WebUI.

Рассмотрим каждую из библиотек в отдельности.

Services.DTO предоставляет наборы DTO (data transfer object) для каждой сущности модели. Это позволяет обеспечить необходимый уровень абстракции и избежать зависимостей от сущностей DAL слоя.

Services.Implementations предоставляет реализации для интерфейсов библиотеки Services.Interfaces. Services.Interfaces в свою очередь предоставляет внешний API для уровня презентации и Web API.

Data.Extensions предоставляет набор расширений для стандартных функций EntityFramework, в частности, пагинацию и загрузку бесконечных списков.

Data.Implementations предоставляет реализации интерфейсов библиотеки Data.Interfaces. Помимо этого включает в себя всю необходимую информацию о базе

данных, такую, как миграции и контекст. Здесь же реализованы паттерны «Репозиторий» и «Единица работы».

Contracts представляет из себя библиотеку контрактов. Эти контракты позволяют делать приложение более надежным в разработке и избежать большого числа проверок данных.

Entity включает в себя полную модель данных.

WebUI включает в себя интерфейсы Web API для приложения для настольного ПК, для панели администратора и для мобильного приложения. Здесь реализованы механизмы аутентификации и авторизации.

WebUI.Tests содержит в себе тесты для контроллеров WebAPI.

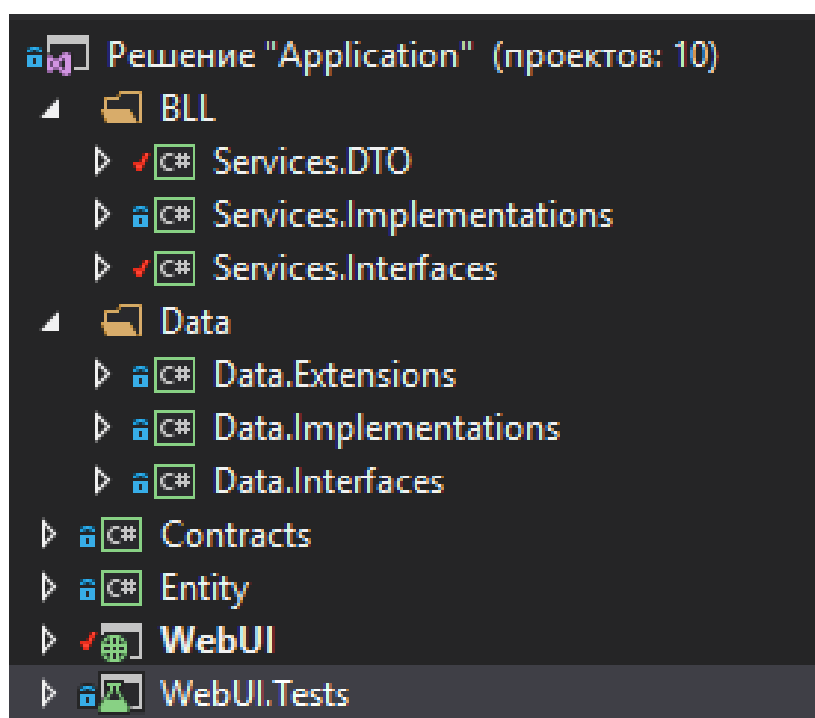


Рисунок 3.14 – Отображение структуры проекта в окне Обозревателя решений Visual Studio 2017

Модель данных библиотеки классов Entity представлена в Приложении Б.

3.2 Разработка панели администрирования

3.2.1 Разработка диаграмм вариантов использования и деятельности. Панель администрирования должна предоставлять интерфейс и реализацию для:

- регистрации и входу в учетную запись;
- созданию, редактированию и удалению критерий;
- созданию, редактированию и удалению профессий;
- созданию, редактированию и удалению упражнений.

В Приложении В отображена диаграмма вариантов использования для пользователя административной панели. В Приложении Г отображена диаграмма деятельности для основной функции, которую должна обеспечивать административная панель – создание и редактирование упражнений.

3.2.2 Разработка программного интерфейса. Панель администрирования разработана с помощью JavaScript библиотеки VueJs.

Web-приложения должны представлять графический интерфейс для работы с серверной частью. Работа с серверной частью осуществляется через Web API. Для работы с серверной частью используется библиотека Axios, которая представляет собой полноценный REST клиент.

REST (или «передача состояния представления») это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.

В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В RESTful-сервисах данные обычно передаются в формате JSON.

3.2.3 Использование компонентного подхода. Современные web-приложения столь же сложны, как и любые другие программные приложения, и зачастую создаются несколькими людьми, объединяющими усилия для создания финального продукта. В таких условиях, чтобы повысить эффективность, естественно искать правильные способы разделения работы на участки с минимальными пересечениями между людьми и подсистемами. Внедрение

компонентного подхода (в целом) – это то, как обычно решается такая задача. Любая компонентная система должна уменьшать общую сложность через предоставление изоляции, или естественных барьеров, скрывающих сложность одних систем от других. Хорошая изоляция также облегчает повторное использование и внедрение сервисных парадигм.

Все эти механизмы покрывает система однофайловых компонентов VueJs.

Компоненты развивают идею плагинов. Каждый из них реализует какую-то свою возможность (а если нет существующих, то можно написать и свой). Если понадобится реализовать подобное в другом месте – легко переиспользовать плагин снова. Взаимодействие можно описать простым интерфейсом: отправляем в плагин входные параметры, а для обратной связи можем отслеживать события.

Всё это справедливо и для компонентов. С тем лишь отличием, что компонент может представлять собой не только одну вещь (например, красивый и функциональный select), но и какую-то часть приложения, которая должна работать и выглядеть везде единообразно (например, форма комментирования, с аватаркой, редактором и красивым select'ом).

Компонентный подход позволяет избежать мешанины кода и чётко выстраивать архитектуру приложения. Любую сложную страницу всегда можно разбить на меньшие составляющие. Каждую из таких частей при выделении в компонент проще поддерживать, а при необходимости повторять разбиение внутри компонента на ещё меньшие части.

Первым заметным плюсом подобного разбиения на компоненты будет удобство в поддержке – больше не нужно держать в голове логику всего приложения, можно сосредоточиться на конкретной его части. Вносить изменения или доработки нужно будет только в одном месте. Изолированность же компонентов избавит от появления конфликтов с другими частями приложения.

Поэтому применение компонентного подхода теперь широко используется во многих фреймворках. Vue не остался в стороне и предоставляет прекрасные возможности по работе с компонентами.

3.3 Разработка приложения для ПК

3.3.1 Разработка диаграмм вариантов использования и деятельности. Приложение для настольных ПК должно предоставлять интерфейс и реализацию для:

- Регистрации и входу в учетную запись;
- Просмотру всех упражнений;
- Настройки профессии;
- Настройки предпочтительного времени;

- Оповещения о необходимости провести профилактическую тренировку;
- Просмотра и отслеживания результатов выполнения тренировки.

В Приложении Д отображена диаграмма возможностей пользователя приложения для настольного ПК. В Приложении Е отображена диаграмма деятельности для основной функции, которую должно обеспечить приложение для настольного ПК – выполнение профилактической разминки.

3.3.2 Разработка программного интерфейса. Работа программного интерфейса очень схожа с работой программного интерфейса административной части. Т.к. для разработки приложения для настольных ПК используется Electron, то есть возможность применить библиотеку Axios, которая реализует RESTful клиент.

Основной особенностью разработки программного интерфейса является способ взаимодействия приложений, написанных с помощью Electron, с возможностями платформы, на которой запускается такое приложение.

В Electron доступен ряд API-интерфейсов, поддерживающих разработку настольных приложений в обоих процессах - main process и render process. Для доступа к этим процессам API Electron вам необходимо включить в проект модуль «electron».

Все электронные API-интерфейсы назначаются типам процессов. Многие из них могут использоваться только из основного процесса, некоторые из них только из процесса рендеринга. В документации для каждого отдельного API указывается, из какого процесса он может быть использован.

Поскольку возможна связь между процессами, процесс визуализации может вызвать основной процесс для выполнения задач. Electron поставляется с модулем, называемым remote, который предоставляет API, обычно доступные только в основном процессе. Чтобы создать BrowserWindow из процесса рендеринга, необходимо использовать remote модуль как промежуточный модуль.

Все API, доступные в Node.js, доступны и в Electron.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Административная панель

Для того, чтобы попасть в административную панель, нужно зайти или зарегистрироваться. На Рисунке 1 представлена форма аутентификации. Форма аутентификации содержит 2 поля: E-mail и пароль. Для того, чтобы войти, необходимо заполнить поля и нажать кнопку «Вход».

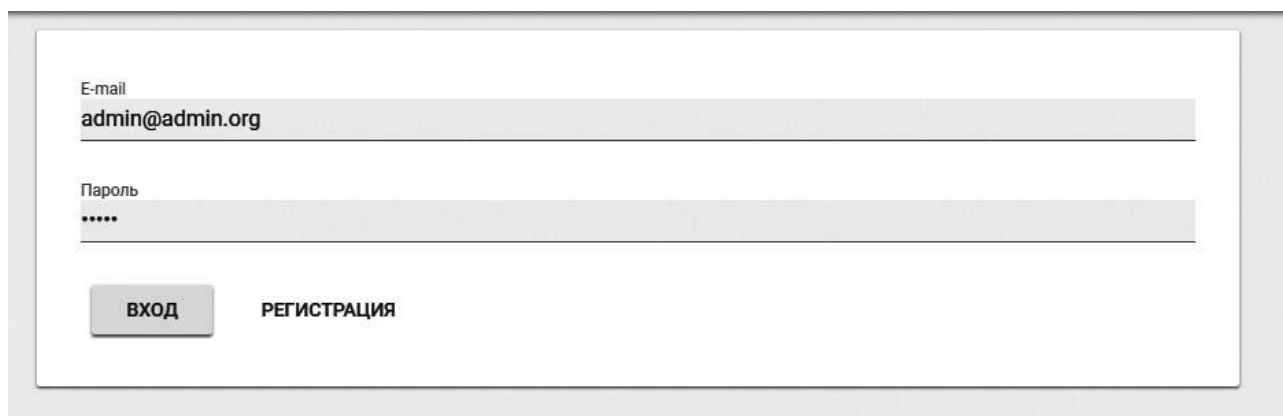


Рисунок 4.1 – Форма аутентификации пользователя

Если данные не были введены или были введены неверно, поля формы, в которых данные введены некорректно, будут окрашены в красный цвет и будет выведено соответствующее сообщение об ошибке ввода. Пока форма не будет заполнена корректно – данные не будут отправлены на сервер. Этот процесс называется «клиентская валидация» или «валидация на стороне клиента».



Рисунок 4.2 – Валидация на стороне клиента

В случае, если поля формы аутентификации были заполнены верно и прошли валидацию, но существует некая проблема на стороне сервера, или же сервер определил, что логин или пароль введен неправильно, справа внизу пользователь увидит предупреждение в красном блоке.

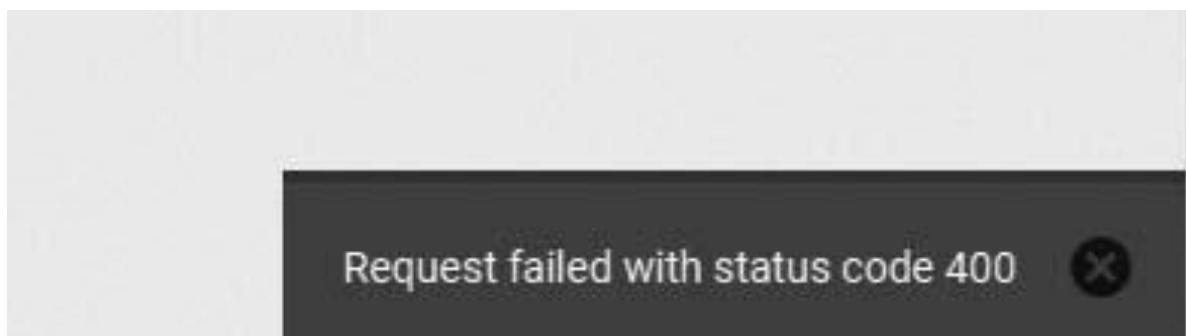


Рисунок 4.3 – Пример ошибки аутентификации на стороне сервера

Если пользователь отсутствует в базе, то он может зарегистрироваться в системе. Для этого необходимо нажать кнопку «Зарегистрироваться». Пользователь увидит форму регистрации, которая работает так же, как и форма входа. Пользователю необходимо заполнить форму и отправить ее, нажав кнопку «Зарегистрироваться».

E-mail

Email является обязательным

Пароль

Пароль является обязательным

Подтвердите пароль

ЗАРЕГИСТРИРОВАТЬСЯ

ВХОД

Рисунок 4.4 – Пример ошибки подтверждения пароля при регистрации

Как только пользователь проходит процесс аутентификации и авторизации, он будет перенаправлен на окно отображения всех критериев. С критериями можно проводить следующие действия:

- Создание нового критерия;
- Редактирование существующего критерия;
- Удаление критерия.

Форма создания/редактирования критерия представляет собой всплывающий диалог, в котором можно изменить имя критерия. Предусмотрена клиентская валидация.

Для того, чтобы создать критерий, необходимо нажать розовую кнопку со значком «+» в правом нижнем углу экрана.

Для того, чтобы обновить критерий, необходимо нажать кнопку в левом нижнем углу критерия.

Для того, чтобы удалить критерий, необходимо нажать на значок корзины в правом нижнем углу упражнения.



Рисунок 4.5 – Окно отображения всех критериев

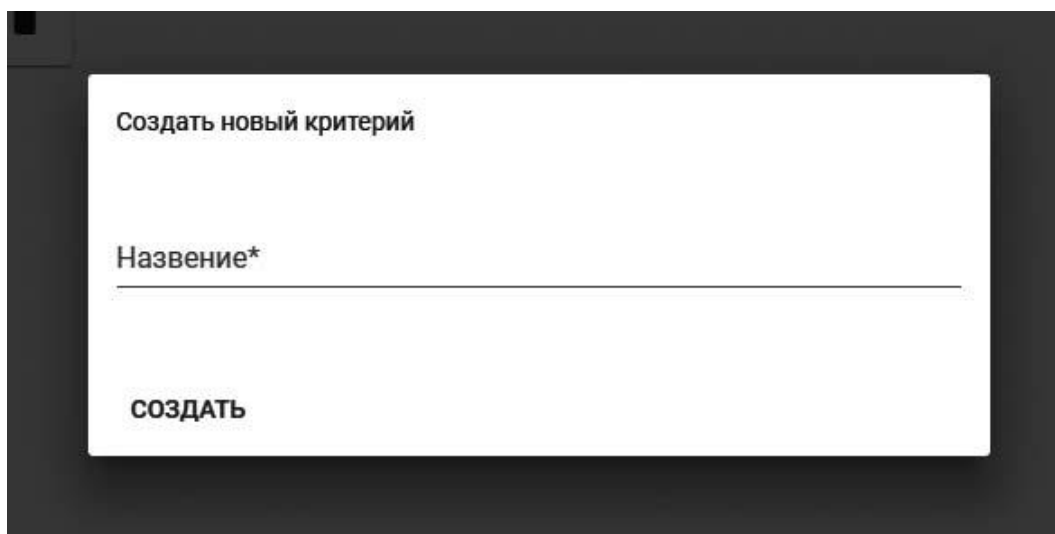


Рисунок 4.6 – Всплывающее окно создания критерия



Рисунок 4.7 – Всплывающее окно обновления критерия

Следующим окном в приложении является окно отображения всех упражнений. Для того, чтобы перейти на него, необходимо выбрать в меню пункт «Упражнения». Активная страница в меню подсвечивается.

С упражнениями можно проводить следующие действия:

- создание нового упражнения;
- редактирование существующего упражнения;
- удаление упражнения.

Для того, чтобы создать упражнение, необходимо нажать розовую кнопку со значком «+» в правом нижнем углу экрана.

Для того, чтобы обновить упражнение, необходимо нажать кнопку в левом нижнем углу упражнения.

Для того, чтобы удалить критерий, необходимо нажать на значок корзины в правом нижнем углу упражнения.



Рисунок 4.8 – Окно отображения всех упражнений

Название
Отжимания

Видео
<https://www.youtube.com/embed/EZf7IDkxnLc&t=1s>

Описание*
Отжимания - хорошее упражнение!

ДОБАВИТЬ КРИТЕРИЙ **СОХРАНИТЬ**

Рисунок 4.9 – Форма создания упражнения

Форма создания/редактирования упражнения состоит из следующего набора полей:

- Название упражнения;
- Ссылка на видео упражнения;
- Описание упражнения;
- Критерии и их вес для упражнения.

Присутствует особенность при заполнении полей «Видео» и «Критерии». При заполнении поля «Видео» необходимо обязательно указать ссылку на видео, которое размещено на хостинге youtube.com. Если ссылка введена неверно – пользователь увидит соответствующее сообщение. Если введена верно – пользователь увидит под полем «Видео» мини проигрыватель, в котором можно воспроизвести выбранное видео.

Для того, чтобы добавить новый критерий, необходимо нажать кнопку «Добавить критерий». После нажатия на кнопку пользователь увидит 2 поля для заполнения и кнопку удаления критерия со знаком «-». Поле «Выберите критерий» представляет собой выпадающий список. Когда пользователь выбирает одно из значений в списке, то оно блокируется и помечается, как занятое. Это нужно для того, чтобы не привязывать несколько одинаковых критериев к одному упражнению. В поле «Вес» необходимо определить, насколько полезным является это упражнение для данного критерия.

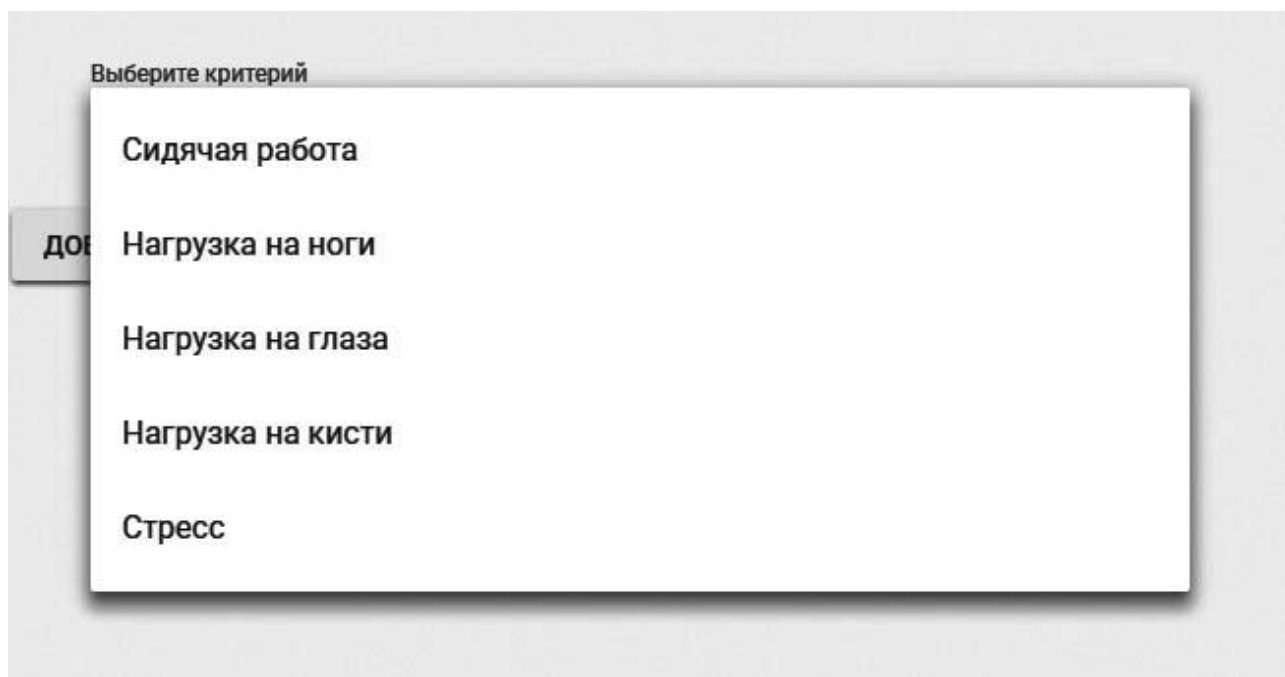


Рисунок 4.10 – Выпадающий список «Выберите критерий»

Для того, чтобы создать критерий, необходимо нажать кнопку «Сохранить».

Следующим окном является окно профессий. Для того, чтобы попасть на окно профессий, необходимо в меню выбрать пункт «Профессии». В окне «Профессии» отображаются все профессии.

С профессиями можно проводить следующие действия:

- Создание новой профессии;
- Редактирование существующей профессии;
- Удаление профессии.

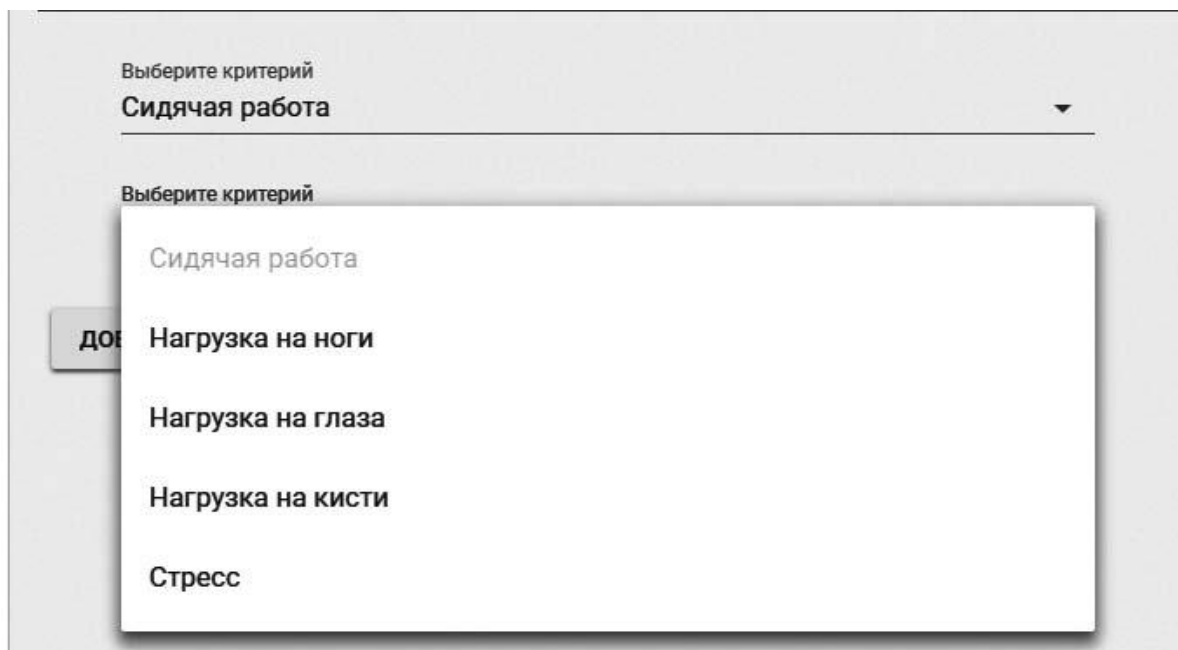


Рисунок 4.11 – Блокировка опции, если она уже выбрана ранее

Название

Отжимания

Видео

<https://www.youtube.com/embed/EZf7lDkxLc&t=1s>

Описание*

Nulla quis lorem ut libero malesuada feugiat. Vivamus suscipit tortor eget felis porttitor volutpat.

Выберите критерий

Сидячая работа

Вес

10

ДОБАВИТЬ КРИТЕРИЙ

СОХРАНИТЬ

Рисунок 4.12 – Форма редактирования упражнения

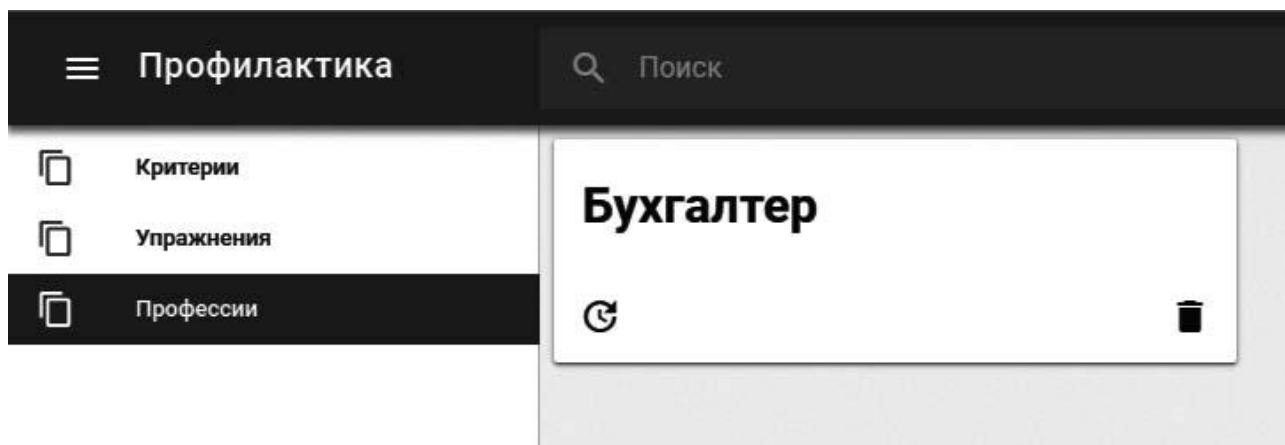


Рисунок 4.13 – Окно отображения всех профессий

Для того, чтобы создать профессию, необходимо нажать розовую кнопку со значком «+» в правом нижнем углу экрана.

Для того, чтобы обновить профессию, необходимо нажать кнопку в левом нижнем углу карточки профессии.

Для того, чтобы удалить профессию, необходимо нажать на значок корзины в правом нижнем углу карточки профессии.

Рисунок 4.14 – Форма создания профессии

Форма создания/редактирования профессии состоит из следующего набора полей:

- Название профессии;
- Описание профессии;
- Критерии и их вес для профессии.

Реализована клиентская валидация.

Для того, чтобы добавить новый критерий, необходимо нажать кнопку «Добавить критерий». После нажатия на кнопку пользователь увидит 2 поля для заполнения и кнопку удаления критерия со знаком «-». Поле «Выберите критерий» представляет собой выпадающий список. Когда пользователь выбирает одно из значений в списке, то оно блокируется и помечается, как занятое. Это нужно для того, чтобы не привязывать несколько одинаковых критериев к одному упражнению. В поле «Вес» необходимо определить, насколько влияющим является этот критерий для данной профессии.

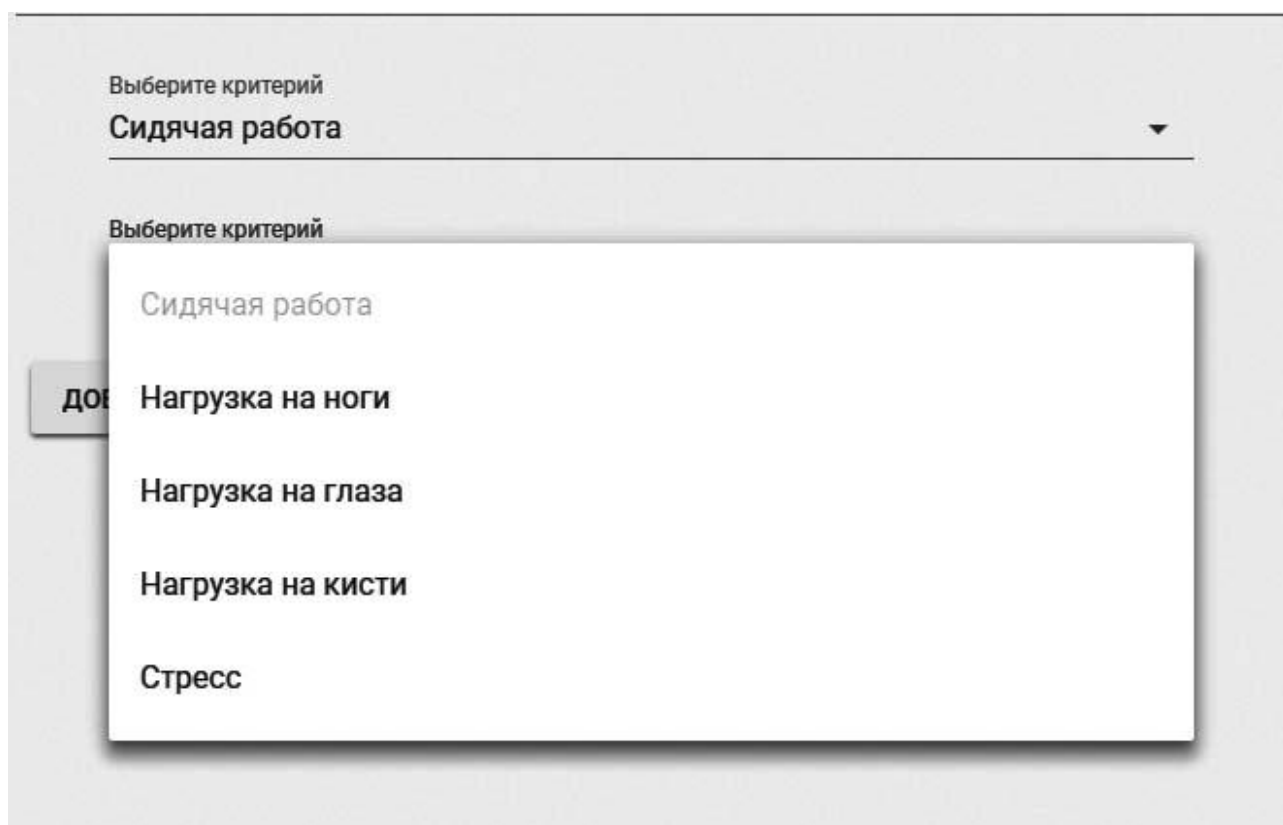


Рисунок 4.15 – Выпадающий список с критериями

Для того, чтобы создать профессию, необходимо нажать кнопку «Сохранить».

В случае ошибки в правом нижнем углу пользователь увидит соответствующее сообщение.

Выберите критерий
Сидячая работа

Вес
10

Выберите критерий

Вес
10

ДОБАВИТЬ КРИТЕРИЙ

СОХРАНИТЬ

Request failed with status code 500

Рисунок 4.16 – Ошибка при сохранении профессии, т.к. не указан критерий

4.2 Приложение для ПК

Для того, чтобы начать пользоваться приложением, необходимо пройти процесс аутентификации или зарегистрироваться. На Рисунке 1 представлена форма аутентификации. Форма аутентификации содержит 2 поля: E-mail и пароль. Для того, чтобы войти, необходимо заполнить поля и нажать кнопку «Вход».

E-mail

Email является обязательным

Пароль

Пароль является обязательным

ВХОД

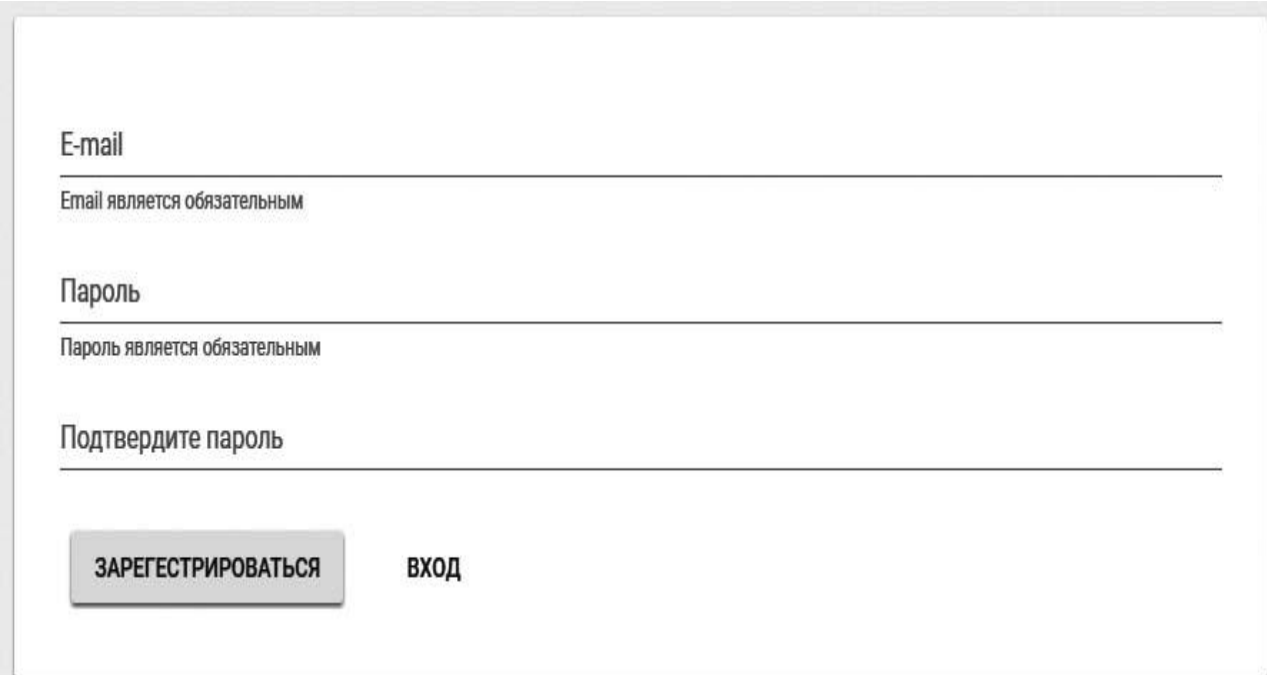
РЕГИСТРАЦИЯ

Рисунок 4.17 – Форма аутентификации пользователя

Если пользователь отсутствует в базе, то он может зарегистрироваться в системе. Для этого необходимо нажать кнопку «Зарегистрироваться». Пользователь увидит форму

регистрации, которая работает так же, как и форма входа. Пользователю необходимо заполнить форму и отправить ее, нажав кнопку «Зарегистрироваться».

Для форм регистрации и аутентификации реализована клиентская валидация. Если пользователь прошел этап клиентской валидации, но существует проблема на стороне сервера или другая проблема, в правом нижнем углу пользователь увидит соответствующее сообщение.



The image shows a registration form with the following elements:

- E-mail**: Input field with a label "Email является обязательным" below it.
- Пароль**: Input field with a label "Пароль является обязательным" below it.
- Подтвердите пароль**: Input field.
- Buttons/Links**: A button labeled "ЗАРЕГИСТРИРОВАТЬСЯ" and a link labeled "ВХОД".

Рисунок 4.18 – Форма регистрации пользователя

Как только пользователь пройдет процедуру аутентификации, он будет перенаправлен на страницу всех упражнений. На странице всех упражнений пользователь имеет возможность просмотра всех упражнений, но, что самое главное, здесь он может выполнять поиск.

Поиск выполняется автоматически, как только пользователь вводит что-то в поле поиска. Для того, чтобы выполнить переход к настройкам поиска, пользователь должен сделать двойной клик мышью на поле поиска или же дважды нажать клавишу «Enter».

В окне настроек поиска пользователь может настроить поиск:

- По профессии;
- По критериям;
- По упражнениям.

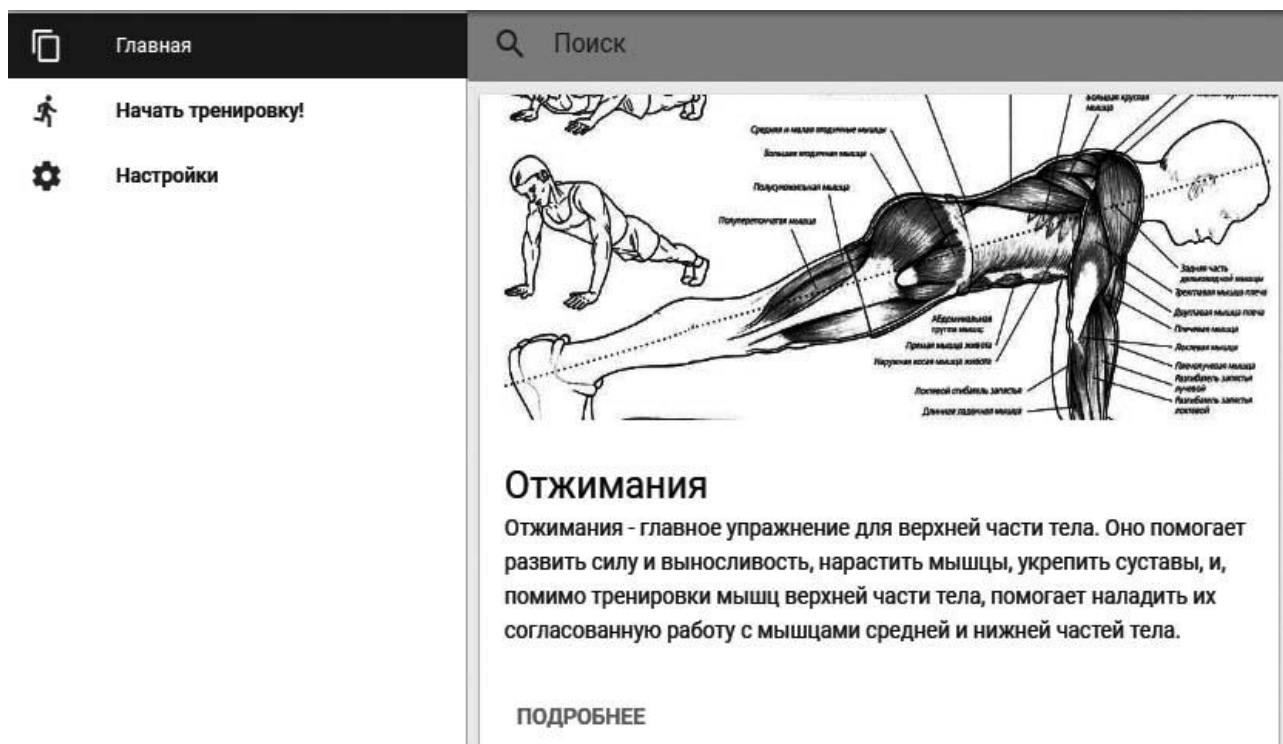


Рисунок 4.19 – Окно всех упражнений

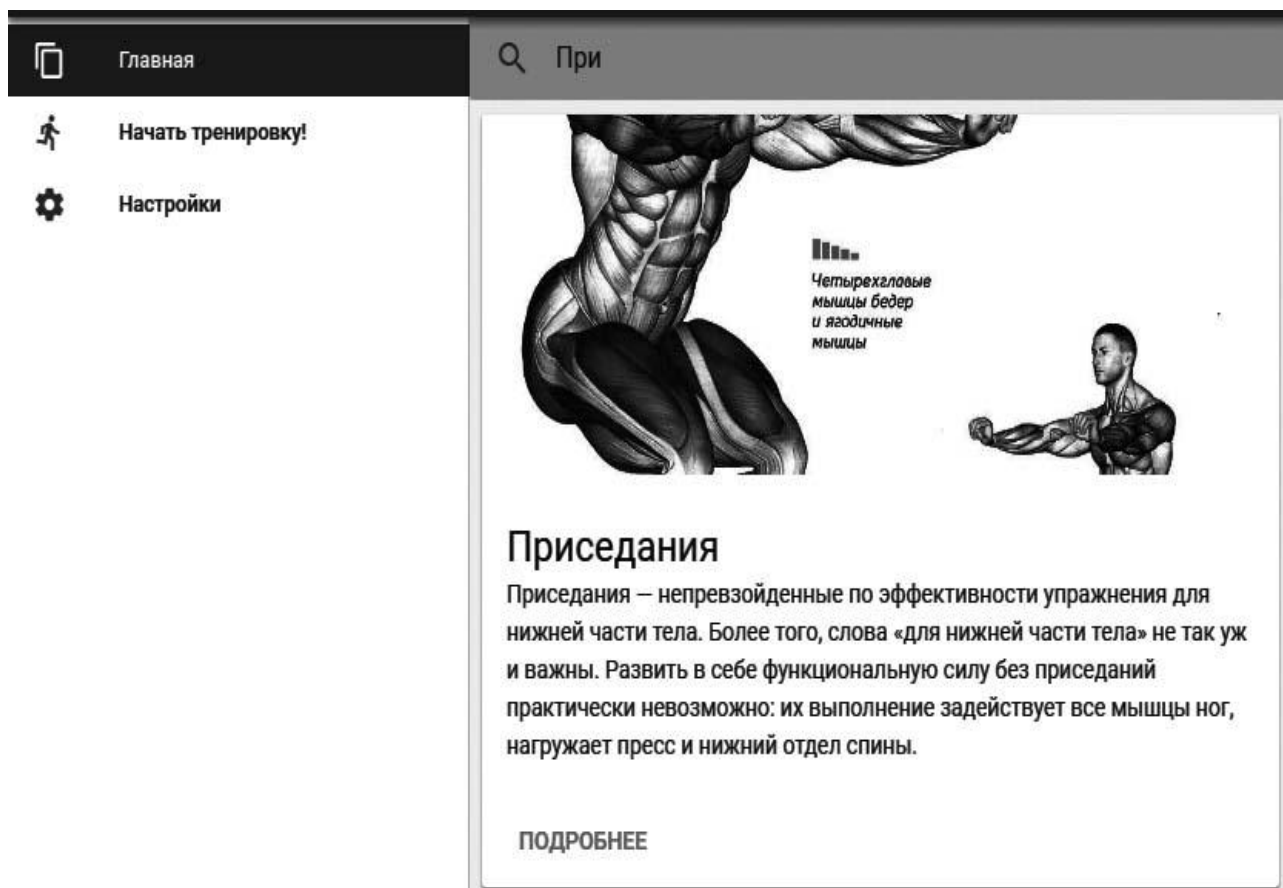


Рисунок 4.20 – Пример выполнения фильтрации по запросу

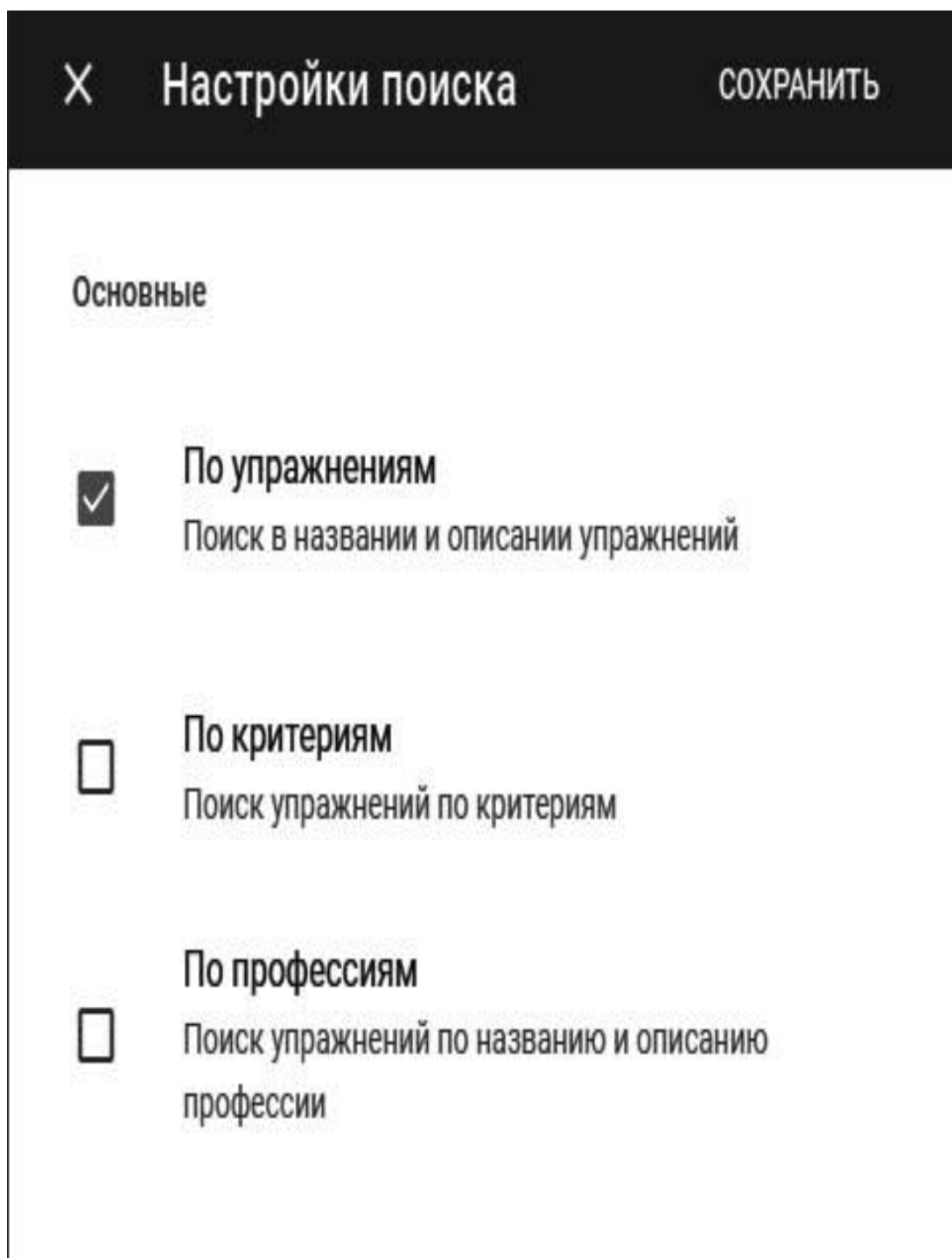


Рисунок 4.21 – Окно настроек поиска

Пользователь может перейти к конкретному упражнению и посмотреть видео, нажав на кнопку «Подробнее». Он будет перенаправлен на окно упражнения, где он сможет посмотреть видео и прочитать описание упражнения.



Рисунок 4.22 – Окно упражнения

Следующим окном является окно «Настройки». Для того, чтобы перейти в это окно, нужно в меню слева выбрать пункт меню «Настройки». Активное окно выделено характерным цветом.

ВЫБРАТЬ ПРОФЕССИЮ

Бухгалтер

Предпочтительное время

⌚ Выберите время 1

⌚ Выберите время 2

⌚ Выберите время 3

Рисунок 4.23 – Окно настройки

В окне «Настройки» пользователь может осуществить настройки работы приложения. Пользователь может настроить свою профессию и предпочтительное время перерывов.

Для того, чтобы выбрать профессию, необходимо нажать на кнопку «Выбрать профессию» или нажать на текущую профессию. Текущая профессия отображена в бирке рядом с кнопкой «Выбрать профессию». Выбор профессии осуществляется в всплывающем окне, в котором отображены все профессии. Так же есть возможность выполнять фильтрацию по профессии. Если пользователь определился с профессией, он должен нажать кнопку «Сохранить», чтобы сохранить результат. Иначе он должен нажать кнопку «Закрыть» или на любую свободную зону за пределами всплывающего окна.

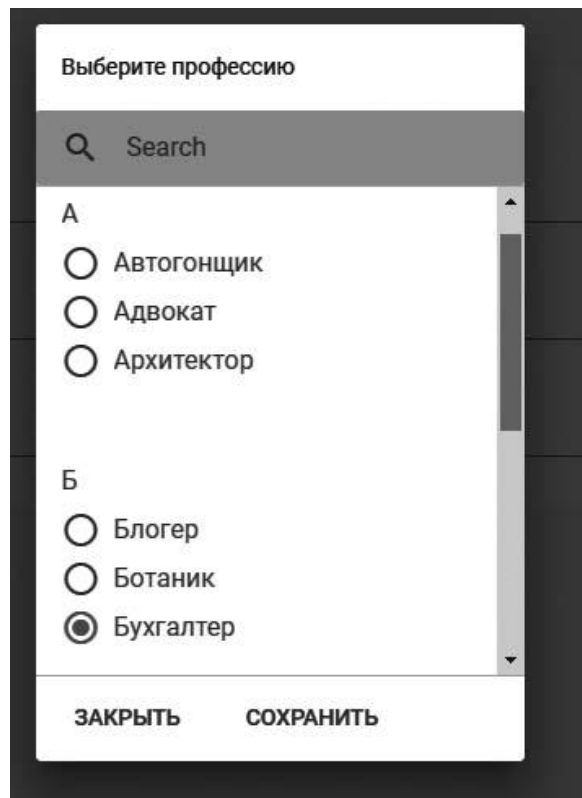


Рисунок 4.24 – Всплывающее окно настройки профессии

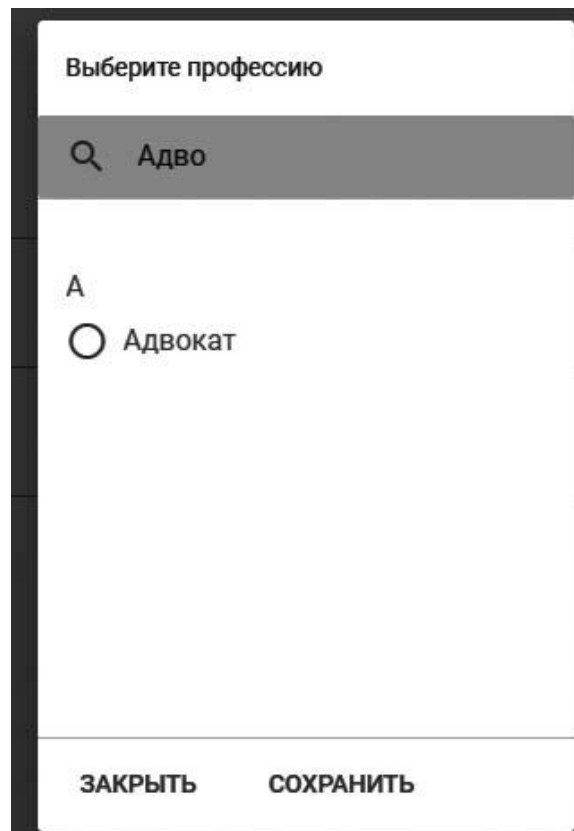


Рисунок 4.25 – Пример фильтрации профессий

Для того, чтобы выбрать предпочтительное время, необходимо совершить клик на поле ввода времени. Пользователь увидит всплывающее окно с интерактивными часами, на которых он может выбрать время.

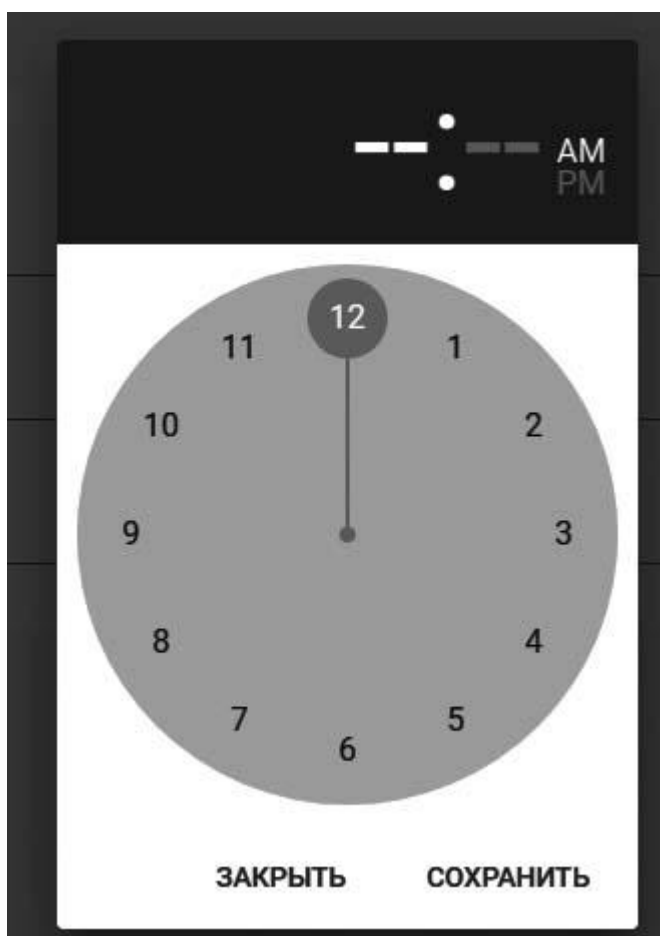


Рисунок 4.26 – Всплывающее окно с интерактивными часами

Для того, чтобы сохранить результат, необходимо нажать кнопку «Сохранить». В противном случае нужно нажать кнопку «Закрыть» или на любое свободное пространство не занятое модальным окном.

Следующим окном является окно профилактической тренировки. Здесь перечислены упражнения для ближайшей тренировки. Они сгруппированы по уровню сложности выполнения упражнения. Для каждого упражнения отображены число повторений и краткое описание. Для того, чтобы начать тренировку, нужно кликнуть на любое из упражнений. Начатая тренировка отменит ближайшую и ближайшей будет считаться следующая тренировка.

Окно упражнения тренировки представляет собой карточку, на которой размещено видео и описание техники выполнения упражнения.

При нажатии кнопки «Выполнено», пользователь будет перенаправлен к следующему упражнению. Как только упражнения заканчиваются, пользователь будет перенаправлен на окно всех упражнений.

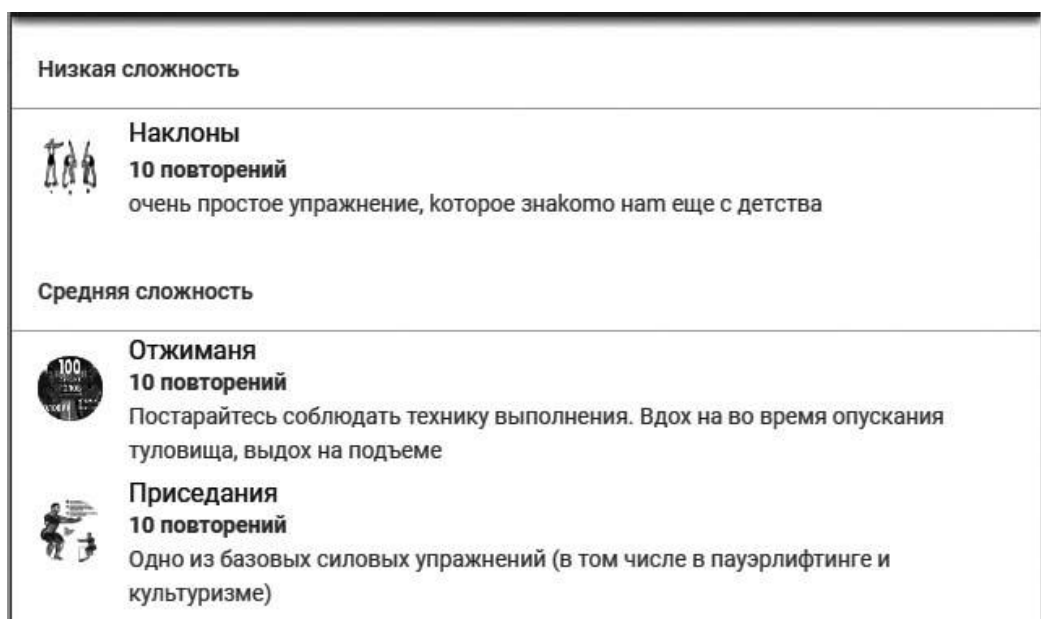


Рисунок 4.27 – Окно профилактической тренировки



Рисунок 4.28 – Окно упражнения тренировки

5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЯ ДЛЯ ПРОФИЛАКТИКИ ПРОФЕССИОНАЛЬНЫХ ЗАБОЛЕВАНИЙ

5.1 Краткая характеристика разрабатываемого web-приложения

Дипломный проект представляет собой проектирование и разработку web-приложения для профилактики профессиональных заболеваний на рабочем месте.

Целью разработки является web-приложение и приложение для настольных ПК под управлением ОС Windows или под управлением ОС системы UNIX.

Разработанная система, содержащая базу упражнений, обеспечит возможность профилактики профессиональных заболеваний, подобранных по индивидуальным особенностям с возможностью напоминания и отсроченной разминки.

5.2 Расчет себестоимости и разрабатываемого web-приложения

Себестоимость продукции представляет собой выраженные в денежной форме затраты организации на ее производство и реализацию.

Расчет полной себестоимости единицы продукции осуществляется по следующим калькуляционным статьям затрат:

- материалы и комплектующие изделия;
- затраты на оплату труда:
 - а) основная заработная плата научно-технического персонала;
 - б) дополнительная заработная плата научно-технического персонала;
- отчисления в Фонд социальной защиты населения;
- отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний;
- командировочные расходы
- услуги сторонних организаций
- прочие прямые расходы;
- накладные расходы.

Полная себестоимость разработки проекта рассчитывается по формуле 5.3:

$$C_{\pi} = MЗ + ЗОТ + ОСН + БГС + ПР + НР + КР + УСО \quad (5.3)$$

Где МЗ – материалы и комплектующие изделия;

ЗОТ – затраты на оплату труда;

ОСН – отчисления в Фонд социальной защиты населения;

БГС – отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний;

КР – командировочные расходы;

УСО – услуги сторонних организаций;

ПР – прочие прямые расходы;

НР – накладные расходы.

Расчет затрат по статье «Материалы и комплектующие изделия». В статью «Материалы и комплектующие изделия» включается стоимость основных и вспомогательных материалов необходимых для изготовления единицы продукции по установленным нормам.

При разработке программного обеспечения, в случае данного курсового проекта, элемент затрат материалы и комплектующие будет отсутствовать.

В статью «Затраты на оплату труда» включаются основная и дополнительная заработная плата всех работников, непосредственно занятых выполнением разработки.

Произведем расчет основной заработной платы по формуле 5.4:

$$З_0 = \sum_{i=1}^n T_{дi} \cdot \Phi_{п} \cdot K_{пр}, \quad (5.4)$$

где $T_{дi}$ – дневная тарифная ставка i -го исполнителя (денежные единицы);

$\Phi_{п}$ – плановый фонд рабочего времени (т.е. время, в течение которого работники i -ой категории принимали участие в разработке), дней;

$K_{пр}$ – коэффициент премирования за выполнение плановых показателей (1,2 – 1,4).

Дневная тарифная ставка i -го исполнителя рассчитывается следующим образом:

$$T_{дi} = T_{ми} / \Phi_{р}, \quad (5.5)$$

где $T_{ми}$ – месячная тарифная ставка i -го исполнителя, тыс.руб.;

$\Phi_{р}$ – среднемесячный фонд рабочего времени, дни.

Расчетная норма рабочего времени в днях на 2018 г. при пятидневной рабочей неделе составляет 253 дней. Тогда среднемесячный фонд рабочего времени составляет:

$$\Phi_{р} = 253/12 = 21,08 \text{ дней}$$

Месячная тарифная ставка каждого исполнителя (T_m) определяется путем умножения действующей месячной тарифной ставки первого разряда (T_{m1p}) (с 1.03.2018 составляет 34 руб. по Республике Беларусь) на тарифный коэффициент (T_k), соответствующий установленному тарифному разряду (выбирается из единой тарифной сетки для определенной категории работников, которые будут работать над проектом):

$$T_m = T_{m1p} \cdot T_k \quad (5.6)$$

Произведем расчет необходимых показателей по указанным формулам.

Месячная тарифная ставка 1-ого исполнителя соответствующего разряда (2 разряд):

$$T_{m1} = T_{m1p} \cdot T_k = 68 \cdot 2,48 = 168,64 \text{ руб.}$$

Дневная тарифная ставка 1-ого исполнителя

$$T_{d1} = T_{m1} / \Phi_p = 168,64 / 21,08 = 8 \text{ руб.}$$

Оплата за отработанное время одного работника:

$$З_{п1} = T_{d1} \cdot \Phi_n = 8 \cdot 9 = 72 \text{ руб.}$$

Основная заработная плата одного работника с учетом премии:

$$З_{пполн1} = З_{п1} \cdot K_{пр} = 72 \cdot 1,4 = 100,8 \text{ руб.}$$

Общая основная заработная плата с учетом премии будет равна $З_{пполн1}$, а именно

$$З_o = 100,8 \cdot 2 = 201,6 \text{ руб.}$$

Результаты расчетов представим в виде таблицы 5.3.

Таблица 5.3 – Расчет основной заработной платы научно-производственного персонала

| Категория работников | Кол-во чел | Тарифный коэф-т | Месячная тарифная ставка соотв. разряда, руб. | Дневная тарифная ставка, руб. | Плановый фонд рабочего времени, дней | Оплата за отработанное время одного работника, руб. | Основная заработная плата одного работника с учетом премии, руб. | Основная заработная плата с учетом премии, руб. |
|----------------------|------------|-----------------|---|-------------------------------|--------------------------------------|---|--|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Инженер программист | 1 | 2,48 | 168,64 | 8 | 9 | 72 | 100,8 | 100,8 |
| Итого | | | | | | | | 100,8 |

Дополнительная заработная плата научно-технического персонала включает выплаты, предусмотренные действующим законодательством за непроработанное время (оплата очередных и дополнительных отпусков, выплата за выслугу лет и др.) определяется по формуле:

Дополнительная заработная плата рассчитывается по формуле 5.7:

$$З_{\text{д}} = З_{\text{о}} \cdot Н_{\text{д}} / 100\%, \quad (5.7)$$

где $Н_{\text{д}}$ – норматив дополнительной заработной платы, %.

Рассчитаем дополнительную заработную плату:

$$З_{\text{д}} = З_{\text{о}} \cdot Н_{\text{д}} / 100\% = 201,6 \cdot 20\% / 100\% = 40,32 \text{ руб.}$$

В целом затраты на оплату труда составят:

$$ЗОТ = З_о + З_д = 100,8 + 40,32 = 141,12 \text{ руб.}$$

Расчет затрат по статье «Отчисления в фонд социальной защиты населения на социальные нужды» определяется в процентах от суммы основной и дополнительной заработной платы всех категорий работников, причастных к выполнению данной разработки:

$$ОСН = ЗОТ \cdot S_{осн} / 100\% \quad (5.8)$$

$S_{осн}$ – ставка отчислений на социальные нужды, % (на сегодняшний день по РБ установлена ставка 34%)

Рассчитаем ОСН:

$$ОСН = ЗОТ \cdot S_{осн} / 100\% = 141,12 \cdot 34\% / 100\% = 47,98 \text{ руб.}$$

Расчет затрат по статье «Отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний». Отчисления в Белгосстрах на страхование от несчастных случаев на производстве и профессиональных заболеваний рассчитывается по следующей формуле:

$$БГС = ЗОТ \cdot Н_{БГС} / 100\%, \quad (5.9)$$

$Н_{БГС}$ – норматив отчислений в Белгосстрах установленный для организации, % (для организаций связи норматив равен 0,6%).

Рассчитаем БГС:

$$БГС = ЗОТ \cdot Н_{БГС} / 100\% = 141,12 \cdot 0,6\% / 100\% = 8,47 \text{ руб.}$$

Все остальные затраты включаются в себестоимость единицы продукции косвенным путем в процентах от основной заработной платы.

Расчет затрат по статье «Прочие расходы». Данная статья затрат включает в себя затраты на приобретение, перевод специальной научно-технической информации, использование технических средств связи и т.д.

$$ПР = З_о \cdot Н_{пр} / 100\%, \quad (5.9)$$

где $H_{\text{пр}}$ – норматив прочих расходов для расчета себестоимости и отпускной цены НИОКР, %

Рассчитаем ПР:

$$\text{ПР} = Z_o \cdot H_{\text{пр}} / 100\% = 201,6 \cdot 10\% / 100\% = 20,16 \text{ руб.}$$

Расчет затрат по статье «Накладные расходы». Данная статья затрат в равной степени относится ко всем выполняемым НИОКР и включает в себя затраты на содержание и текущий ремонт зданий, сооружений, оборудования, инвентаря, расходы по охране труда и т.д.

$$\text{НР} = Z_o \cdot H_{\text{пр}} / 100\%, \quad (5.10)$$

где $H_{\text{пр}}$ – норматив накладных расходов для расчета себестоимости и отпускной цены НИОКР, %

Рассчитаем НР:

$$\text{НР} = Z_o \cdot H_{\text{пр}} / 100\% = 201,6 \cdot 170\% / 100\% = 342,72 \text{ руб.}$$

Расчет затрат по статье «Командировочные расходы». Командировочные расходы рассчитываются по смете, или по формуле:

$$\text{КР} = Z_o \cdot H_{\text{кр}} / 100\%, \quad (5.11)$$

где $H_{\text{кр}}$ – норматив командировочных расходов, %

Рассчитаем КР:

$$\text{КР} = Z_o \cdot H_{\text{кр}} / 100\% = 201,6 \cdot 13\% / 100\% = 26,20 \text{ руб.}$$

Расчет затрат по статье «Услуги сторонних организаций». Услуги сторонних организаций (УСО) рассчитываются исходя их сметы затрат.

В нашем случае УСО будет равно 0.

Итоговый расчет полной себестоимости разработки проекта. Таким образом, рассчитаем полную себестоимость разработки проекта:

$$C_{\pi} = \text{ЗОТ} + \text{ОСН} + \text{БГС} + \text{ПР} + \text{НР} + \text{КР} = 586,658 \text{ руб.}$$

В ходе расчетов было получено, что полная себестоимость разработки системы составляет 586,658 рубля.

Стоимость разработки такого приложения с учетом разработки приложения для ПК в частном предприятии была бы около 20000 рублей. Таким образом, на разработке было сэкономлено приблизительно 14000 рублей, что позволяет потратить эти деньги на рекламу и продвижение в сети интернет.

6 ОХРАНА ТРУДА ПРИ ЭКСПЛУАТАЦИИ И ОБСЛУЖИВАНИИ ПЭВМ И ВДТ, ОФИСНОЙ ТЕХНИКИ

6.1 Общие требования охраны труда

К работам по эксплуатации и обслуживанию ПЭВМ, ВДТ и офисной техники допускаются лица не моложе 18 лет, прошедшие медицинское освидетельствование, вводный инструктаж, первичный инструктаж, обучение и на рабочем месте, прошедшие проверку знаний требований охраны труда, имеющие группу по электробезопасности не ниже I. Персонал, обслуживающий ПЭВМ, ВДТ и офисную технику обязан:

1. Выполнять только ту работу, которая определена инструкцией по эксплуатации оборудования и должностными инструкциями.
2. Выполнять правила внутреннего трудового распорядка.
3. Правильно применять средства индивидуальной и коллективной защиты.
4. Соблюдать требования охраны труда.
5. Немедленно извещать своего непосредственного или вышестоящего руководителя о любой ситуации, угрожающей жизни и здоровью людей, о каждом несчастном случае, происшедшем на производстве, или об ухудшении состояния своего здоровья, в том числе о проявлении признаков острого профессионального заболевания (отравления).
6. Проходить обучение безопасным методам и приемам выполнения работ и оказанию первой помощи пострадавшим на производстве, инструктаж по охране труда, проверку знаний требований охраны труда.
7. Проходить обязательные периодические (в течение трудовой деятельности) медицинские осмотры (обследования), а также проходить внеочередные медицинские осмотры (обследования) по направлению работодателя в случаях, предусмотренных
8. Трудовым кодексом и иными федеральными законами.
9. Уметь оказывать первую доврачебную помощь пострадавшим от электрического тока и при других несчастных случаях.
10. Уметь применять средства первичного пожаротушения.
11. При эксплуатации и обслуживании ПЭВМ, ВДТ и офисной техники возможны воздействия следующих опасных и вредных производственных факторов:
 - повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;
 - повышенный уровень электромагнитных излучений;
 - повышенный уровень статического электричества;

- пониженная ионизация воздуха;
- зрительных анализаторов;
- недостаточная освещенность рабочего места;
- перенапряжение зрительных анализаторов;
- статические физические перегрузки.

12. В случаях травмирования или недомогания необходимо прекратить работу, известить об этом руководителя работ и обратиться в медицинское учреждение.

6.2 Требования охраны труда перед началом работы

Необходимо:

1. Подготовить рабочее место.
2. Убедиться в достаточной освещенности рабочего места, в отсутствии бликов на экране, при необходимости отрегулировать освещенность рабочего места.
3. Проверить правильность подключения оборудования, осветительных приборов местного освещения к электросети.
4. Проверить исправность проводов питания и отсутствие оголенных участков проводов.
5. Убедиться в наличии заземления системного блока, монитора, другой офисной техники.
6. Протереть антистатической салфеткой поверхность экрана монитора.
7. Проверить правильность установки стола, кресла, угла наклона экрана, положение клавиатуры. При необходимости произвести регулировку элементов компьютера, рабочего стола и кресла в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.
8. О всех недостатках и неисправностях оборудования, обнаруженных при осмотре, доложить непосредственному руководителю для принятия мер к их устранению.

6.3 Требования охраны труда во время работы

Необходимо:

1. Все видеодисплейные терминалы должны иметь гигиенический сертификат.
2. Помещения с ВДТ и ПЭВМ должны иметь естественное и искусственное освещение.

3. Конструкция ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана ВДТ. Дизайн ПЭВМ должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПЭВМ, клавиатура и другие блоки и устройства ПЭВМ должны иметь матовую поверхность с коэффициентом отражения 0,4 - 0,6 и не иметь блестящих деталей, способных создавать блики.

4. Конструкция ВДТ должна предусматривать регулирование яркости и контрастности.

5. Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электронно-лучевой трубки (ЭЛТ) должна составлять не менее 6 м², в помещениях культурно-развлекательных учреждений и с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 м²

6. При использовании ПЭВМ с ВДТ на базе ЭЛТ (без вспомогательных устройств - принтер, сканер и др.), отвечающих требованиям международных стандартов безопасности компьютеров, с продолжительностью работы менее 4-х часов в день допускается минимальная площадь 4,5 м² на одно рабочее место пользователя

7. Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

8. Рабочие места с ПЭВМ должны размещаться таким образом, чтобы расстояние от экрана одного видеомонитора до тыла другого было не менее 2м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2м.

9. Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

10. Оконные проемы в помещениях, где используются ПЭВМ, должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

11. Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

12. Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

13. Рабочая мебель для пользователей ПЭВМ должна отвечать следующим требованиям:

- рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм;

- рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию;

- рабочее место должно быть оборудовано подставкой для ног, имеющей ширину не менее 300мм, глубину не менее 400мм, регулировку по высоте в пределах до 150мм и по углу наклона опорной поверхности подставки до 20о; поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10мм;

- клавиатуру следует располагать на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

- в помещениях, оборудованных ПЭВМ, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

14. Женщины со времени установления беременности переводятся на работы, не связанные с использованием ПЭВМ, или для них ограничивается время работы с ПЭВМ (не более 3-х часов за рабочую смену).

15. Не допускается размещать оборудование ЭВМ и создавать рабочие места с ВДТ и ПЭВМ в подвальных помещениях. В случаях производственной необходимости, эксплуатация ВДТ и ПЭВМ в помещениях без естественного освещения допускается только при соответствующем обосновании и наличии положительного санитарно-эпидемиологического заключения, выданного в установленном порядке.

16. Не допускается размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, технологического оборудования, создающего помехи в работе ПЭВМ.

17. Уровни шума на рабочих местах с ПЭВМ не должны превышать допустимых значений. Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

18. Работник, занятый эксплуатацией и обслуживанием ПЭВМ, ВДТ и другой офисной техники должен соблюдать инструкции по охране труда, инструкции завода-изготовителя на эксплуатируемое оборудование.

19. Текущий ремонт ПЭВМ, ВДТ и другой офисной техники должен производить специально подготовленный персонал, имеющий группу по электробезопасности не ниже III, на специально оборудованном рабочем месте.

20. Все работы, связанные с текущим ремонтом, необходимо выполнять исправным инструментом с изолированными ручками.

21. При работе с коммутационными переносными шнурами необходимо брать за изолированные части штепселя шнура.

22. Работы по чистке офисного оборудования производить только после отключения его от электросети.

23. Чистку производить этиловым ректифицированным спиртом согласно инструкции по обслуживанию. Обязательно мыть руки теплой водой с мылом после каждой чистки оборудования.

6.4 Требования охраны труда в аварийных ситуациях

Необходимо:

1. При возникновении аварий и ситуаций, которые могут привести к авариям и несчастным случаям, необходимо:

2. Немедленно прекратить работы и известить руководителя работ.

3. Под руководством ответственного за производство работ оперативно принять меры по устранению причин аварий или ситуаций, которые могут привести к авариям или несчастным случаям.

4. При возникновении пожара, задымлении:

5. Немедленно сообщить по телефону «01» в пожарную охрану, оповестить работающих, поставить в известность руководителя подразделения, сообщить о возгорании на пост охраны.

6. Открыть запасные выходы из здания, обесточить электропитание, закрыть окна и двери.

7. Приступить к тушению пожара первичными средствами пожаротушения, если это не сопряжено с риском для жизни.

8. Организовать встречу пожарной команды.

9. Покинуть здание и находиться в зоне эвакуации.

10. При несчастных случаях:

10.1 Немедленно организовать первую помощь пострадавшему и при необходимости доставку его в медицинскую организацию.

10.2 Принять неотложные меры по предотвращению развития аварийной или иной чрезвычайной обстановки и воздействия травмирующих факторов на других лиц.

11. Сохранить до начала расследования обстановку, какой она была на момент происшествия, если это не угрожает жизни и здоровью других лиц и не ведет к катастрофе, аварии или возникновению чрезвычайных обстоятельств, а в случае невозможности ее сохранения зафиксировать сложившуюся обстановку (составить схемы, провести другие мероприятия).

6.5 Требования охраны труда по окончании работы

Необходимо:

1. Произвести закрытие всех активных задач;
2. Убедиться, что в дисководов отсутствуют диски;
3. Выключить питание системного блока, всех периферийных устройств;
4. Отключить блок питания;
5. Привести в порядок рабочее место;
6. Выполнить упражнения для глаз и пальцев рук на расслабление;
7. Тщательно вымыть руки теплой водой с мылом;
8. Сообщить лицу, ответственному за производство работ, о всех недостатках, замеченных во время работы, и принятых мерах по их устранению.

ЗАКЛЮЧЕНИЕ

В результате данной дипломной работы было разработано web-приложение для профилактики профессиональных заболеваний и приложение, реализующее клиентскую часть, для ПК. Была рассчитана стоимость разработки приложения и выявлено, что разработка такого приложения частными предприятиями стоила бы намного дороже.

Приложения написаны на самых актуальных технологиях, это значит, что их будет легко поддерживать. Для архитектуры приложения были применены оптимальные паттерны, что открывает возможности для расширения функциональных возможностей приложения без высоких затрат.

Разработанное приложение предоставляет возможности для пользователя повысить качество своего отдыха и проведения рабочего времени. Были решены проблемы аналогов. Проблема низкой персонализации была решена путем добавления системы профессий и возможности настраивать удобное время для профилактической разминки. Проблема узкой специализации была решена таким же образом. Необходимо признать, что узкоспециализированные решение решают конкретные задачи лучше, но комплексные решения имеют большие шансы на успех и популярность в связи с большим охватом аудитории. Так же была решена проблема плохого дизайна. Для дизайна в обоих приложениях были реализованы концепции Google Material Design.

Хорошим улучшением для системы могло стать введение нейронных сетей для анализа упражнений, подходящих для конкретного пользователя, тем самым еще более улучшить опыт работы пользователя с приложением.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статьи, руководства и примеры на языке программирования C# по созданию веб-сервисов и web-приложений на платформе ASP.NET, MVC и Web API. [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/mvc.php>
2. Vue.js — Прогрессивный JavaScript-фреймворк. [Электронный ресурс]. – Режим доступа: <https://ru.vuejs.org/index.html>.
3. Visual Studio IDE, редактор кода, Team Services и Mobile Center. [Электронный ресурс]. – Режим доступа: <https://www.visualstudio.com/ru/?rr=https%3A%2F%2Fwww.google.by%2F>.
4. Справочник по C#. [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/>.
5. Справочник по C# 7.0. [Электронный ресурс]. – Режим доступа: <http://tehnokom.com.ua/catalog/clj.php>.
6. Справочник JavaScript. [Электронный ресурс]. – Режим доступа: <https://javascript.ru/manual>.
7. Electron. Создавайте кроссплатформенные приложения при помощи JavaScript, HTML и CSS. [Электронный ресурс]. – Режим доступа: <https://electronjs.org/>.
8. Краткое руководство (платформа Entity Framework) [Электронный ресурс]. – Режим доступа: [https://msdn.microsoft.com/ru-ru/library/bb399182\(v=vs.100\).aspx](https://msdn.microsoft.com/ru-ru/library/bb399182(v=vs.100).aspx).