МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования   
«Южный федеральный университет»

Институт математики, механики   
и компьютерных наук им. И. И. Воровича

Кафедра прикладной математики и программирования

Дурманенко Всеволод Сергеевич

Оценка положения тела на видео

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
по направлению подготовки  
02.03.02 – Фундаментальная информатика и информационные технологии

**Научный руководитель** –   
доц. Демяненко Яна Михайловна

Допущено к защите:  
заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сидоров С. С.

Ростов-на-Дону – 2024

Оглавление

[Введение 4](#_Toc166081958)

[1. Постановка задачи 5](#_Toc166081959)

[2. Обзор подходов 6](#_Toc166081960)

[2.1. Сверточная нейросеть 6](#_Toc166081961)

[2.2. Методы обнаружения человека 6](#_Toc166081962)

[2.2.1. OpenPose 6](#_Toc166081963)

[2.2.2. AlphaPose (RMPE) 7](#_Toc166081964)

[2.2.3. DeepCut 9](#_Toc166081965)

[2.2.4. Mask R-CNN 10](#_Toc166081966)

[2.2.5. MediaPipe 11](#_Toc166081967)

[2.3. Представление анатомии человека для компьютера 12](#_Toc166081968)

[2.3.1. Моделирование анатомии человека 12](#_Toc166081969)

[3. Основная часть 15](#_Toc166081970)

[3.1. Идея 15](#_Toc166081971)

[3.2. Разработка алгоритма 15](#_Toc166081972)

[3.3. Упражнения 16](#_Toc166081973)

[3.4. Статистика 18](#_Toc166081974)

[3.5. Графический интерфейс 18](#_Toc166081975)

[3.5.1. Основное окно 18](#_Toc166081976)

[3.5.2. Видео окно 20](#_Toc166081977)

[3.5.3. Сохранение комплекса упражнений 20](#_Toc166081978)

[3.5.4. Окно синхронизации данных 21](#_Toc166081979)

[3.6. Удаленное хранение данных 21](#_Toc166081980)

[3.6.1. Идея работы 21](#_Toc166081981)

[3.6.2. Создание проекта 21](#_Toc166081982)

[3.6.3. Разработка API 22](#_Toc166081983)

[3.6.4. Работа с API со стороны пользователя 23](#_Toc166081984)

[3.6.5. Развертывание приложения 24](#_Toc166081985)

[3.6.6. Тестирование 25](#_Toc166081986)

[Приложения 26](#_Toc166081987)

[Заключение 27](#_Toc166081988)

[Литература 28](#_Toc166081989)

# Введение

В современном мире, где активный образ жизни и забота о здоровье становятся все более важными, фитнес-индустрия находится в центре внимания, предлагая различные инновационные способы поддержания физической формы. В рамках этого контекста разработка приложений становится неотъемлемой частью современной фитнес-культуры, обеспечивая пользователей инструментами для эффективного тренировочного процесса и отслеживания своих достижений.

Цель данной дипломной работы заключается в разработке и реализации фитнес-приложения, предназначенного для отслеживания выполнения упражнений. Данное приложение будет ориентировано на широкий круг пользователей, начиная от начинающих заниматься фитнесом до опытных спортсменов, стремящихся оптимизировать свои тренировочные процессы.

Целью данного проекта является не только создание удобного и эффективного инструмента для тренировок, но и способствование привлечению большего числа людей к здоровому образу жизни, путем предоставления им доступных и мотивирующих средств для занятий спортом.

# Постановка задачи

Целью данной работы является разработка приложения для правильного выполнения упражнений, связанных с разминкой и растяжкой. Для достижения данной цели можно выделить следующие задачи:

* Исследовать существующие методы анализа видеоданных и оценки положения объектов на видео.
* Разработать алгоритм для определения правильности выполнения упражнений, используя данные из изученных методов.
* Разработать метод ведения статистики для отслеживания прогресса.
* Разработать графический интерфейс.
* Разработать сервер для хранения данных пользователей.

# Обзор подходов

## Сверточная нейросеть

Сверточная нейронная сеть (CNN) представляет собой специализированный тип нейронной сети, применяемый в основном для обработки изображений и других данных с пространственной структурой, таких как аудио и видео. Основное отличие CNN от обычной нейронной сети заключается в способе обработки входных данных.

Основные концепции сверточных нейронных сетей включают в себя сверточные слои, которые обнаруживают локальные шаблоны, такие как грани и текстуры, путем скольжения окна фильтра по входным данным. Затем следует операция подвыборки, которая уменьшает размерность данных, сохраняя наиболее важные признаки.

После каждого сверточного или подвыборочного слоя обычно применяется нелинейная функция активации, такая как ReLU, для введения нелинейности в модель. И, наконец, после серии сверточных и подвыборочных слоев следует один или несколько полносвязных слоев, которые объединяют глобальные признаки для принятия окончательного решения.

В целом, сверточная нейронная сеть отличается от обычной тем, что она эффективно обрабатывает данные с пространственной структурой, сохраняя локальные шаблоны и признаки, что делает ее более подходящей для задач компьютерного зрения и обработки изображений.

## Методы обнаружения человека

### OpenPose

OpenPose – это восходящий подход, при котором сеть сначала обнаруживает части тела или ключевые точки на изображении, а затем сопоставляет соответствующие ключевые точки для формирования пар.

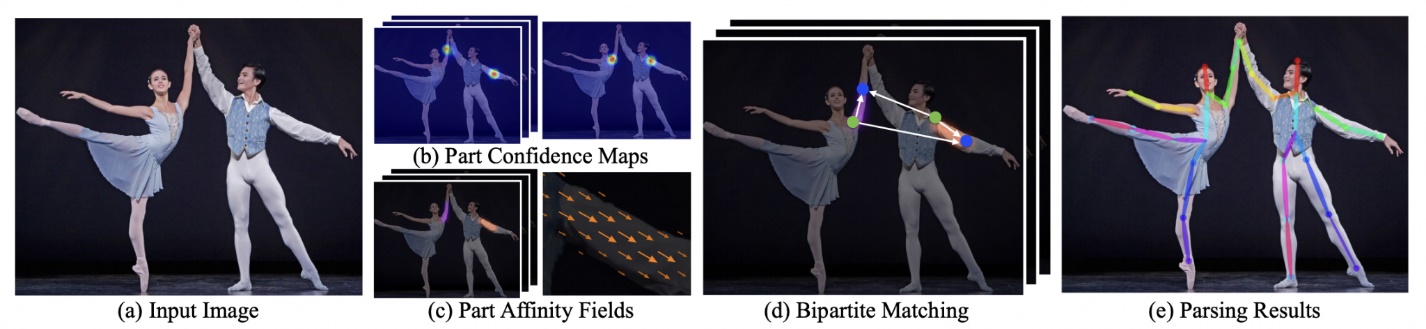
OpenPose также использует CNN в качестве основной архитектуры. Он состоит из сверточной сети VGG-19, которая используется для извлечения шаблонов и представлений из заданных входных данных. Выходные данные VGG-19 попадают в две ветви сверточных сетей.

Первая сеть прогнозирует набор карт достоверности для каждой части тела, а вторая ветвь прогнозирует поля сходства частей (PAF), которые создают определенную степень связи между частями. Также полезно удалить более слабые звенья в двудольных графах.

(фото)

На изображении выше показана архитектура OpenPose, которая представляет собой многоступенчатую CNN.

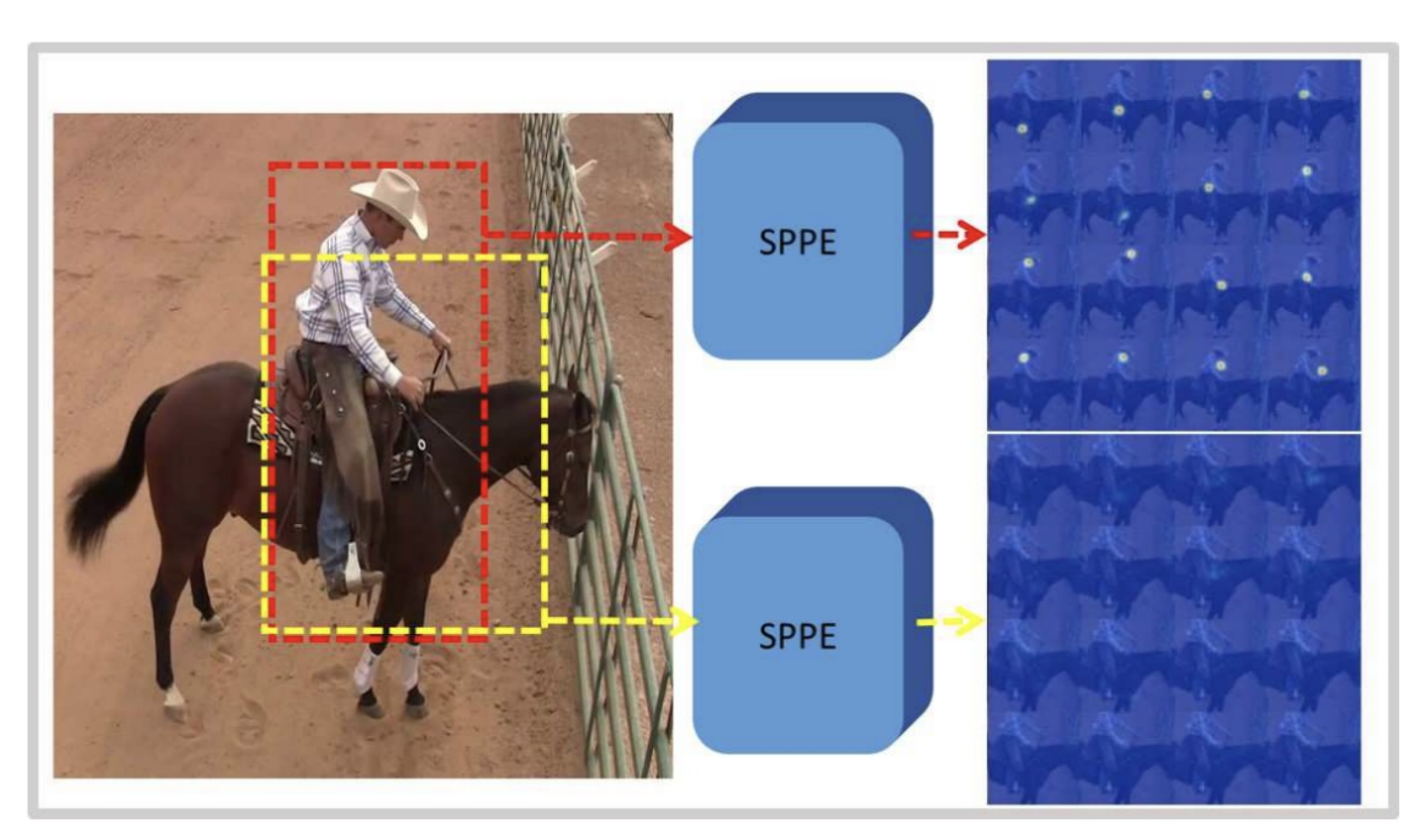
По сути, предсказания двух ветвей вместе с признаками объединяются на следующем этапе, чтобы сформировать человеческий скелет в зависимости от количества людей, присутствующих во входных данных. Последовательные этапы CNN используются для уточнения прогноза.



### AlphaPose (RMPE)

Региональная оценка положения нескольких человек (RMPE) или AlphaPose реализует нисходящий подход к HPE.

Нисходящий подход к HPE приводит к большому количеству ошибок при локализации и неточностям во время прогнозирования и, следовательно, является довольно сложной задачей.



Например, на изображении выше показаны две ограничивающие рамки: красная рамка представляет достоверное выделение, а желтая рамка представляет прогнозируемую ограничивающую рамку.

Хотя, когда дело доходит до классификации, желтая ограничивающая рамка будет считаться «правильной» ограничивающей рамкой для классификации человека. Однако позу человека невозможно оценить даже с помощью «правильной» ограничивающей рамки.

Авторы AlphaPose решили проблему несовершенного обнаружения людей с помощью двухэтапной схемы. В рамках этой структуры они представили две сети:

Симметричная пространственная трансформаторная сеть (SSTN): помогает вырезать соответствующую область на входе, что впоследствии упрощает задачу классификации, что приводит к повышению производительности.

Оценщик позы одного человека (SPPE): он используется для извлечения и оценки позы человека.

Цель AlphaPose — извлечь высококачественную область с одним человеком из неточной ограничивающей рамки путем присоединения SSTN к SPPE. Этот метод повышает эффективность классификации, обеспечивая инвариантность и обеспечивая при этом стабильную основу для оценки позы человека.



### DeepCut

DeepCut был предложен Леонидом Пищулиным и др. ал. в 2016 году с целью совместного решения задач обнаружения и оценки позы одновременно.

Это восходящий подход к оценке позы человека.

Идея заключалась в том, чтобы обнаружить все возможные части тела на данном изображении, затем обозначить их, например, голову, руки, ноги и т. д., с последующим процессом разделения частей тела, принадлежащих каждому человеку.



Сеть использует моделирование интегрального линейного программирования (ILP ) для неявной группировки всех обнаруженных ключевых точек в заданных входных данных так, чтобы конечный результат напоминал скелетное представление человека.

### Mask R-CNN

Модель Mask R-CNN имеет возможность одновременно локализовать и классифицировать объекты, создавая ограничивающую рамку вокруг объекта, а также создавая маску сегментации.

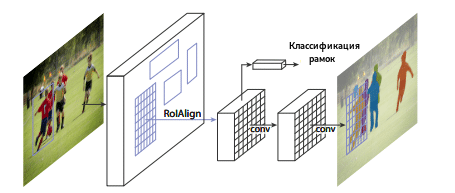


Fast R-CNN использует CNN для извлечения функций и представлений из заданных входных данных.

Извлеченные функции затем используются, чтобы предложить, где объект может присутствовать через сеть предложений регионов (RPN).

Поскольку ограничивающая рамка может иметь разные размеры, как на изображении выше, слой под названием RoIAlign используется для нормализации извлеченных объектов, чтобы все они имели одинаковые размеры.

Извлеченные признаки передаются в параллельные ветви сети для уточнения предлагаемой области интереса (RoI) для создания ограничивающих рамок и масок сегментации.



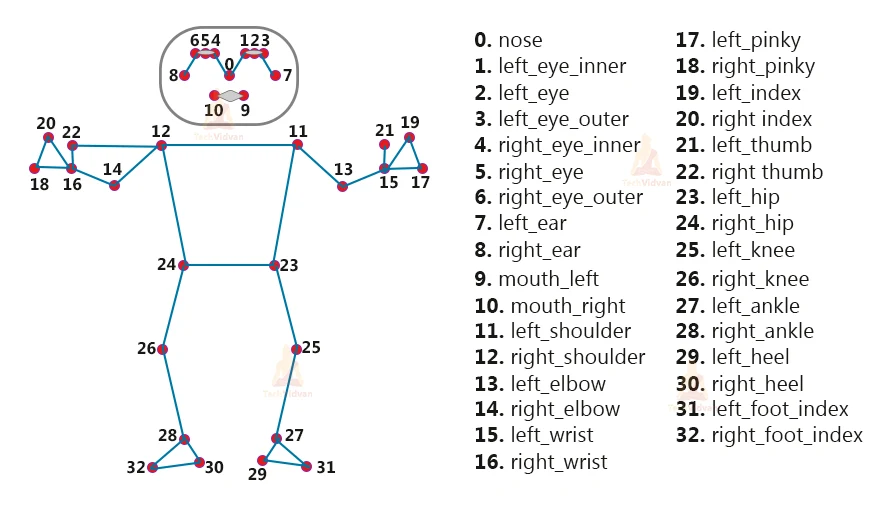
Когда дело доходит до оценки позы человека, выходные данные сегментации маски, полученные сетью, можно использовать для обнаружения людей на заданных входных данных. Поскольку сегментация по маске очень точна при обнаружении объектов, в данном случае при обнаружении человека, позу человека можно оценить довольно легко.

Этот метод напоминает подход «сверху вниз», при котором этап обнаружения человека выполняется параллельно этапу обнаружения деталей.

Другими словами, этап обнаружения ключевой точки и этап обнаружения человека независимы друг от друга.

### MediaPipe

MediaPipe Pose — это система оценки позы одного человека. Она использует топологию ориентиров BlazePose 33. BlazePose — это расширенный набор ключевых точек COCO, топологии Blaze Palm и Blaze Face. В свою очередь COCO – это набор данных, широко используемый для оценки алгоритмов компьютерного зрения, особенно для задачи обнаружения объектов и сегментации. Набор данных COCO содержит большой набор изображений, размеченных для различных задач компьютерного зрения. Алгоритм работает в два этапа – обнаружение и отслеживание. Поскольку обнаружение не выполняется в каждом кадре, MediaPipe может выполнять вывод быстро.



## Представление анатомии человека для компьютера

Определение положения и ориентации различных частей человеческого тела на изображениях или видео - это ключевая задача в области компьютерного зрения и искусственного интеллекта. Эта задача имеет широкий спектр применений, включая разработку систем для управления роботами, анализа видеонаблюдения, виртуальной реальности, медицинской диагностики и даже в области спортивных технологий. Оценка положения тела позволяет компьютерным системам понимать и интерпретировать человеческое движение, что является важным шагом к созданию более умных и автономных устройств.

### Моделирование анатомии человека

В большинстве методов используется жесткая кинематическая модель N-суставов, в которой человеческое тело представлено как объект с суставами и конечностями, содержащими кинематическую структуру тела и информацию о форме тела.

Существуют три типа моделирования анатомии человека:

* **Kinematic model** (Скелетная модель) - это модель также известная как скелетно-ориентированная модель, и она применяется для анализа позы в форматах 2D и 3D. Эта удобная и интуитивно понятная модель человеческого тела включает набор суставов и ориентацию конечностей для описания структуры тела. Таким образом, модели, оценивающие положение скелета, используются для улавливания взаимосвязей между различными частями тела. Однако кинематические модели имеют ограничения в передаче информации о текстуре или форме.
* **Planar model** (Плоская модель) – это модель, известная также как модель на основе контура, применяется для оценки 2D позы. Плоские модели используются для представления внешнего вида и формы человеческого тела. Обычно различные части тела представлены несколькими прямоугольниками, которые приближены к контуру человеческого тела. Хорошим примером такой модели является модель активных форм, которая используется для охвата полного контура человеческого тела и его деформации с помощью анализа главных компонентов.
* **Volumetric model** (Объемная модель) – это объемная модель, применяемая для трехмерной оценки положения, представляет собой важный инструмент в анализе поз. Они предназначены для моделирования статистических и артикулированных форм человеческого тела и поз. Эти модели могут быть использованы для вывода различных характеристик и параметров позы.



# Основная часть

## Идея

Основная идея приложения заключается в том, чтобы предоставить пользователям удобный инструмент для отслеживания и учета выполненных повторений упражнений во время тренировок. Это приложение предназначено для людей, которым важно держать себя в форме.

Пользователи могут выбирать упражнения из списка. Для начала тренировки необходимо выбрать количество желаемых повторений или выбрать заранее сохраненный комплекс упражнений. Приложение сохраняет информацию о проведенной тренировке и предоставляет пользователю возможность просматривать свой прогресс в удобном графическом виде.

## Разработка алгоритма

Для решения поставленной задачи был выбран язык программирования Python с библиотеками Mediapipe и OpenCV для работы с изображениями и камерой, PyQt для разработки интерфейса, Matplotlib и Pandas для работы со статистикой.

Для начала работы было создано виртуальное окружение, чтобы библиотеки устанавливались только для текущего проекта. Также был создан репозиторий на Github, и в файл .gitignore были добавлены файлы, содержащие временную информацию, библиотеки и файлы пользователей.

Суть алгоритма заключается в том, что его на вход подается изображение, полученное с веб-камеры пользователя. MediaPipe обрабатывает данное изображение. В результате получаются координаты ключевых точек и коэффициенты их видимости. Для корректного распознавания частей тела, необходимо, чтобы тело полностью находилось в кадре. Формат, в котором Mediapipe предоставляет данные, не очень удобен для дальнейшей работы. В связи с этим был создан класс Workout, в котором используется словарь в формате string:Point. Ключом здесь является название части тела, а значением – точка, содержащая координаты расположения этой части тела и коэффициенты видимости. Данный класс также содержит методы для определения выполнения упражнений и вспомогательный метод, отвечающий за то, что необходимые для выполнения части тела находятся в кадре целиком. Для удобства понимания стадии выполнения упражнения был создан Enum класс или класс с перечислением. В нем содержатся такие фазы выполнения упражнения как начальная, в процессе выполнения и законченная.

Основная логика алгоритма находится в функции display\_frame класса VideoWindow. Данная функция отвечает за отображение изображения с камеры. В качестве параметра данному методу передается словарь упражнений. В первую очередь необходимо проверить, остались ли невыполненные упражнения. Если таковых нет, то выводится сообщение о том, что комплекс выполнен, и окно можно закрывать. Также в конце происходит подсчет выполненных повторений для каждого упражнения, и эта информация записывается в файл data.csv для статистики. Если упражнения еще остались, то изображение с камеры преобразуется и происходит поиск ключевых точек. На данном этапе необходимо убедиться, что часть тела, соответствующая выполняемому упражнению, полностью находится в кадре, иначе возникает исключение и выводится сообщение о том, что необходимо сместиться. Если все в порядке, то на экран выводится список оставшихся к выполнению упражнений. Происходит проверка фазы выполнения, и если она имеет значение законченной, то из словаря с упражнениями вычитается одно повторение для текущего упражнения. В случае если количество оставшихся повторений равно нулю, то данное упражнение из словаря убирается.

## Упражнения

Упражнения можно разделить на две группы. К первой относятся те, для которых признак завершения – это угол между ключевыми точками. Ко второй группе относятся те, которые завершаются при определенном положении частей тела относительно друг друга.

Для отслеживания выполнения каждого упражнения из первой группы необходимо выделить ключевые точки, по которым будут определяться углы выполнения. По этим углам можно понять, если повторение уже выполнено или еще выполняется. Для начала выполнения упражнения угол должен быть из колонки “Угол начала”, а для завершения – из колонки “Угол завершения”.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Ключевые точки | Угол начала | Угол завершения |
| Приседания | Бедра  Колени  Лодыжки | 170 | 80 |
| Наклоны шеи | Нос  Ухо  Плечо | 110 | 80 |
| Сгибания в локтях | Плечо  Локоть  Запястье | 140 | 50 |
| Разгибания из-за головы | Плечо  Локоть  Кисть | 50 | 140 |
| Наклоны корпуса вперед | Нос  Бедра  Колени | 150 | 100 |
| Наклоны корпуса в стороны | Плечо  Бедро  Колено | 170 | 160 |
| Выпады ногами | Бедро  Колено  Лодыжка | 170 | 90 |

Для второй группы все гораздо проще. Необходимо отследить только положение ключевых точек относительно друг друга. Таким образом выбираются главные ключевые точки и рассматривается положение остальных точек относительно них.

|  |  |  |
| --- | --- | --- |
| Название | Ключевые точки | Главные ключевые точка |
| Подъемы рук над головой | Левое запястье  Правое запястье | Нос |
| Разведение рук в стороны | Левое запястье  Правое запястье | Левое плечо  Правое плечо |

## Статистика

Как было сказано ранее, статистика является неотъемлемой частью тренировок. В данном случае для ее отображения используется библиотека Matplotlib и Pandas. Исходные данные хранятся в файле data.csv, куда информация поступает при завершении комплекса упражнений. Для отображения гистограммы данные из этого файла считываются при помощи функции read\_csv из Pandas. Полученную информацию уже преобразуется Matplotlib. В результате можно увидеть наглядный результат тренировок.

## Графический интерфейс

Для создания графического интерфейса была использована библиотека PyQt5. PyQt5 – это набор Python-обёрток над Qt, фреймворком для создания графических интерфейсов. Выбор был сделан в пользу данной библиотеки из-за простоты её использования и кроссплатформенности.

### Основное окно

Для создания основного графического окна был создан класс MainWindow, унаследованный от QMainWindow. Внутри вызывается конструктор, в котором определяются элементы интерфейса и их расположение. Для вертикального расположения элементов используется QVBoxLayout, обернутый в QWidget. Для удобства представления списка упражнений и заданных повторений используется таблица (QTableWidget). Чтобы перемещать элементы внутри таблицы используются две кнопки (QPushButton). Это нужно для того, чтобы задать порядок выполнения упражнений. PyQt5 содержит некоторое количество изображений, которые можно использовать для создания понятного графического интерфейса. В данном случае оттуда были взяты изображения стрелок вверх и вниз.

Приложение рассчитано на то, что сразу несколько пользователей могут его использовать. Для начала использования необходимо создать хотя бы одного пользователя. Каждому пользователю создается отдельная папка в проекте для хранения статистики только для него.

Пользователь также может устанавливать количество повторений не только вручную, но и использовать уже сохраненные наборы упражнений. Для этого необходимо открыть выпадающий список (QHBoxLayout) и выбрать желаемый комплекс. Для изменения количества повторений в выбранном комплексе достаточно ввести новые значения и нажать на кнопку “Сохранить”. Для удаления не желаемого комплекса достаточно выбрать его из списка и нажать на кнопку “Удалить”.

Одним из условий прогресса является статистика, которая отображает ваши достижения. Чтобы увидеть свой прогресс на протяжении определенного периода, нужно воспользоваться кнопкой “Показать статистику”. При нажатии будет отображена гистограмма по датам и количеству выполненных упражнений. В случае отсутствия файла со статистикой вы получите сообщение об этом. В самом конце графического окна расположена кнопка для того, чтобы начать тренировку. При нажатии на данную кнопку произойдет парсинг данных из таблицы и будет запущено другое окно, в котором присутствует изображение с камеры и отображаются выбранные упражнения.

### Видео окно

Основная логика приложения заложена именно в классе VideoWindow. Для получения изображения с камеры используется библиотека OpenCV. Видеопоток, полученный оттуда передается в QLabel. И каждые 30мс обновляется при помощи QTimer. Такая обертка нужна для того, чтобы окна могли работать друг с другом и реагировать на определенные события. Например, при закрытии основного окна все остальные окна тоже закрываются.

Сам список упражнений отображается в левом верхнем углу по порядку, заданному пользователем. При завершении комплекса упражнений можно увидеть сообщение об этом. В случае если алгоритм не может определить вас, вы увидите соответствующее сообщение.

### Сохранение комплекса упражнений

Данное окно предназначено для ввода названия комплекса упражнений, который пользователь хочет сохранить. Оно содержит надпись (QLabel), поле для ввода (QLineEdit) и кнопку “Сохранить” (QPushButton). В качестве параметра при открытии данного окна передается набор упражнений из таблицы. Для удобства хранения используется JSON. Формат хранения следующий:

* Набор упражнений 1
  + Упражнение 1: Количество повторений
  + Упражнение 2: Количество повторений
  + …
* Набор упражнений 2
  + …

При нажатии на кнопку “Сохранить” информация из файла сначала считывается в виде словаря. Затем в данном словаря перезаписывается значение, соответствующее введенному ключу. В конце новые данные записываются обратно в файл.

### Окно синхронизации данных

Окно содержит четыре кнопки. Они предназначены для того, чтобы у пользователя была возможность выбора между загрузкой данных на сервер для дальнейшего хранения, выгрузкой своих данных с сервера для дальнейшего использования, обновления данных и удаления с сервера.

## Удаленное хранение данных

### Идея работы

Так как на сервере необходимо хранить файлы, предоставляемые пользователями, то было принято решение использовать .zip формат для передачи данных. От пользователя будет поступать архив с файлами data.csv и saved\_sets.json. В случае успешной передачи архив должен быть проверен и выдан ответ на запрос. Если же пользователю будет необходимо получить свои файлы с сервера, то в ответ будет отправляться также архив, который должен быть распакован со стороны клиента в соответствующую директорию пользователя.

Каждый архив, находящийся на сервере, будет назван специальным именем, которое предварительно должен указать пользователь. Это имя будет соответствовать ключу для доступа пользователей к файлам.

### Создание проекта

Для создания сервера для удаленного хранения был выбран фреймворк NodeJS с ExpressJS. Разработка началась с инициализации самого проекта. Все настройки были оставлены по умолчанию. Для удобства дальнейшей разработки в конфигурационный файл package.json были добавлены два скрипта:

"scripts": {

"start": "node index.js",

"dev": "node --watch index.js"

}

Первый отвечает за запуск сервера без дополнительных параметров, а второй за запуск с параметром –watch, который позволяет автоматически перезагружать сервер при сохранении изменений в коде.

### Разработка API

Разработка началась с поиска необходимых библиотек. В первую очередь для обработки файлов из тела запроса была установлена библиотека express-fileupload. Для работы с .zip файлами используется adm-zip.

Для работы с API используются четыре метода HTTP запроса: GET, POST, PUT и DELETE.

* GET – получение файлов с сервера
* POST – первоначальное сохранение данных
* PUT – обновление уже существующих данных
* DELETE – удаление данных

В случае отправки пользователем GET запроса необходимо также указать ключ доступа. В противном случае будет получен ответ с кодом 400, что означает, что запрос был некорректный. По ключу происходит поиск файла с названием как у ключа. В случае успешного нахождения такого файла он отправляется обратно в теле ответа. Что касается неудачного результата поиска, то в таком случае будет отправлен код 404, который означает, что файл не найден.

При отправке POST запроса также необходимо указать имя ключа. По этому имени будет назван файл, содержащий данные пользователя. Перед сохранением отработает middleware функция для фильтрации запроса. Данная функция проверяет, что запрос вообще содержит файлы и в каком количестве. Также она проверяет mime type, который должен соответствовать application/zip. Если данная проверка не пройдена, то в ответ пользователю отправляется ответ с кодом 4XX и сообщением, что не так было с запросом.

Перед сохранением файлов идет проверка того, что файлы в архиве соответствуют тем, которые должны быть переданы. Если данная проверка пройдена, то данные сохраняются на сервере, и пользователю отправляется код 200.

Запрос PUT аналогичен запросу POST. Отличие заключается в том, что файл по ключу уже должен быть загружен на сервер. Иначе в ответ пользователю уйдет код 404. Если же файл найден на сервере, то он будет перезаписан, а ключ останется без изменений.

Если отправляется запрос DELETE, то идет проверка наличия файла по ключу. Если файл найден, то он удаляется. В ответ пользователь получает код 202. Если же файл по коду не найден, то в ответ высылается код 404.

### Работа с API со стороны пользователя

Для отправки запросов и обработки ответов на клиенте используется библиотека requests, а для архивации и разархивации используется zipfile. Чтобы не создавать файлов в процессе сохранения и отправки было принято решение хранить их в памяти при помощи io.BytesIO. Обязательно необходимо учесть от какого пользователя идет запрос, чтобы файлы сохранялись соответствующему отправителю.

Для разделения логики между четырьмя запросами были написаны четыре функции, каждая из которых соответствует запросам GET, POST, PUT и DELETE. Каждая из функций отправляет соответствующий запрос на адрес <http://localhost:3000> c параметром key, который обозначает введенный ключ.

При неудачном запросе пользователь увидит окно с ошибкой. Ответ, в случае успешного выполнения запроса, разный. Ниже приведена таблица ответов от сервера.

|  |  |  |
| --- | --- | --- |
| Запрос | Тело запроса | Ответ |
| GET | Пусто | Архив с данными, который распаковывается в папку выбранного пользователя |
| POST | Архив с данными выбранного пользователя | Код 200 |
| PUT | Архив с данными выбранного пользователя | Код 202 |
| DELETE | Пусто | Код 202 |

### Развертывание приложения

Развертывание на стороннем сервере является одним из важнейших компонентов полноценного приложения. Для удобства развертывания используется технология Docker. В папку с сервером помещен Dockerfile. В нем написаны инструкции создания Docker образа.

FROM node:22-alpine

WORKDIR /server

COPY . .

RUN npm install

RUN mkdir -p files

EXPOSE 3000

CMD ["npm", "start"]

В данном случае образ основан на дистрибутиве Linux под названием Alpine Linux. Он был выбран из-за своей легковесности. Рабочей папкой была назначена папка server. Сначала все файлы сервера копируются в основную директорию. После этого устанавливаются все необходимые библиотеки. В случае отсутствия папки files, в которой хранятся архивы с данными пользователей, она создается. Порт для работы был выбран 3000. И в конце идет запуск самого образа.

Помимо Dockerfile также был добавлен файл .dockerignore. Он отвечает за те файлы и папки, которые не должны попасть в контейнер. В данном случае это папки files и node\_modules. Первая создается автоматически при отсутствии ее, а вторая будет создана при загрузке библиотек.

Оберткой над Docker образами является технология Docker Compose. Для ее использования необходимо добавить docker-compose.yml файл в корневой каталог проекта. В данном случае он выглядит следующим образом:

version: '3.8'

services:

server:

build: ./server

container\_name: server

ports:

- 3000:3000

volumes:

- files:/server/files

volumes:

files:

Версия выбрана последняя 3.8. В разделе services указывается название сервиса server, расположение Dockerfile для данного сервиса, название контейнера, порт и раздел. Раздел необходим для того, чтобы данные, хранящиеся в определенной директории, при пересоздании контейнера не удалялись.

### Тестирование

Тестирование в случае сервера производилось при помощи Postman. Данная утилита очень удобна тем, что имеет расширение для VSCode. Таким образом не было необходимости устанавливать ничего дополнительно. Для тестирования были написаны четыре шаблона запросов: по одному для GET, POST, PUT и DELETE запросов. В случае GET запроса единственным параметром является key, то есть ключ доступа к файлам. В POST запросе же еще прикрепляется архив с файлами. PUT запрос аналогичен POST запросу. В случае DELETE запроса также как и в GET запросе передача параметра key, то есть ключа, является обязательной.

# Приложения

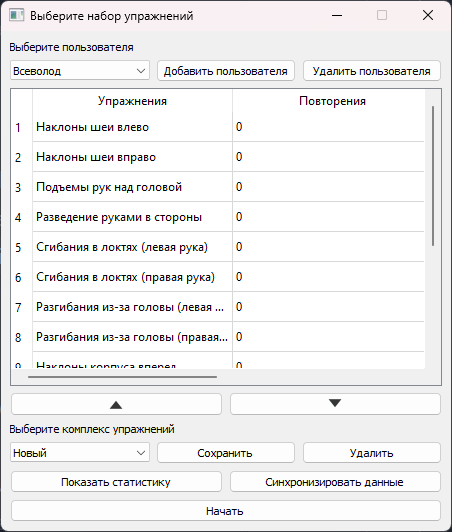


Рис. 1. Основное окно

# Заключение

В заключении следует отразить ключевые аспекты и достижения приложения, а также его потенциальное влияние на пользователей.

Приложение для отслеживания выполнения упражнений представляет собой мощный инструмент для улучшения физической активности и заботы о здоровье. За время разработки данного приложения были решены ряд ключевых задач:

* Разработан алгоритм, позволяющий распознавать различные упражнения и оценивать их правильность выполнения.
* Создан пользовательский интерфейс, интуитивно понятный и удобный в использовании, что способствует привлечению широкой аудитории пользователей.
* Реализована система ведения статистики выполнения упражнений, которая позволяет пользователям отслеживать свой прогресс, оценивать эффективность тренировок и планировать дальнейшие занятия.
* Разработан сервер для хранения файлов пользователей.

Эти достижения делают приложение не только эффективным инструментом для тренировок, но и мощным мотиватором для пользователей. Предоставление обратной связи о правильности выполнения упражнений и возможность наблюдать свой прогресс стимулируют регулярные тренировки и помогают пользователям достигать своих фитнес-целей.

В целом, разработанное приложение открывает новые перспективы для здорового образа жизни и физического развития, подтверждая свою значимость как для отдельных пользователей, так и для общества в целом.

# Литература

…