Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка программного модуля**

**«Система учета спортивных тренировок»**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Выполнил студент группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Корчагин Д.К.**

**317ИС-22**

**Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Назаров В.Ю.**

**Москва 2024**

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 
   1. Введение в предметную область
   2. Обзор и анализ существующих программных решений
2. **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ**
   1. Назначение системы
   2. Требования к функциям в системе
3. **РЕАЛИЗАЦИЯ ПРОГРАММНОГО МОДУЛЯ**
   1. Описание среды разработки
   2. Обоснование выбора инструментов при разработке
   3. Реализация пользовательского интерфейса

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

**ПРИЛОЖЕНИЕ**

**Введение**

В современном мире, где здоровый образ жизни и физическая активность становятся все более популярными, эффективное управление спортивными тренировками является ключевым фактором достижения спортивных результатов и поддержания физической формы. Одной из важнейших задач, стоящих перед спортсменами, тренерами и любителями фитнеса, является сбор, хранение, анализ и интерпретация данных, получаемых в процессе тренировок. Эти данные позволяют не только контролировать текущие результаты, но и прогнозировать будущие достижения, корректировать программы тренировок и оптимизировать их эффективность.

В связи с этим, актуальность разработки программного модуля «Система учета спортивных тренировок» не вызывает сомнений. Данный модуль позволит автоматизировать процесс сбора и хранения данных о тренировках, обеспечит удобный и наглядный анализ прогресса, а также предоставит возможность создания индивидуальных программ тренировок с учетом индивидуальных особенностей каждого пользователя.

В ходе работы будут рассмотрены существующие решения для учета спортивных тренировок, проанализированы их преимущества и недостатки. Будет разработана структура программного модуля, определены его основные функции и интерфейс. В результате будет создан работающий прототип программного модуля, который позволит продемонстрировать его функциональность и эффективность.

**Актуальность курсовой работы**

В современном мире, где стремление к здоровому образу жизни и физической активности становится все более массовым явлением, эффективное управление спортивными тренировками играет ключевую роль в достижении спортивных результатов и поддержании физической формы. Однако, несмотря на растущую популярность фитнеса и спорта, многие люди сталкиваются с проблемами, связанными с отсутствием систематизации и анализа своих тренировок. Это приводит к неэффективному расходованию времени и сил, отсутствию мотивации и, как следствие, к снижению результативности.

Таким образом, разработка программного модуля «Система учета спортивных тренировок» является актуальной задачей, которая позволит решить проблему неэффективного управления тренировками, повысить мотивацию и результативность спортсменов и любителей фитнеса, а также предоставить удобный и доступный инструмент для систематизации и анализа данных о тренировках.

**Проблематика курсовой работы**

При разработке программы нужно учитывать, что предстоит работать с большими объемами данных, ведь в наше время спорт очень популяризирован и в программу для учета тренировок придется добавить много разных функций, от авторизации и до таблиц с тренировками. Система должна эффективно использовать полученные данные.

Также стоит обратить внимание и на то, что в базе данных приложения будут храниться конфиденциальные данные. Такие как: возраст, ФИО, логины, пароли и тому подобное. Поэтому стоит учесть что данные должны надежно храниться и быть защищенными от утечек и взлома.

Не мало важно сделать удобный, интуитивный интерфейс, чтобы пользователь мог без проблем использовать все функции программы. Удобные приложение сильно влияют на количество пользователей и на количество спортсменов в целом.

**Предмет исследования:** приложения для учета спортивных тренировок.

**Объект исследования:** спорткомплекс.

**Цели и задачи курсовой работы**

**Цель курсовой работы:**

Разработка программного модуля «Система учета спортивных тренировок», обеспечивающего удобный и эффективный инструмент для сбора, хранения, анализа и планирования тренировочных данных.

**Задачи курсовой работы:**

1. Анализ существующих решений
2. Проектирование программного модуля
3. Реализация программного модуля
4. Тестирование и отладка
5. Разработка документации

Выбор методов исследования определяется необходимостью и возможностью решить поставленные задачи.

Исследовательская работа состоит из введения, трех глав, заключения, списка источников и литературы. Во введении расписаны актуальность, проблематика, определены предмет и объект исследования, цели и задачи, методологические принципы.

При написании курсовой работы были использованы различные источники информации: учебно-методическая литература и интернет-ресурсы.

1. **Анализ предметной области**
   1. **Введение в предметную область**

В современном мире, где стремление к здоровому образу жизни и физической активности становится все более массовым явлением, спорт играет ключевую роль в поддержании физического и психического здоровья. Регулярные занятия спортом способствуют укреплению иммунной системы, улучшению работы сердечно-сосудистой системы, повышению выносливости и силы, а также снижению риска развития многих заболеваний. Кроме того, спорт является эффективным средством борьбы со стрессом и депрессией, способствует улучшению настроения и повышению самооценки.

Тренировки являются основным инструментом достижения спортивных результатов и поддержания физической формы. Они представляют собой целенаправленные физические упражнения, направленные на развитие определенных физических качеств, таких как сила, выносливость, гибкость, координация и скорость. Эффективность тренировок во многом зависит от правильного планирования, систематичности и контроля прогресса.

История спорта насчитывает тысячелетия, начиная с древних Олимпийских игр в Греции. Однако, только в XX веке спорт стал массовым явлением, благодаря развитию медицины, спортивной науки и технологий. В последние десятилетия наблюдается стремительный рост интереса к спорту и фитнесу, что привело к появлению огромного количества спортивных клубов, фитнес-центров, тренажерных залов и других объектов спортивной инфраструктуры.

Развитие информационных технологий оказало значительное влияние на спорт и фитнес-индустрию. С появлением носимых устройств, таких как фитнес-браслеты и умные часы, стало возможным отслеживать различные показатели во время тренировок, такие как пульс, количество шагов, пройденное расстояние и сожженные калории. Кроме того, мобильные приложения и онлайн-платформы позволяют спортсменам и любителям фитнеса планировать свои тренировки, отслеживать прогресс и получать мотивацию.

Несмотря на важность тренировок, многие спортсмены и любители фитнеса сталкиваются с проблемами, связанными с управлением своим тренировочным процессом. К основным проблемам относятся:

* Отсутствие систематизации данных: Многие спортсмены ведут записи о тренировках в тетрадях, блокнотах или просто запоминают их. Это приводит к потере важной информации, сложности в поиске и анализе данных.
* Трудности в анализе прогресса: Оценка прогресса часто основывается на субъективных ощущениях, что не всегда объективно отражает реальное состояние. Отсутствие наглядности в представлении данных затрудняет выявление тенденций и закономерностей.
* Неэффективное планирование тренировок: Многие стандартные программы тренировок не учитывают индивидуальные особенности спортсмена, такие как уровень подготовки, цели, возраст, пол и т.д. Отсутствие систематизации и анализа данных может привести к снижению мотивации и, как следствие, к срыву тренировочного процесса.

Информационные технологии открывают новые возможности для повышения эффективности тренировочного процесса. Программные модули для учета спортивных тренировок позволяют автоматизировать процесс сбора, хранения и анализа данных, обеспечивают удобный и наглядный интерфейс для отслеживания прогресса, а также предоставляют возможность создания индивидуальных программ тренировок с учетом индивидуальных особенностей каждого пользователя.

* 1. **Обзор и анализ существующих программных решений**

В настоящее время на рынке представлено множество программных решений для учета спортивных тренировок, которые различаются по функционалу, целевой аудитории, ценовой политике и другим параметрам. Давайте рассмотрим наиболее популярные и востребованные решения, а также проведен их анализ.

**Strava** - одно из самых популярных приложений для бега, велоспорта и других видов активности. Оно позволяет отслеживать пробежки, велотренировки, плавание и другие виды спорта с помощью GPS, анализировать прогресс, ставить цели и сравнивать результаты с другими пользователями.

Преимущества: большое сообщество пользователей, разнообразие видов спорта, интеграция с другими приложениями и устройствами.

Недостатки: ограниченный функционал для других видов спорта, высокая стоимость премиум-подписки.

**Endomondo** - приложение для отслеживания физической активности с помощью GPS. Оно поддерживает широкий спектр видов спорта, включая бег, велосипед, плавание, ходьбу и многое другое.

Преимущества: разнообразие видов спорта, интеграция с другими приложениями и устройствами, возможность создания индивидуальных тренировочных планов.

Недостатки: не самый удобный интерфейс, недостаточная наглядность анализа данных.

**MyFitnessPal** - приложение для отслеживания питания и физической активности. Оно позволяет вести дневник питания, отслеживать калории, микронутриенты и микронутриенты, а также анализировать прогресс в достижении фитнес-целей.

Преимущества: большая база продуктов и блюд, интеграция с другими приложениями и устройствами, возможность создания индивидуальных планов питания.

Недостатки: основной упор на питание, ограниченный функционал для отслеживания тренировок.

**TrainingPeaks - п**рофессиональное приложение для планирования и анализа тренировок для спортсменов и тренеров. Оно позволяет создавать индивидуальные тренировочные планы, отслеживать прогресс, анализировать данные о тренировках и питании, а также общаться с тренером.

**Преимущества:** широкий функционал, возможность работы с тренером, интеграция с другими приложениями и устройствами.

**Недостатки:** высокая стоимость, сложность освоения для новичков.

**Final Surge – еще** один профессиональный инструмент для планирования и анализа тренировок. Он предоставляет широкий спектр функций для спортсменов и тренеров, включая создание индивидуальных планов, отслеживание прогресса, анализ данных и многое другое.

**Преимущества:** широкий функционал, возможность работы с тренером, интеграция с другими приложениями и устройствами.

**Недостатки:** высокая стоимость, сложность освоения для новичков.

1. **Проектирование программного модуля**
   1. **Назначение системы**

Программный модуль “Система учета спортивных тренировок” предназначен администраторов и спортсменов. В нем реализована автоматизация процессов обработки, хранения, сбора и анализа данных, функции для взаимодействия пользователя с администратором и наоборот. Функции непосредственно для тренировок. Давайте рассмотрим основные назначения системы в нескольких пунктах:

**1. Управление со стороны администратора**

Администратор имеет доступ к следующим функциям:

**1.1. Профили пользователей**

* **Просмотр профилей**: Администратор может просматривать профили всех зарегистрированных пользователей, включая их личную информацию (фамилию, имя, отчество, возраст).

**1.2. Тренировки**

* **Управление тренировками**: Администратор может добавлять, удалять и редактировать тренировки на каждый день месяца.
* **Просмотр расписания**: Администратор может просматривать расписание тренировок на выбранный день, включая информацию о виде спорта, времени и тренере.

**1.3. Отзывы**

* **Просмотр отзывов**: Администратор может просматривать отзывы, оставленные пользователями, включая оценку и текст отзыва.

**2. Взаимодействие с пользователями**

Пользователи могут выполнять следующие действия:

**2.1. Регистрация**

* **Создание аккаунта**: Пользователи могут зарегистрироваться в системе, указав свои личные данные (фамилию, имя, отчество, возраст) и учетные данные (логин и пароль).

**2.2. Вход в систему**

* **Вход как пользователь**: Зарегистрированные пользователи могут войти в систему, используя свои логин и пароль.

**2.3. Мой профиль**

* **Просмотр профиля**: Пользователи могут просматривать свою личную информацию (фамилию, имя, отчество, возраст).

**2.4. Тренировки**

* **Просмотр расписания**: Пользователи могут просматривать расписание тренировок на выбранный день, включая информацию о виде спорта, времени и тренере.
* **Запись на тренировку**: Пользователи могут записываться на выбранные тренировки, используя кнопку "Выбрать тренировку".

**2.5. Мои тренировки**

* **Просмотр записанных тренировок**: Пользователи могут просматривать список тренировок, на которые они записаны, отсортированный по дню месяца и времени.
* **Удаление тренировки**: Пользователи могут удалять свои записи на тренировки.

**2.6. Оставить отзыв**

* **Отправка отзыва**: Пользователи могут оставлять отзывы о спортивном комплексе, выбрав оценку и написав текст отзыва. Отзывы отправляются администратору для просмотра.
  1. **Требования к функциям в системе**

В процессе исследования предметной области были выявлены следующие требования в системе:

* Регистрация.
* Авторизация.
* Просмотр профиля.
* Учет тренировок, их изменение, удаление и добавление.
* Возможность оставлять и читать отзывы.
* Возможность добавлять тренировки в раздел с сохраненными тренировками.

Проведем анализ среды разработки:

Таблица “Регистрация”. Она формируется из данных, вводимых пользователем при регистрации. Содержит в себе: id пользователя, логин пользователя, пароль пользователя, ФИО пользователя, возраст пользователя.

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id\_user | INTEGER PRIMARY KEY | Уникальный идентификатор пользователя |
| login\_user | TEXT | Логин пользователя |
| password\_user | TEXT | Пароль пользователя |
| first\_name | TEXT | Имя пользователя |
| last\_name | TEXT | Фамилия пользователя |
| middle\_name | TEXT | Отчество пользователя |
| age | INTEGER | Возраст пользователя |

Таблица “Отзывы”. Содержит данные об отзывах пользователей. В ней содержатся такие данные как: id отзыва, id пользователя, который отставил отзыв, оценка пользователя, содержание отзыва.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id\_review | INTEGER PRIMARY KEY | Уникальный идентификатор отзыва |
| user\_id | INTEGER PRIMARY KEY | Идентификатор пользователя, оставившего отзыв |
| rating | INTEGER | Оценка отзыва (от 1 до 5) |
| review | TEXT | Текст отзыва |

Таблица “Тренировки”. Содержит в себе данные о тренировках. В ней содержатся такие данные как: id тренировки, вид спорта, время тренировки, ФИО тренера, номер дня месяца.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id\_training | INTEGER PRIMARY KEY | Уникальный идентификатор тренировки |
| sport | TEXT | Вид спорта |
| time | TEXT | Время тренировки |
| сoach | TEXT | ФИО тренера |
| day | INTEGER | День месяца (от 1 до 31) |

Таблица “Мои тренировки”. Содержит в себе такие данные как: id тренировки пользователя, id пользователя, записавшегося на тренировку, id тренировки, номер дня месяца.

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id\_user\_training | INTEGER PRIMARY KEY | Уникальный идентификатор записи на тренировку |
| user\_id | UNTEGER | Идентификатор пользователя, записавшегося на тренировку |
| training\_id | INTEGER | Идентификатор тренировки |
| day | INTEGER | День месяца (от 1 до 31) |

1. **РЕАЛИЗАЦИЯ ПРОГРАММНОГО МОДУЛЯ**
   1. **Описание среды разработки**

Для создания программного модуля системы "Система учета спортивных тренировок" использовались следующие программные инструменты:

* Язык программирования Python
* Среда разработки PyCharm
* Библиотека PyQt6
* База данных SQLite3

Выбранные инструменты обеспечивает необходимый уровень функциональности для разработки консольного приложения.

* 1. **Обоснование выбора инструментов при разработке**

Язык программирования Python - простой и мощный язык программирования с интуитивно понятным синтаксисом, идеально подходящий для быстрой разработки и поддержки приложений.

Преимущества:

* Простота и читаемость: Python известен своей простотой и легкостью в изучении. Его синтаксис интуитивно понятен, что ускоряет разработку и облегчает поддержку кода.
* Большое сообщество: Python имеет активное и большое сообщество разработчиков, что обеспечивает доступ к широкому спектру библиотек, фреймворков и документации.
* Многофункциональность: Python подходит для различных задач, включая веб-разработку, научные вычисления, анализ данных и многое другое.
* Кроссплатформенность: Python работает на различных операционных системах, что обеспечивает переносимость приложения.

IDE PyCharm - мощная интегрированная среда разработки (IDE) для Python, предлагающая широкий набор инструментов для эффективной разработки, отладки и рефакторинга кода.

Преимущества:

* Интегрированная среда разработки (IDE): PyCharm предоставляет мощную и удобную среду для разработки на Python, включая поддержку автодополнения кода, отладки, рефакторинга и интеграции с системами контроля версий.
* Поддержка множества фреймворков: PyCharm поддерживает широкий спектр фреймворков и библиотек, включая Django, Flask, PyQt и другие.
* Интеграция с инструментами анализа кода: PyCharm включает встроенные инструменты для анализа кода, что помогает обнаруживать ошибки и улучшать качество кода.
* Кроссплатформенность: PyCharm доступен для Windows, macOS и Linux.

Библиотека PyQt6 - мощная библиотека для создания графических интерфейсов пользователя (GUI) на Python, обеспечивающая богатый набор инструментов для разработки сложных и интерактивных приложений.

Преимущества:

* Мощный инструмент для создания GUI: PyQt6 предоставляет широкий набор инструментов для создания сложных и интерактивных графических интерфейсов пользователя.
* Кроссплатформенность: PyQt6 работает на Windows, macOS и Linux, что обеспечивает единообразный интерфейс на разных платформах.
* Интеграция с Python: PyQt6 хорошо интегрируется с Python, что позволяет использовать все преимущества языка в разработке GUI.
* Большое сообщество и документация: PyQt имеет активное сообщество и обширную документацию, что облегчает разработку и решение проблем.

База данных SQLite3 - легковесная и простая в использовании реляционная база данных, не требующая отдельного сервера и идеально подходящая для небольших и средних проектов.

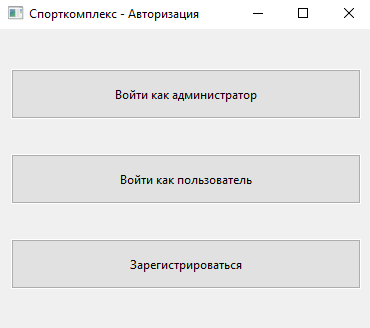
Преимущества:

* Легковесность: SQLite3 — это легковесная база данных, которая не требует отдельного сервера и может быть легко интегрирована в приложение.
* Простота использования: SQLite3 предоставляет простой и понятный интерфейс для работы с базами данных, что упрощает разработку и поддержку.
* Кроссплатформенность: SQLite3 работает на всех основных операционных системах.
* Отсутствие настройки: SQLite3 не требует настройки и может быть использован сразу после установки.

**3.3 Реализация пользовательского интерфейса**

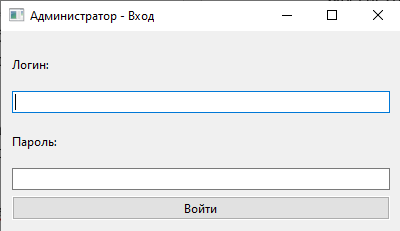
**Окно авторизации**

При запуске программы перед администратором открывается окно с тремя кнопками: “Войти как администратор”, “Войти как пользователь”, “Зарегистрироваться”.



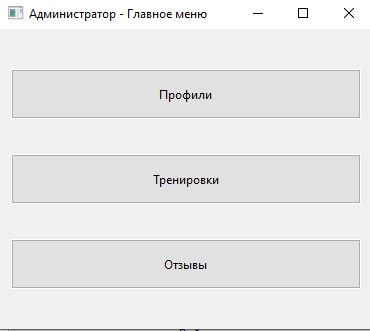
**Авторизация администратора**

Нажав на кнопку “Войти как администратор”, перед пользователем открывается окно. Окно авторизации администратора.



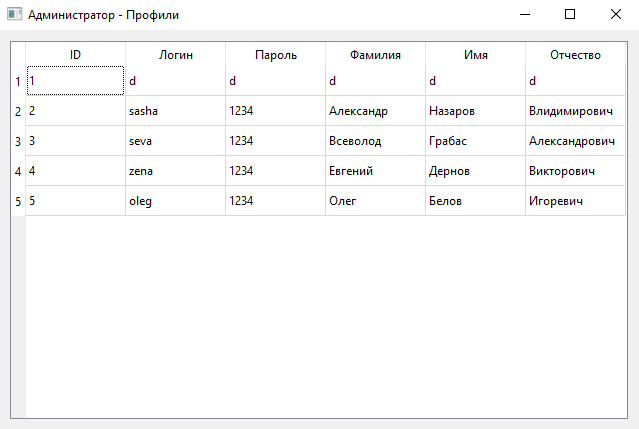
**Меню администратора**

Введя логин (admin) и пароль (1234) перед администратором открывается окно с тремя кнопками: “Профили”, “Тренировки”, “Отзывы”.



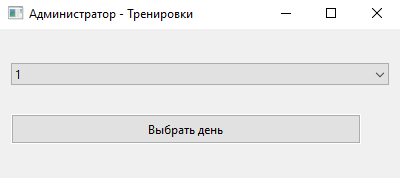
**Профили:**

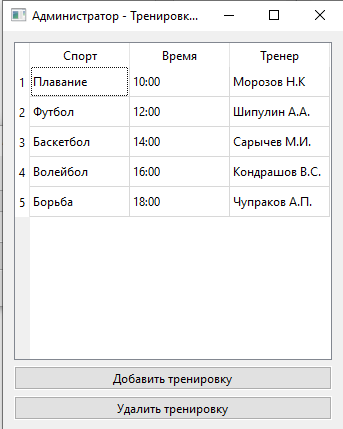
Нажав на кнопку “Профили”, перед администратором открывается окно с данными всех зарегистрированных пользователей.



**Тренировки:**

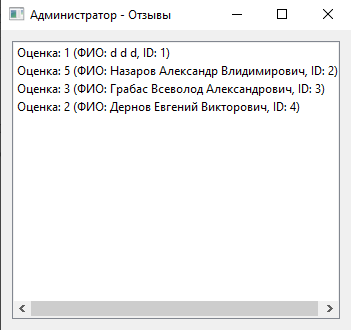
Нажав на кнопку “Тренировки”, перед администратором открывается окно, где он может выбрать день из месяца, а котором хочет посмотреть расписание тренировок. В таблице с тренировками отображаются: вид спорта. Время тренировки, тренер. Также администратор может добавлять и удалять тренировки.





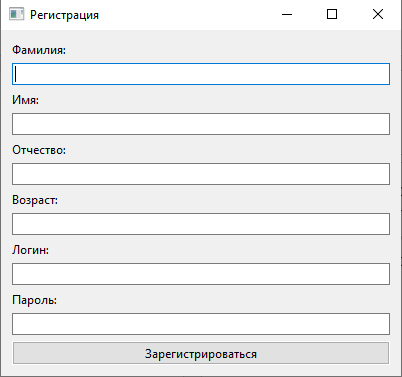
**Отзывы**

Нажав на кнопку “Отзывы”, перед администратором открывается окно с отзывами пользователей. Администратор видит оценку, ФИО отправителя и его ID.



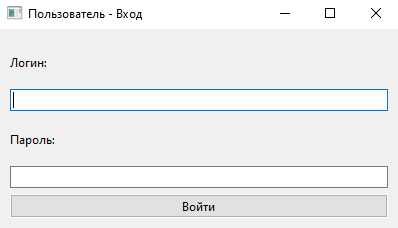
**Регистрация**

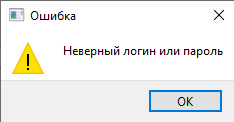
Нажав на кнопку “Зарегистрироваться”, перед пользователем открывается окно регистрации. Там он вводит свои данные: ФИО, возраст, логин и пароль. После регистрации окно закрывается и пользователя возвращает в окно авторизации.



**Вход пользователя**

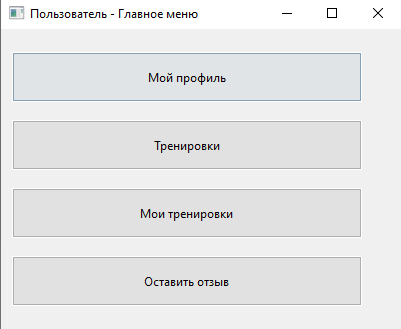
Если пользователь нажал кнопку “Войти как пользователь”, перед ним открывается окно авторизации пользователя. Там он вводит свои логин и пароль. Если такого логина нет или пароль не подходит к логину, выскакивает сообщение об ошибке.





**Меню пользователя**

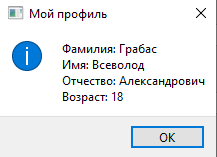
После ввода логина и пароля пользователя переносит в окно пользователя, где ему на выбор даются 4 кнопки: “Мой профиль”, “Тренировки”, “Мои тренировки”, “Оставить отзыв”.



**Мой профиль**

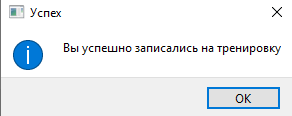
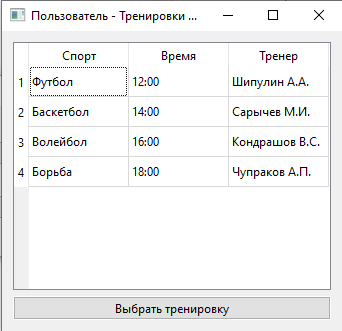
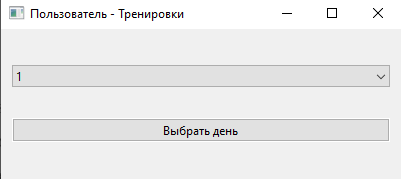
Выбрав кнопку “Мой профиль”, перед пользователем открывается окно

с данными его профиля: ФИО и возраст.



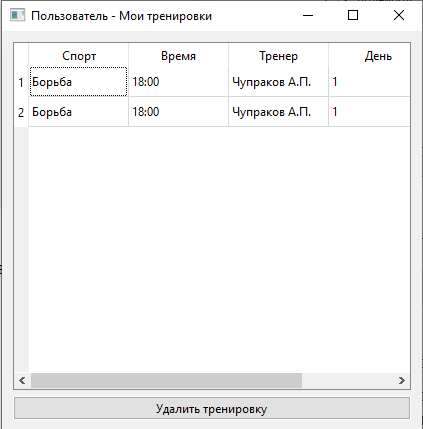
**Тренировки**

Выбрав кнопку “Тренировки”, перед пользователем открывается окно с выбором дня месяца. После выбора дня открывается окно с таблицей тренировок. Ниже расположена кнопка “Выбрать тренировку”. Выбранная тренировка добавляется с список тренировок пользователя.



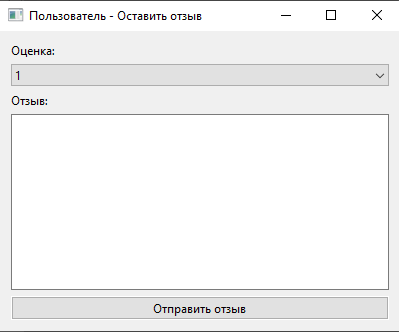
**Мои тренировки**

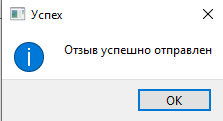
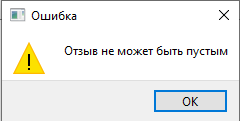
Нажав на кнопку “Мои тренировки”, перед польваотелем открывается окно с добавленными им тренировками. Они располагаются в виде таблицы. Также пользователь может удалить тренировку из этого списка.



**Мои отзывы**

Выбрав кнопку “Мои отзывы”, перед пользователем открывается окно, к котором он сможет написать отзыв о тренировке и отравить администратору. Если отзыв будет пустым, выскачет сообщение об ошибке. Если все в порядке, сообщение о том что отзыв успешно отправлен.



**Заключение**

В ходе написания курсовой работы я разработал программу для учета тренировок.

**Результаты работы**

В ходе выполнения курсовой работы мною был разработан программный модуль «Система учета спортивных тренировок», который предназначен для автоматизации процесса регистрации, анализа и отслеживания спортивных тренировок. Основной целью проекта было создание удобного и эффективного инструмента для пользователей и администраторов.

В процессе разработки мною были решены следующие задачи:

1. **Анализ требований**: Проведено исследование потребностей пользователей, определены основные функциональные требования к системе.
2. **Проектирование архитектуры**: Разработана архитектура программного модуля, включающая в себя базу данных для хранения информации о тренировках и пользователях , а также интерфейс пользователя для взаимодействия с системой.
3. **Реализация функционала**: Созданы основные компоненты системы, такие как модуль регистрации и авторизации пользователей, модуль ввода данных о тренировках, модуль анализа и отчетности.
4. **Тестирование и отладка**: Проведено тестирование системы, выявлены и устранены ошибки, обеспечена стабильная работа программного модуля.
5. **Оформление документации**: Разработана документация для пользователей.

**СПИСОК ИСПОЛЬЗУЕМЫX ИСТОЧНИКОВ**

Нормативно-правовые источники:

1. ГОСТ 7.32–2001. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.
2. ГОСТ Р 7.0.5–2008. Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу библиографическая ссылка.
3. ГОСТ 7.1–2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.
4. ГОСТ 7.80–2000. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления.

**Интернет-ресурсы:**

1. PyQt6 — полное руководство для новичков - <https://habr.com/ru/companies/skillfactory/articles/599599/>
2. Официальный сайт PyQt6 - <https://www.riverbankcomputing.com/static/Docs/PyQt6/>
3. Google картинки

import sys  
import sqlite3  
from PyQt6.QtWidgets import QApplication, QWidget, QVBoxLayout, QLabel, QLineEdit, QPushButton, QMessageBox, QTableWidget, QTableWidgetItem, QComboBox, QTextEdit, QInputDialog, QListWidget, QListWidgetItem  
from PyQt6.QtGui import QPalette, QColor  
from PyQt6.QtCore import Qt  
  
# Подключение к базе данных SQLite3  
conn = sqlite3.connect('sportcomplex.db')  
cursor = conn.cursor()  
  
# Создание таблиц в базе данных  
cursor.execute('''CREATE TABLE IF NOT EXISTS users (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 login TEXT UNIQUE,  
 password TEXT,  
 first\_name TEXT,  
 last\_name TEXT,  
 middle\_name TEXT,  
 age INTEGER)''')  
  
cursor.execute('''CREATE TABLE IF NOT EXISTS reviews (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 user\_id INTEGER,  
 rating INTEGER,  
 review TEXT,  
 FOREIGN KEY (user\_id) REFERENCES users(id))''')  
  
cursor.execute('''CREATE TABLE IF NOT EXISTS trainings (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 sport TEXT,  
 time TEXT,  
 coach TEXT,  
 day INTEGER)''')  
  
cursor.execute('''CREATE TABLE IF NOT EXISTS user\_trainings (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 user\_id INTEGER,  
 training\_id INTEGER,  
 day INTEGER,  
 FOREIGN KEY (user\_id) REFERENCES users(id),  
 FOREIGN KEY (training\_id) REFERENCES trainings(id))''')  
  
conn.commit()  
  
# Функция для инициализации начальных данных о тренировках  
def initialize\_trainings():  
 cursor.execute('SELECT COUNT(\*) FROM trainings')  
 count = cursor.fetchone()[0]  
 if count == 0:  
 trainings = [  
 ('Плавание', '10:00', 'Морозов Н.К', 1),  
 ('Футбол', '12:00', 'Шипулин А.А.', 1),  
 ('Баскетбол', '14:00', 'Сарычев М.И.', 1),  
 ('Волейбол', '16:00', 'Кондрашов В.С.', 1),  
 ('Борьба', '18:00', 'Чупраков А.П.', 1)  
 ]  
 cursor.executemany('INSERT INTO trainings (sport, time, coach, day) VALUES (?, ?, ?, ?)', trainings)  
 conn.commit()  
  
# Вызов функции для инициализации начальных данных  
initialize\_trainings()  
  
class HoverButton(QPushButton):  
 def \_\_init\_\_(self, text, parent=None):  
 super().\_\_init\_\_(text, parent)  
 self.setMouseTracking(True) # Включаем отслеживание мыши  
 self.setAutoFillBackground(True) # Включаем автоматическое заполнение фона  
 self.default\_color = QColor(255, 255, 255) # Цвет по умолчанию  
 self.hover\_color = QColor(200, 200, 255) # Цвет при наведении  
 self.pressed\_color = QColor(150, 150, 255) # Цвет при нажатии  
 self.update\_color(self.default\_color)  
  
 def enterEvent(self, event):  
 self.update\_color(self.hover\_color)  
  
 def leaveEvent(self, event):  
 self.update\_color(self.default\_color)  
  
 def mousePressEvent(self, event):  
 if event.button() == Qt.MouseButton.LeftButton:  
 self.update\_color(self.pressed\_color)  
 super().mousePressEvent(event)  
  
 def mouseReleaseEvent(self, event):  
 if event.button() == Qt.MouseButton.LeftButton:  
 self.update\_color(self.hover\_color)  
 super().mouseReleaseEvent(event)  
  
 def update\_color(self, color):  
 palette = self.palette()  
 palette.setColor(QPalette.ColorRole.Button, color)  
 self.setPalette(palette)  
  
class LoginWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(350, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Спорткомплекс - Авторизация')  
 layout = QVBoxLayout()  
  
 self.btn\_admin = HoverButton('Войти как администратор')  
 self.btn\_user = HoverButton('Войти как пользователь')  
 self.btn\_register = HoverButton('Зарегистрироваться')  
 self.btn\_admin.setFixedSize(350, 50)  
 self.btn\_user.setFixedSize(350, 50)  
 self.btn\_register.setFixedSize(350, 50)  
  
 self.btn\_admin.clicked.connect(self.openAdminLogin)  
 self.btn\_user.clicked.connect(self.openUserLogin)  
 self.btn\_register.clicked.connect(self.openRegister)  
  
 layout.addWidget(self.btn\_admin)  
 layout.addWidget(self.btn\_user)  
 layout.addWidget(self.btn\_register)  
  
 self.setLayout(layout)  
  
 def openAdminLogin(self):  
 self.admin\_login\_window = AdminLoginWindow()  
 self.admin\_login\_window.show()  
  
 def openUserLogin(self):  
 self.user\_login\_window = UserLoginWindow()  
 self.user\_login\_window.show()  
  
 def openRegister(self):  
 self.register\_window = RegisterWindow()  
 self.register\_window.show()  
  
class AdminLoginWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(400, 200) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Администратор - Вход')  
 layout = QVBoxLayout()  
  
 self.lbl\_login = QLabel('Логин:')  
 self.txt\_login = QLineEdit()  
 self.lbl\_password = QLabel('Пароль:')  
 self.txt\_password = QLineEdit()  
 self.txt\_password.setEchoMode(QLineEdit.EchoMode.Password)  
 self.btn\_login = HoverButton('Войти')  
  
 self.btn\_login.clicked.connect(self.login)  
  
 layout.addWidget(self.lbl\_login)  
 layout.addWidget(self.txt\_login)  
 layout.addWidget(self.lbl\_password)  
 layout.addWidget(self.txt\_password)  
 layout.addWidget(self.btn\_login)  
  
 self.setLayout(layout)  
  
 def login(self):  
 login = self.txt\_login.text()  
 password = self.txt\_password.text()  
  
 if login == 'admin' and password == '1234':  
 self.admin\_window = AdminWindow()  
 self.admin\_window.show()  
 self.close()  
 else:  
 QMessageBox.warning(self, 'Ошибка', 'Логин или пароль введены неправильно')  
  
class AdminWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(350, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Администратор - Главное меню')  
 layout = QVBoxLayout()  
  
 self.btn\_profiles = HoverButton('Профили')  
 self.btn\_trainings = HoverButton('Тренировки')  
 self.btn\_reviews = HoverButton('Отзывы')  
 self.btn\_profiles.setFixedSize(350, 50)  
 self.btn\_trainings.setFixedSize(350, 50)  
 self.btn\_reviews.setFixedSize(350, 50)  
  
 self.btn\_profiles.clicked.connect(self.openProfiles)  
 self.btn\_trainings.clicked.connect(self.openTrainings)  
 self.btn\_reviews.clicked.connect(self.openReviews)  
  
 layout.addWidget(self.btn\_profiles)  
 layout.addWidget(self.btn\_trainings)  
 layout.addWidget(self.btn\_reviews)  
  
 self.setLayout(layout)  
  
 def openProfiles(self):  
 self.profiles\_window = ProfilesWindow()  
 self.profiles\_window.show()  
  
 def openTrainings(self):  
 self.trainings\_window = TrainingsWindow()  
 self.trainings\_window.show()  
  
 def openReviews(self):  
 self.reviews\_window = ReviewsWindow()  
 self.reviews\_window.show()  
  
class ProfilesWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(640, 400) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Администратор - Профили')  
 layout = QVBoxLayout()  
  
 self.table = QTableWidget(self)  
 self.table.setColumnCount(6)  
 self.table.setHorizontalHeaderLabels(['ID', 'Логин', 'Пароль', 'Фамилия', 'Имя', 'Отчество'])  
  
 self.loadProfiles()  
  
 layout.addWidget(self.table)  
 self.setLayout(layout)  
  
 def loadProfiles(self):  
 cursor.execute('SELECT \* FROM users')  
 users = cursor.fetchall()  
  
 self.table.setRowCount(len(users))  
 for row, user in enumerate(users):  
 for col, item in enumerate(user):  
 self.table.setItem(row, col, QTableWidgetItem(str(item)))  
  
class TrainingsWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(400, 150) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Администратор - Тренировки')  
 layout = QVBoxLayout()  
  
 self.day\_combo = QComboBox()  
 for i in range(1, 32):  
 self.day\_combo.addItem(str(i))  
  
 self.btn\_select\_day = HoverButton('Выбрать день')  
 self.btn\_select\_day.setFixedSize(350, 30)  
 self.btn\_select\_day.clicked.connect(self.selectDay)  
  
 layout.addWidget(self.day\_combo)  
 layout.addWidget(self.btn\_select\_day)  
  
 self.setLayout(layout)  
  
 def selectDay(self):  
 day = int(self.day\_combo.currentText())  
 self.day\_window = DayTrainingsWindow(day)  
 self.day\_window.show()  
  
class DayTrainingsWindow(QWidget):  
 def \_\_init\_\_(self, day):  
 super().\_\_init\_\_()  
 self.day = day  
 self.initUI()  
 self.resize(340, 400) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle(f'Администратор - Тренировки на {self.day} день')  
 layout = QVBoxLayout()  
  
 self.table = QTableWidget(self)  
 self.table.setColumnCount(3)  
 self.table.setHorizontalHeaderLabels(['Спорт', 'Время', 'Тренер'])  
  
 self.loadTrainings()  
  
 self.btn\_add = HoverButton('Добавить тренировку')  
 self.btn\_delete = HoverButton('Удалить тренировку')  
  
 self.btn\_add.clicked.connect(self.addTraining)  
 self.btn\_delete.clicked.connect(self.deleteTraining)  
  
 layout.addWidget(self.table)  
 layout.addWidget(self.btn\_add)  
 layout.addWidget(self.btn\_delete)  
  
 self.setLayout(layout)  
  
 def loadTrainings(self):  
 cursor.execute('SELECT sport, time, coach FROM trainings WHERE day = ?', (self.day,))  
 trainings = cursor.fetchall()  
  
 self.table.setRowCount(len(trainings))  
 for row, training in enumerate(trainings):  
 for col, item in enumerate(training):  
 self.table.setItem(row, col, QTableWidgetItem(str(item)))  
  
 def addTraining(self):  
 sport, ok = QInputDialog.getText(self, 'Добавить тренировку', 'Спорт:')  
 if ok and sport:  
 time, ok = QInputDialog.getText(self, 'Добавить тренировку', 'Время:')  
 if ok and time:  
 coach, ok = QInputDialog.getText(self, 'Добавить тренировку', 'Тренер:')  
 if ok and coach:  
 cursor.execute('INSERT INTO trainings (sport, time, coach, day) VALUES (?, ?, ?, ?)', (sport, time, coach, self.day))  
 conn.commit()  
 self.loadTrainings()  
  
 def deleteTraining(self):  
 selected\_row = self.table.currentRow()  
 if selected\_row >= 0:  
 sport = self.table.item(selected\_row, 0).text()  
 time = self.table.item(selected\_row, 1).text()  
 coach = self.table.item(selected\_row, 2).text()  
 cursor.execute('DELETE FROM trainings WHERE sport = ? AND time = ? AND coach = ? AND day = ?', (sport, time, coach, self.day))  
 conn.commit()  
 self.loadTrainings()  
  
class ReviewsWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(350, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Администратор - Отзывы')  
 layout = QVBoxLayout()  
  
 self.review\_list = QListWidget()  
 self.loadReviews()  
  
 self.review\_list.itemClicked.connect(self.showReview)  
  
 layout.addWidget(self.review\_list)  
 self.setLayout(layout)  
  
 def loadReviews(self):  
 cursor.execute('''SELECT r.rating, r.review, u.last\_name, u.first\_name, u.middle\_name, u.id  
 FROM reviews r  
 JOIN users u ON r.user\_id = u.id''')  
 reviews = cursor.fetchall()  
  
 for rating, review, last\_name, first\_name, middle\_name, user\_id in reviews:  
 self.review\_list.addItem(f'Оценка: {rating} (ФИО: {last\_name} {first\_name} {middle\_name}, ID: {user\_id})')  
  
 def showReview(self, item):  
 user\_id = int(item.text().split('ID: ')[1])  
 cursor.execute('SELECT review FROM reviews WHERE user\_id = ?', (user\_id,))  
 review = cursor.fetchone()[0]  
  
 msg\_box = QMessageBox(self)  
 msg\_box.setWindowTitle('Отзыв')  
 msg\_box.setText(review)  
 msg\_box.setStandardButtons(QMessageBox.StandardButton.Ok | QMessageBox.StandardButton.Cancel)  
 msg\_box.buttonClicked.connect(lambda btn: self.openUserProfile(btn, user\_id) if btn.text() == 'OK' else None)  
 msg\_box.exec()  
  
 def openUserProfile(self, btn, user\_id):  
 self.user\_profile\_window = UserProfileWindow(user\_id)  
 self.user\_profile\_window.show()  
  
class UserProfileWindow(QWidget):  
 def \_\_init\_\_(self, user\_id):  
 super().\_\_init\_\_()  
 self.user\_id = user\_id  
 self.initUI()  
 self.resize(400, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Профиль пользователя')  
 layout = QVBoxLayout()  
  
 cursor.execute('SELECT last\_name, first\_name, middle\_name, age FROM users WHERE id = ?', (self.user\_id,))  
 profile = cursor.fetchone()  
  
 profile\_info = f'Фамилия: {profile[0]}\nИмя: {profile[1]}\nОтчество: {profile[2]}\nВозраст: {profile[3]}'  
 self.lbl\_profile = QLabel(profile\_info)  
  
 layout.addWidget(self.lbl\_profile)  
 self.setLayout(layout)  
  
class RegisterWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(400, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Регистрация')  
 layout = QVBoxLayout()  
  
 self.lbl\_last\_name = QLabel('Фамилия:')  
 self.txt\_last\_name = QLineEdit()  
 self.lbl\_first\_name = QLabel('Имя:')  
 self.txt\_first\_name = QLineEdit()  
 self.lbl\_middle\_name = QLabel('Отчество:')  
 self.txt\_middle\_name = QLineEdit()  
 self.lbl\_age = QLabel('Возраст:')  
 self.txt\_age = QLineEdit()  
 self.lbl\_login = QLabel('Логин:')  
 self.txt\_login = QLineEdit()  
 self.lbl\_password = QLabel('Пароль:')  
 self.txt\_password = QLineEdit()  
 self.txt\_password.setEchoMode(QLineEdit.EchoMode.Password)  
 self.btn\_register = HoverButton('Зарегистрироваться')  
  
 self.btn\_register.clicked.connect(self.register)  
  
 layout.addWidget(self.lbl\_last\_name)  
 layout.addWidget(self.txt\_last\_name)  
 layout.addWidget(self.lbl\_first\_name)  
 layout.addWidget(self.txt\_first\_name)  
 layout.addWidget(self.lbl\_middle\_name)  
 layout.addWidget(self.txt\_middle\_name)  
 layout.addWidget(self.lbl\_age)  
 layout.addWidget(self.txt\_age)  
 layout.addWidget(self.lbl\_login)  
 layout.addWidget(self.txt\_login)  
 layout.addWidget(self.lbl\_password)  
 layout.addWidget(self.txt\_password)  
 layout.addWidget(self.btn\_register)  
  
 self.setLayout(layout)  
  
 def register(self):  
 last\_name = self.txt\_last\_name.text()  
 first\_name = self.txt\_first\_name.text()  
 middle\_name = self.txt\_middle\_name.text()  
 age = self.txt\_age.text()  
 login = self.txt\_login.text()  
 password = self.txt\_password.text()  
  
 if not last\_name or not first\_name or not middle\_name or not age or not login or not password:  
 QMessageBox.warning(self, 'Ошибка', 'Все поля должны быть заполнены')  
 return  
  
 try:  
 age = int(age)  
 except ValueError:  
 QMessageBox.warning(self, 'Ошибка', 'Возраст должен быть числом')  
 return  
  
 try:  
 cursor.execute('INSERT INTO users (login, password, first\_name, last\_name, middle\_name, age) VALUES (?, ?, ?, ?, ?, ?)',  
 (login, password, first\_name, last\_name, middle\_name, age))  
 conn.commit()  
 QMessageBox.information(self, 'Успех', 'Регистрация прошла успешно')  
 self.close()  
 except sqlite3.IntegrityError:  
 QMessageBox.warning(self, 'Ошибка', 'Логин уже занят')  
  
class UserLoginWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.initUI()  
 self.resize(400, 200) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Пользователь - Вход')  
 layout = QVBoxLayout()  
  
 self.lbl\_login = QLabel('Логин:')  
 self.txt\_login = QLineEdit()  
 self.lbl\_password = QLabel('Пароль:')  
 self.txt\_password = QLineEdit()  
 self.txt\_password.setEchoMode(QLineEdit.EchoMode.Password)  
 self.btn\_login = HoverButton('Войти')  
  
 self.btn\_login.clicked.connect(self.login)  
  
 layout.addWidget(self.lbl\_login)  
 layout.addWidget(self.txt\_login)  
 layout.addWidget(self.lbl\_password)  
 layout.addWidget(self.txt\_password)  
 layout.addWidget(self.btn\_login)  
  
 self.setLayout(layout)  
  
 def login(self):  
 login = self.txt\_login.text()  
 password = self.txt\_password.text()  
  
 cursor.execute('SELECT id FROM users WHERE login = ? AND password = ?', (login, password))  
 user\_id = cursor.fetchone()  
  
 if user\_id:  
 self.user\_window = UserWindow(user\_id[0])  
 self.user\_window.show()  
 self.close()  
 else:  
 QMessageBox.warning(self, 'Ошибка', 'Неверный логин или пароль')  
  
class UserWindow(QWidget):  
 def \_\_init\_\_(self, user\_id):  
 super().\_\_init\_\_()  
 self.user\_id = user\_id  
 self.initUI()  
 self.resize(400, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Пользователь - Главное меню')  
 layout = QVBoxLayout()  
  
 self.btn\_profile = HoverButton('Мой профиль')  
 self.btn\_trainings = HoverButton('Тренировки')  
 self.btn\_my\_trainings = HoverButton('Мои тренировки')  
 self.btn\_review = HoverButton('Оставить отзыв')  
  
 self.btn\_profile.setFixedSize(350, 50)  
 self.btn\_trainings.setFixedSize(350, 50)  
 self.btn\_my\_trainings.setFixedSize(350, 50)  
 self.btn\_review.setFixedSize(350, 50)  
  
 self.btn\_profile.clicked.connect(self.openProfile)  
 self.btn\_trainings.clicked.connect(self.openTrainings)  
 self.btn\_my\_trainings.clicked.connect(self.openMyTrainings)  
 self.btn\_review.clicked.connect(self.openReview)  
  
 layout.addWidget(self.btn\_profile)  
 layout.addWidget(self.btn\_trainings)  
 layout.addWidget(self.btn\_my\_trainings)  
 layout.addWidget(self.btn\_review)  
  
 self.setLayout(layout)  
  
 def openProfile(self):  
 cursor.execute('SELECT last\_name, first\_name, middle\_name, age FROM users WHERE id = ?', (self.user\_id,))  
 profile = cursor.fetchone()  
  
 profile\_info = f'Фамилия: {profile[0]}\nИмя: {profile[1]}\nОтчество: {profile[2]}\nВозраст: {profile[3]}'  
 QMessageBox.information(self, 'Мой профиль', profile\_info)  
  
 def openTrainings(self):  
 self.trainings\_window = UserTrainingsWindow(self.user\_id)  
 self.trainings\_window.show()  
  
 def openMyTrainings(self):  
 self.my\_trainings\_window = MyTrainingsWindow(self.user\_id)  
 self.my\_trainings\_window.show()  
  
 def openReview(self):  
 self.review\_window = ReviewWindow(self.user\_id)  
 self.review\_window.show()  
  
class UserTrainingsWindow(QWidget):  
 def \_\_init\_\_(self, user\_id):  
 super().\_\_init\_\_()  
 self.user\_id = user\_id  
 self.initUI()  
 self.resize(400, 150) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Пользователь - Тренировки')  
 layout = QVBoxLayout()  
  
 self.day\_combo = QComboBox()  
 for i in range(1, 32):  
 self.day\_combo.addItem(str(i))  
  
 self.btn\_select\_day = HoverButton('Выбрать день')  
 self.btn\_select\_day.clicked.connect(self.selectDay)  
  
 layout.addWidget(self.day\_combo)  
 layout.addWidget(self.btn\_select\_day)  
  
 self.setLayout(layout)  
  
 def selectDay(self):  
 day = int(self.day\_combo.currentText())  
 self.day\_window = UserDayTrainingsWindow(self.user\_id, day)  
 self.day\_window.show()  
  
class UserDayTrainingsWindow(QWidget):  
 def \_\_init\_\_(self, user\_id, day):  
 super().\_\_init\_\_()  
 self.user\_id = user\_id  
 self.day = day  
 self.initUI()  
 self.resize(340, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle(f'Пользователь - Тренировки на {self.day} день')  
 layout = QVBoxLayout()  
  
 self.table = QTableWidget(self)  
 self.table.setColumnCount(3)  
 self.table.setHorizontalHeaderLabels(['Спорт', 'Время', 'Тренер'])  
  
 self.loadTrainings()  
  
 self.btn\_select = HoverButton('Выбрать тренировку')  
 self.btn\_select.clicked.connect(self.selectTraining)  
  
 layout.addWidget(self.table)  
 layout.addWidget(self.btn\_select)  
  
 self.setLayout(layout)  
  
 def loadTrainings(self):  
 cursor.execute('SELECT sport, time, coach FROM trainings WHERE day = ?', (self.day,))  
 trainings = cursor.fetchall()  
  
 self.table.setRowCount(len(trainings))  
 for row, training in enumerate(trainings):  
 for col, item in enumerate(training):  
 self.table.setItem(row, col, QTableWidgetItem(str(item)))  
  
 def selectTraining(self):  
 selected\_row = self.table.currentRow()  
 if selected\_row >= 0:  
 sport = self.table.item(selected\_row, 0).text()  
 time = self.table.item(selected\_row, 1).text()  
 coach = self.table.item(selected\_row, 2).text()  
 cursor.execute('SELECT id FROM trainings WHERE sport = ? AND time = ? AND coach = ? AND day = ?', (sport, time, coach, self.day))  
 training\_id = cursor.fetchone()[0]  
 cursor.execute('INSERT INTO user\_trainings (user\_id, training\_id, day) VALUES (?, ?, ?)', (self.user\_id, training\_id, self.day))  
 conn.commit()  
 QMessageBox.information(self, 'Успех', 'Вы успешно записались на тренировку')  
  
class MyTrainingsWindow(QWidget):  
 def \_\_init\_\_(self, user\_id):  
 super().\_\_init\_\_()  
 self.user\_id = user\_id  
 self.initUI()  
 self.resize(420, 400) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Пользователь - Мои тренировки')  
 layout = QVBoxLayout()  
  
 self.table = QTableWidget(self)  
 self.table.setColumnCount(4)  
 self.table.setHorizontalHeaderLabels(['Спорт', 'Время', 'Тренер', 'День'])  
  
 self.loadTrainings()  
  
 self.btn\_delete = HoverButton('Удалить тренировку')  
 self.btn\_delete.clicked.connect(self.deleteTraining)  
  
 layout.addWidget(self.table)  
 layout.addWidget(self.btn\_delete)  
  
 self.setLayout(layout)  
  
 def loadTrainings(self):  
 cursor.execute('''SELECT t.sport, t.time, t.coach, ut.day  
 FROM user\_trainings ut  
 JOIN trainings t ON ut.training\_id = t.id  
 WHERE ut.user\_id = ?  
 ORDER BY ut.day, t.time''', (self.user\_id,))  
 trainings = cursor.fetchall()  
  
 self.table.setRowCount(len(trainings))  
 for row, training in enumerate(trainings):  
 for col, item in enumerate(training):  
 self.table.setItem(row, col, QTableWidgetItem(str(item)))  
  
 def deleteTraining(self):  
 selected\_row = self.table.currentRow()  
 if selected\_row >= 0:  
 sport = self.table.item(selected\_row, 0).text()  
 time = self.table.item(selected\_row, 1).text()  
 coach = self.table.item(selected\_row, 2).text()  
 day = self.table.item(selected\_row, 3).text()  
 cursor.execute('SELECT id FROM trainings WHERE sport = ? AND time = ? AND coach = ? AND day = ?', (sport, time, coach, day))  
 training\_id = cursor.fetchone()[0]  
 cursor.execute('DELETE FROM user\_trainings WHERE user\_id = ? AND training\_id = ? AND day = ?', (self.user\_id, training\_id, day))  
 conn.commit()  
 QMessageBox.information(self, 'Успех', 'Вы успешно удалили тренировку')  
 self.loadTrainings()  
  
class ReviewWindow(QWidget):  
 def \_\_init\_\_(self, user\_id):  
 super().\_\_init\_\_()  
 self.user\_id = user\_id  
 self.initUI()  
 self.resize(400, 300) # Установка размера окна  
  
 def initUI(self):  
 self.setWindowTitle('Пользователь - Оставить отзыв')  
 layout = QVBoxLayout()  
  
 self.lbl\_rating = QLabel('Оценка:')  
 self.rating\_combo = QComboBox()  
 for i in range(1, 6):  
 self.rating\_combo.addItem(str(i))  
  
 self.lbl\_review = QLabel('Отзыв:')  
 self.txt\_review = QTextEdit()  
  
 self.btn\_submit = HoverButton('Отправить отзыв')  
 self.btn\_submit.clicked.connect(self.submitReview)  
  
 layout.addWidget(self.lbl\_rating)  
 layout.addWidget(self.rating\_combo)  
 layout.addWidget(self.lbl\_review)  
 layout.addWidget(self.txt\_review)  
 layout.addWidget(self.btn\_submit)  
  
 self.setLayout(layout)  
  
 def submitReview(self):  
 rating = int(self.rating\_combo.currentText())  
 review = self.txt\_review.toPlainText()  
  
 if not review:  
 QMessageBox.warning(self, 'Ошибка', 'Отзыв не может быть пустым')  
 return  
  
 cursor.execute('INSERT INTO reviews (user\_id, rating, review) VALUES (?, ?, ?)', (self.user\_id, rating, review))  
 conn.commit()  
 QMessageBox.information(self, 'Успех', 'Отзыв успешно отправлен')  
 self.close()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 login\_window = LoginWindow()  
 login\_window.show()  
 sys.exit(app.exec())