САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Разработка адаптивного веб-приложения для восстановления мелкой моторики рук у пациентов в период реабилитации

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Марахова Екатерина Игоревна

2 курс, группа 23.М07-мм

Научный руководитель:
доц. кафедры Системного программирования
Д.В. Луцив

Рецензент:
менеджер проектов ООО «СКЗ»
Н.А. Пенкрат

Консультант:
к.т.н., ведущий научный сотрудник лаборатории когнитивных интерфейсов
С.А. Пермяков

Санкт-Петербург, 2025

Оглавление

ВВЕДЕНИЕ

Восстановление мелкой моторики рук является важной составляющей реабилитации пациентов после инсультов, травм, неврологических заболеваний и операций [1, 2]. Традиционные методы реабилитации, такие как механические тренажеры и экзоскелеты, требуют значительных финансовых затрат [3, 4], индивидуальной настройки и зачастую недоступны для широкого круга пациентов и лечебных учреждений. В то же время развитие компьютерных технологий и машинного обучения открывает новые возможности для создания программных решений, способных, сохранив эффективность реабилитации, сделать её доступнее: для анализа моторики рук можно применить стандартную веб-камеру.

Современные компьютеры и мобильные устройства, будучи доступными широкому кругу потребителей и организаций, уже обладают достаточной мощностью и функциональностью для использования их в задачах реабилитации. На практике это уже подтверждено опытом применения ряда специализированных программных продуктов, которые либо предоставляют ограниченную функциональность [5], либо обладают высокой стоимостью использования [4].

Сейчас программные продукты с развитым пользовательским интерфейсом часто создаются в виде веб-приложений. Это позволяет упростить администрирование приложений, что важно применительно к тематике данной работы, так как ни пациенты, ни медицинские работники не обязаны быть IT-специалистами. Адаптивное веб-приложение, использующее алгоритмы компьютерного зрения для анализа движений, выглядит оптимальным решением, позволяющим проводить реабилитационные упражнения без дорогостоящего оборудования, с использованием лишь пользовательских устройств или на типичных «офисных» конфигурациях компьютеров. Применение микросервисной архитектуры [6] и современных технологий развёртывания, основанных на контейнеризации сервисов приложения [7], позволит сделать развёртывание приложения максимально

простым, а также, в случае использования лечебными учреждениями, потенциально облегчит интеграцию приложения с существующими сервисами.

В связи с этим разработка открытого, удобного и технологичного программного решения для восстановления мелкой моторики представляется актуальной задачей, способной улучшить качество реабилитационного процесса для широкого круга пациентов. Результаты работы могут быть полезны для реабилитологов, пациентов с нарушениями моторики, а также разработчиков медицинских программных решений, заинтересованных в создании современных цифровых инструментов для восстановительной медицины.

Цель и задачи работы

Целью данной работы является создание адаптивного веб-приложения для восстановления мелкой моторики рук, использующего технологии компьютерного зрения.

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ современных методов реабилитации мелкой моторики и существующих программных решений.
2. Проанализировать требования к приложению.
3. Выработать архитектуру приложения, обеспечивающую модульность и масштабируемость.
4. Создать адаптивное веб-приложение, для задач реабилитации.
5. Реализовать отказоустойчивое развёртывание приложения.

1. АНАЛИЗ СОВРЕМЕННЫХ МЕТОДОВ ВОССТАНОВЛЕНИЯ МЕЛКОЙ МОТОРИКИ РУК

1.1. Современные подходы к реабилитации мелкой моторики

Восстановление мелкой моторики рук является важной составляющей реабилитации пациентов с последствиями инсультов, черепно-мозговых травм, повреждений периферических нервов и нейродегенеративных заболеваний. Современные подходы к реабилитации можно условно разделить на три основные группы.

1. Традиционные методы (лечебная физкультура, мануальная терапия, эрготерапия) – основаны на повторяющихся упражнениях под контролем специалиста [8].
2. Аппаратные методы (механические тренажеры, роботизированные системы, экзоскелеты) – обеспечивают пассивную и активную разработку суставов с помощью технических устройств [9, 10].
3. Программно-аппаратные и цифровые решения (компьютерные тренажеры, VR/AR-системы, приложения с компьютерным зрением) – используют интерактивные технологии для мотивации пациента и объективной оценки прогресса [11, 13].

В последние годы наблюдается тенденция к цифровизации реабилитационных методик, что позволяет повысить их доступность и эффективность за счет автоматизации контроля выполнения упражнений и адаптивного подхода к нагрузкам [11, 12].

1.2. Обзор существующих решений: аппаратные и программные методы

Аппаратные решения

Среди российских разработок в области аппаратной реабилитации мелкой моторики можно выделить следующие :

– экзоскелеты (например, «Аника») – механические устройства, фиксирующие пальцы и запястье, обеспечивающие дозированную нагрузку. Их главный недостаток – высокая стоимость (от 300 тыс. руб.) и необходимость индивидуальной настройки [9, 14];
– роботизированные тренажеры (например, «Армед») – устройства с электроприводом, помогающие выполнять сгибание-разгибание пальцев. Требуют подключения к стационарным системам и контроля со стороны врача [10, 15] ;
– механические тренажеры (например, «Бутон») – простые устройства для разработки хватательных движений. Имеют ограниченную функциональность и не обеспечивают обратную связь [10].

Программные решения

В отличие от аппаратных методов, цифровые технологии предлагают более гибкие и доступные варианты реабилитации :

– VR-тренажеры (например, «Мотива VR») – используют виртуальную реальность для игровой реабилитации, но требуют дорогостоящего оборудования [11];
– мобильные приложения (например, «Rehand») – предлагают упражнения с тачскрином, но не анализируют правильность выполнения движений [13];
– компьютерные системы с камерой (например, Kinect-реабилитация) – позволяют отслеживать движения без дополнительных датчиков, но имеют ограниченную точность [12].

Наиболее перспективными представляются гибридные решения, сочетающие программные методы с минимальным использованием аппаратных средств (например, веб-камера).

1.3. Обзор существующих моделей компьютерного зрения для распознавания жестов рук

Современные подходы к распознаванию жестов рук можно разделить на три основные категории.

1. Классические алгоритмы компьютерного зрения (например, на основе OpenCV) :
   - используют детекцию ключевых точек через фильтры (SIFT, SURF);
   - требуют сложной предобработки изображений;
   - чувствительны к освещению и фону.
2. Традиционные нейросетевые архитектуры (CNN, LSTM)
   - эффективны для статичных жестов;
   - требуют больших размеченных датасетов;
   - пример: HSE GestureNet (разработка ВШЭ, 2021) – компактная CNN для распознавания 20 жестов с точностью ~89%.
3. Современные end-to-end решения (MediaPipe, MMPose);
   - комбинируют детекцию и классификацию;
   - работают в реальном времени;
   - оптимизированы для мобильных и веб-приложений.

Рассмотренные ИИ-модели для детекции рук также отображены в сравнительной таблице 1.

«Таблица 1. Сравнение популярных моделей»

Модель	Точность	FPS	Особенности
MediaPipe Hands	95% (21 точка)	30+	Готовая JS-реализация, не требует серверной обработки
MMPose	93%	25	Поддержка 3D-реконструкции
OpenPose	91% (21 точка)	10	Тяжелая модель, требует GPU
HSE GestureNet	89% (20 жестов)	15	Компактный размер (5 МБ), требует дообучения

Для проекта была выбрана библиотека MediaPipe Hands по следующим причинам :

- готовая интеграция с JavaScript через CDN (`@mediapipe/hands`);
- высокая производительность даже на слабых устройствах;
- отсутствие серверной нагрузки – обработка происходит на клиенте;
- поддержка нескольких рук одновременно;
- кросс-платформенность (работает в любом современном браузере);
- библиотека MediaPipe обеспечивает высокую точность распознавания жестов в реальном времени [16].

Альтернатива в виде HSE GestureNet потребовала бы:

- дополнительного серверного API для инференса;
- создания датасета под конкретные реабилитационные упражнения;
- оптимизации для работы в реальном времени.

Таким образом, MediaPipe оптимально сочетает точность, производительность и простоту интеграции в веб-приложение.

1.4. Обоснование выбора программного подхода

Анализ существующих решений показывает, что аппаратные методы, несмотря на высокую точность, остаются малодоступными из-за стоимости и сложности эксплуатации. VR-системы также требуют значительных затрат на оборудование, а мобильные приложения не обеспечивают достаточной обратной связи.

В связи с этим программный подход на основе веб-технологий и компьютерного зрения (MediaPipe) представляется оптимальным по следующим причинам.

1. Доступность – не требует специализированного оборудования, только ПК/ноутбук с камерой.
2. Масштабируемость – веб-приложение может использоваться одновременно множеством пациентов.

3. Точность – современные библиотеки (MediaPipe, OpenCV) позволяют анализировать движения с достаточной для реабилитации точностью.
4. Адаптивность – упражнения в будущем можно будет корректировать в зависимости от прогресса пациента.

2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ВЕБ-ПРИЛОЖЕНИЯ

2.1. Анализ требований к системе

Разработка адаптивного веб-приложения для реабилитации мелкой моторики рук предъявляет комплекс требований к системе, которые можно разделить на функциональные и нефункциональные. Ключевым функциональным требованием является обеспечение точного распознавания жестов и движений кисти в реальном времени с использованием исключительно встроенных возможностей веб-браузера. Это подразумевает реализацию механизма захвата видеопотока с веб-камеры средствами JavaScript без применения дополнительных библиотек компьютерного зрения, таких как OpenCV. Use-Case диаграмма представлена на рисунке 1.

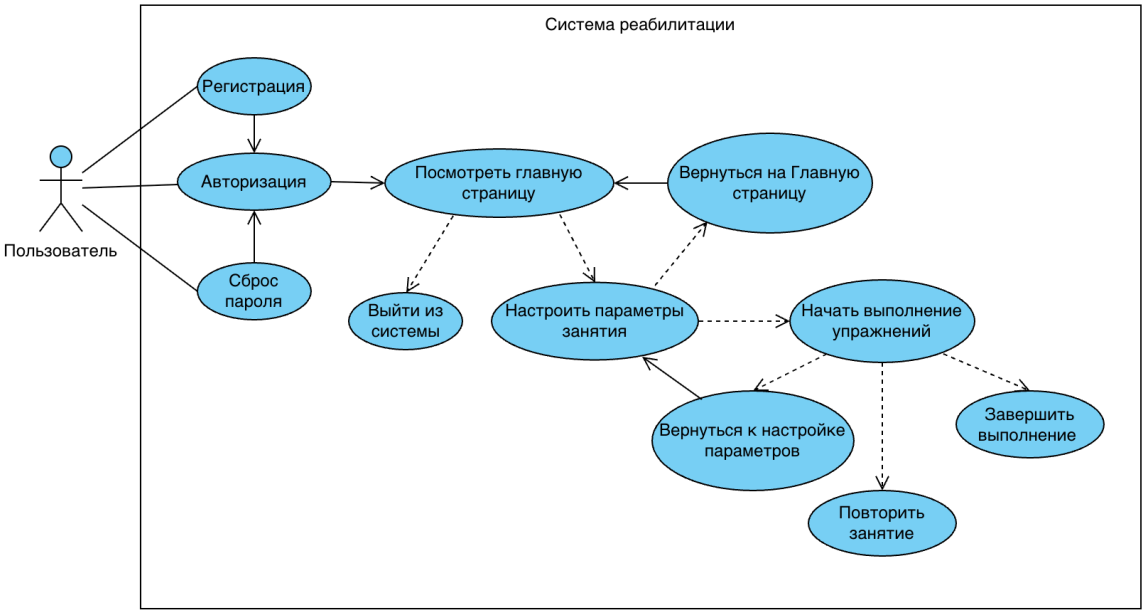

Рисунок 1. Use-Case диаграмма

Важным аспектом является обеспечение модульности системы, что позволяет выделить сервис аутентификации и сервис упражнений в независимые компоненты. Такое разделение обусловлено различной природой этих функциональных блоков и необходимостью их независимого

масштабирования. Сервис аутентификации должен обеспечивать безопасное хранение пользовательских данных и поддерживать механизм JWT-авторизации, в то время как сервис упражнений отвечает за обработку данных о выполнении реабилитационных заданий.

2.2. Сравнительный анализ архитектурных подходов

При проектировании системы рассматривались две принципиально разные архитектурные парадигмы: монолитная и микросервисная. Монолитная архитектура, при всей своей простоте реализации и развертывания, демонстрирует существенные ограничения в контексте разрабатываемого приложения. Основным недостатком монолитного подхода является высокая степень связанности компонентов, что затрудняет внесение изменений в отдельные функциональные модули и ограничивает возможности масштабирования системы.

Микросервисная архитектура, напротив, предоставляет значительные преимущества для данного проекта. Разделение системы на независимые сервисы позволяет оптимизировать нагрузку на отдельные компоненты и обеспечивает гибкость при дальнейшем расширении функциональности. В частности, выделение сервиса аутентификации в отдельный модуль повышает безопасность системы и упрощает реализацию механизмов авторизации. При этом важно отметить, что выбор микросервисной архитектуры потребовал решения дополнительных задач, связанных с организацией межсервисного взаимодействия и обеспечением согласованности данных. Преимущества микросервисной архитектуры для медицинских систем подробно описаны в работе [17].

2.3. Обоснование выбора технологического стека

Технологический стек проекта был подобран с учетом специфических требований к системе. Для реализации серверной части был выбран фреймворк Django в сочетании с Django REST Framework. Этот выбор обусловлен наличием встроенной системы аутентификации, поддержкой

ORM для работы с базой данных и высокой производительностью при обработке API-запросов [18]. В качестве клиентского фреймворка был выбран React, который благодаря компонентной архитектуре и эффективному механизму виртуального DOM обеспечивает необходимую интерактивность пользовательского интерфейса.

Особое внимание было уделено выбору технологии для распознавания жестов. Библиотека MediaPipe Hands, доступная через CDN (https://cdn.jsdelivr.net/npm/@mediapipe/hands/), была выбрана благодаря своей способности работать непосредственно в браузере без необходимости серверной обработки видеопотока. Это решение обеспечивает распознавание 21 ключевой точки на кисти с частотой до 30 кадров в секунду, что полностью удовлетворяет требованиям реабилитационных упражнений. Для контейнеризации сервисов был выбран Docker, который обеспечивает изоляцию компонентов и упрощает процесс развертывания системы [19].

2.4. Организация межсервисного взаимодействия

Взаимодействие между компонентами системы организовано через REST API с использованием JSON-формата данных. Сервис аутентификации предоставляет эндпоинты для регистрации и авторизации пользователей, возвращая JWT-токены для последующей аутентификации запросов. Механизм JWT был выбран благодаря его статистической природе, что позволяет избежать необходимости хранения состояния сессии на сервере [20, 21, 22].

Клиентское приложение, построенное на React, взаимодействует с сервисом упражнений через защищенные HTTPS-соединения. Особенностью архитектуры является то, что обработка видеопотока и распознавание жестов полностью осуществляется на клиентской стороне с использованием MediaPipe Hands, что значительно снижает нагрузку на серверную часть. Данные о выполнении упражнений передаются на сервер только после

завершения сеанса реабилитации, что оптимизирует использование сетевых ресурсов.

2.5. Модель данных и проектирование API

На данный момент модель данных системы сущность пользователя. Дальнейшая разработка предполагает включение трех основных сущностей: пользователя, упражнение и сессию выполнения упражнения. Пользовательская сущность содержит информацию для аутентификации и персональные данные. Сущность упражнения описывает параметры реабилитационных заданий, включая степень сложности и целевую группу мышц. Сущность сессии фиксирует результаты выполнения упражнения конкретным пользователем.

API системы спроектировано в соответствии с принципами REST и включает набор эндпоинтов для работы с каждой сущностью. Для обеспечения безопасности все запросы, кроме операций регистрации и аутентификации, требуют наличия валидного JWT-токена в заголовке Authorization. Особенностью API является его ориентация на работу с клиентским приложением, что выражается в оптимизированных ответах, содержащих только необходимые для отображения данные [22].

2.6. Общая архитектура решения

Архитектура разрабатываемого веб-приложения построена по принципам микросервисной модели и предполагает логическое разделение компонентов на независимые сервисы: клиентское приложение, сервис аутентификации и сервис упражнений. Такая структура упрощает поддержку, масштабирование и развитие функциональности системы. Обработка видеопотока и распознавание жестов полностью выполняются на клиентской стороне, что снижает нагрузку на сервер и обеспечивает отзывчивость интерфейса.

Ниже представлена диаграмма компонентов архитектуры (рис. 2).

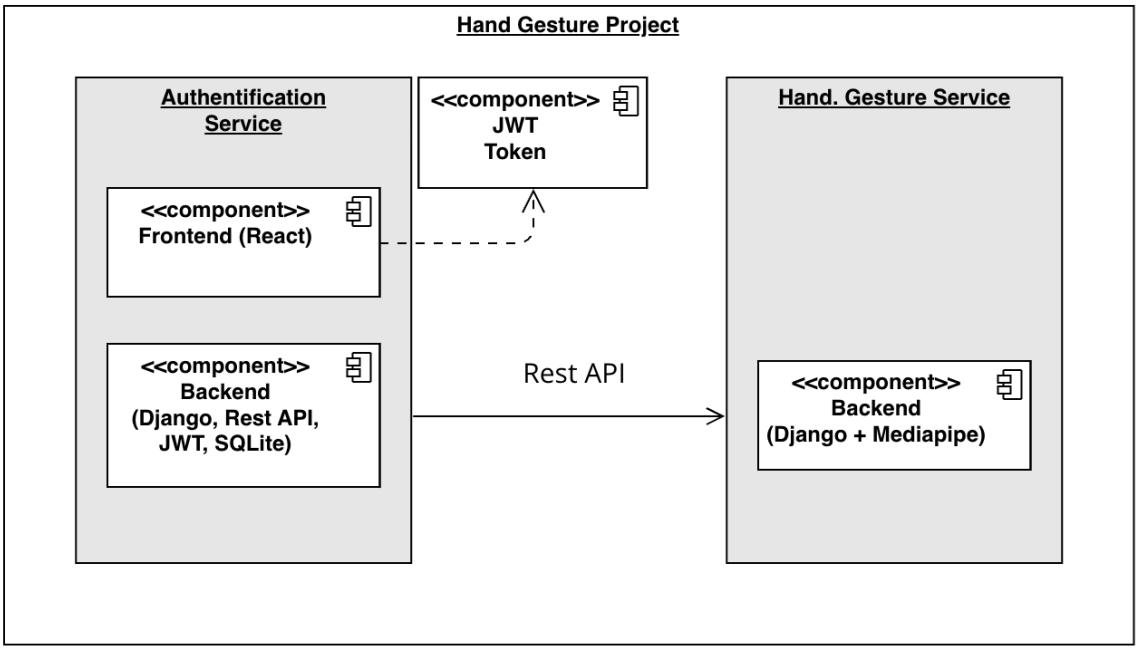

Рисунок 2. Диаграмма компонентов архитектуры веб-приложения

Комментарии к диаграмме:

- веб-клиент реализован на React с использованием библиотеки MediaPipe для захвата и анализа движений руки в реальном времени. Все вычисления, связанные с обработкой видеопотока, выполняются в браузере;
- сервис аутентификации — отдельный компонент на Django, реализующий регистрацию, вход и выдачу JWT-токенов;
- сервис упражнений — обрабатывает данные сессий упражнений, сохраняет результаты и управляет реабилитационным контентом;
- SQLite используется в качестве базы данных. Этот легковесный встроенный движок хранения выбран из-за простоты использования и отсутствия необходимости в развертывании отдельного сервера базы данных, что особенно удобно на ранних этапах разработки и тестирования.

Архитектура обеспечивает изоляцию компонентов, минимизирует точки отказа и позволяет легко масштабировать систему в будущем, при

необходимости заменив SQLite на более мощную СУБД (например, PostgreSQL или MySQL) без изменений в бизнес-логике.

3. РЕАЛИЗАЦИЯ МИКРОСЕРВИСОВ

3.1. Сервис аутентификации и авторизации

Сервис аутентификации представляет собой ключевой компонент системы, обеспечивающий безопасное управление пользовательскими данными. Реализация выполнена на базе Django REST Framework с использованием JWT (JSON Web Tokens) в качестве основного механизма аутентификации. Архитектура сервиса построена по принципу RESTful API, что обеспечивает его совместимость с различными клиентскими приложениями[20].

3.1.1. Регистрация, вход и восстановление пароля

Процесс регистрации пользователя реализован через эндпоинт /api/auth/register, принимающий обязательные параметры: электронную почту и пароль. При валидации данных система проверяет уникальность email и соответствие пароля требованиям сложности (минимум 8 символов, включая цифры и специальные знаки). После успешной регистрации создается запись в базе данных SQLite в зашифрованном виде с использованием алгоритма PBKDF2 с SHA-256 с хэшированием, что обеспечивает необходимый уровень безопасности[22].

Для входа в систему используется эндпоинт /api/auth/login, который при корректных учетных данных возвращает пару токенов:

- Access Token (срок действия 15 минут);
- Refresh Token (срок действия 24 часа).

Механизм восстановления пароля включает:

1. генерацию уникального токена с привязкой к пользователю;
2. отправку письма с ссылкой для сброса через SMTP-сервер (например, SendGrid) (см. рис. 3);
3. валидацию токена при переходе по ссылке;
4. обновление пароля через эндпоинт /api/auth/reset-password.

```python
class PasswordResetRequestView(APIView):
    murka
    @swagger_auto_schema(
        request_body=PasswordResetSerializer,
        operation_description="Сброс пароля (отправка email)",
        responses={200: 'Если email зарегистрирован, письмо отправлено.'}
    )
    def post(self, request):
        load_dotenv()
        email = request.data.get('email')
        user = User.objects.filter(email=email).first()
        if user:
            uid = urlsafe_base64_encode(force_bytes(user.pk))
            token = default_token_generator.make_token(user)
            reset_url = f"http://localhost:3000/password-reset-confirm/{uid}/{token}/"
            send_mail(
                subject: 'Сброс пароля',
                message: f'Перейдите по ссылке для сброса пароля: {reset_url}',
                os.getenv('EMAIL_HOST_USER'),
                recipient_list: [email],
            )
        return Response( data: {'message': 'Если email зарегистрирован, письмо отправлено.'}, status=status.HTTP_200_OK)
```

Рисунок 3. Пример кода Django для отправки письма с токеном

3.1.2. JWT-аутентификация

Система использует библиотеку Simple JWT, которая реализована с использованием стандартных возможностей DRF [22]:

- подпись токенов по алгоритму HS256;
- автоматическую проверку срока действия;
- механизм обновления Access Token через Refresh Token.

Схема работы сервиса (см. рис. 4):

1. клиент отправляет credentials (email + пароль);
2. сервер проверяет их и выдает токены;
3. все последующие запросы содержат Access Token в заголовке Authorization: Bearer <token>;
4. при истечении срока Access Token клиент запрашивает новый через Refresh Token.

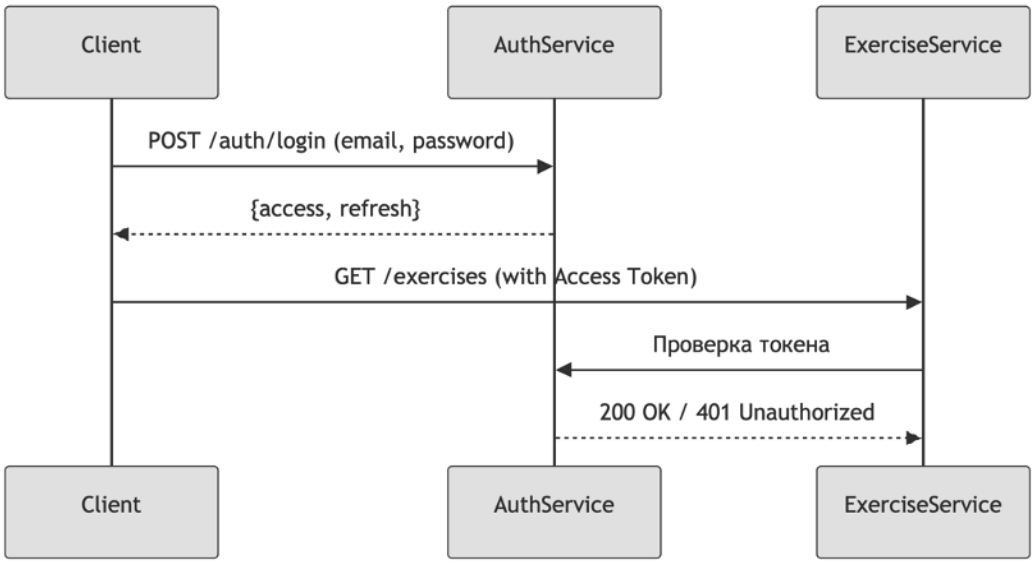

Рисунок 4. Схема работы сервиса

3.1.3. Интеграция с фронтендом (React)

Интеграция сервиса аутентификации с клиентской частью на React реализована через систему маршрутизации и централизованное управление состоянием аутентификации. Основой интеграции служит компонент App, который выполняет функции корневого элемента приложения и обеспечивает следующие ключевые аспекты взаимодействия.

1. Управление состоянием аутентификации.

Состояние аутентификации контролируется через механизм React-состояния с использованием хука useState. Токен доступа сохраняется в локальном хранилище браузера (localStorage), что позволяет сохранять сеанс пользователя между перезагрузками страницы. При инициализации приложения выполняется проверка наличия токена в localStorage, и его значение устанавливается в начальное состояние компонента App.

2. Система маршрутизации.

Приложение использует библиотеку react-router-dom версии 6 для организации навигации. Маршрутизация построена по принципу защищенных маршрутов (protected routes), где доступ к определенным страницам (например, /profile) ограничен для неавторизованных пользователей. Реализация защиты маршрутов выполнена через условный рендеринг - если

токен отсутствует, пользователь перенаправляется на страницу входа с помощью компонента Navigate.

3. Обработка выхода из системы.

Функция handleLogout обеспечивает завершение сеанса пользователя путем удаления токена из localStorage и обновления состояния приложения. Это приводит к автоматическому перенаправлению на страницу входа благодаря механизму защищенных маршрутов.

4. Передача параметров между компонентами.

Система реализует однонаправленный поток данных:

- компонент Login получает callback-функцию setToken для обновления состояния аутентификации;
- компонент Profile получает текущий токен и функцию выхода из системы;
- компоненты восстановления пароля работают с динамическими параметрами маршрута (uid и token).

5. Интеграция с UI-библиотеками.

Приложение использует Bootstrap для базовых стилей, что обеспечивает согласованный визуальный интерфейс всех компонентов аутентификации. Дополнительные кастомные стили подключаются через файл index.css.

Особенностью реализации является отсутствие глобального контекста аутентификации - вместо этого состояние передается явно через props. Такой подход упрощает отслеживание потока данных в приложении, хотя и может потребовать модификации при значительном увеличении количества защищенных маршрутов.

Архитектура интеграции демонстрирует следующие преимущества:

- ясное разделение ответственности между компонентами;
- простота отслеживания изменений состояния аутентификации;
- легкость тестирования отдельных компонентов;
- минимальная зависимость от сторонних библиотек управления состоянием.

3.2. Сервис выполнения упражнений

Сервис выполнения упражнений представляет собой ключевой функциональный модуль системы, обеспечивающий распознавание жестов верхних конечностей и контроль за правильностью выполнения реабилитационных упражнений. В основе сервиса лежит оригинальная методика классификации жестов, разработанная специально для задач восстановления мелкой моторики.

3.2.1. Работа с MediaPipe: распознавание жестов

Для распознавания жестов используется библиотека MediaPipe Hands, интегрированная в клиентскую часть приложения через CDN [23, 24]. Особенностью реализации является обработка видеопотока непосредственно в браузере без передачи данных на сервер, что обеспечивает конфиденциальность пользовательской информации и снижает нагрузку на сетевые ресурсы.

Библиотека MediaPipe Hands предоставляет 21 ключевую точку для каждой кисти, включая координаты суставов и кончиков пальцев. На основе этих данных разработан алгоритм определения положения пальцев, который учитывает углы между фалангами и их ориентацию относительно ладони. Для повышения точности распознавания введены дополнительные проверки, исключающие ложные срабатывания при частичном перекрытии пальцев или нестандартном положении кисти.

3.2.2. Процесс получения изображения с камеры и формирования упражнений

Обработка видеопотока осуществляется через специальный класс Camera, который инкапсулирует работу с API MediaDevices, что позволяет получать изображение с веб-камеры пользователя без использования дополнительных библиотек. Хотя прямое обращение к

navigator.mediaDevices.getUserMedia() не отражено в коде, класс Camera выполняет аналогичные функции, обеспечивая:

- инициализацию видеопотока с разрешением 640x480 пикселей;
- регулярный захват кадров через callback-функцию onFrame;
- передачу видеокадров в обработчик MediaPipe Hands.

Ключевые особенности реализации:

- видеоэлемент (videoElement) служит источником изображения;
- кадры передаются в MediaPipe в формате объекта {image: videoElement};
- обработка выполняется асинхронно через await hands.send().

Алгоритм обработки данных включает несколько этапов.

1. Предварительная фильтрация кадров: устранение шумов и нормализация освещенности с помощью алгоритмов бинаризации.
2. Определение ключевых точек: анализ положения суставов с использованием предобученной модели MediaPipe.

Классификация распознанного жеста реализована по принципу пятиразрядного числа, где каждый бит соответствует определенному пальцу (от большого к мизинцу). Значение бита "1" указывает на разогнутое положение пальца, "0" - на согнутое. Полученный двоичный код преобразуется в десятичное число, которое служит уникальным идентификатором жеста.

Количество и вариативность упражнений определяется на конфигурационной странице (configpage.html, см. рис. 5), которая реализует форму сбора параметров для генерации упражнений через систему радиокнопок и числового ввода. Форма включает четыре ключевых раздела конфигурации:

1. «Выбор руки»:
   - левая рука (значение "Left");
   - правая рука (значение "Right");
   - обе руки поочередно (значение "Multi").
2. «Количество пальцев»:

– один палец (значение 1);
– два пальца (значение 2);
– комбинированный режим (значение 5).

3. «Режим выполнения»:
   – стандартный последовательный (значение "std");
   – случайный порядок (значение "rnd").
4. «Длительность занятия»:
   – числовое поле ввода от 1 до 1000 упражнений.

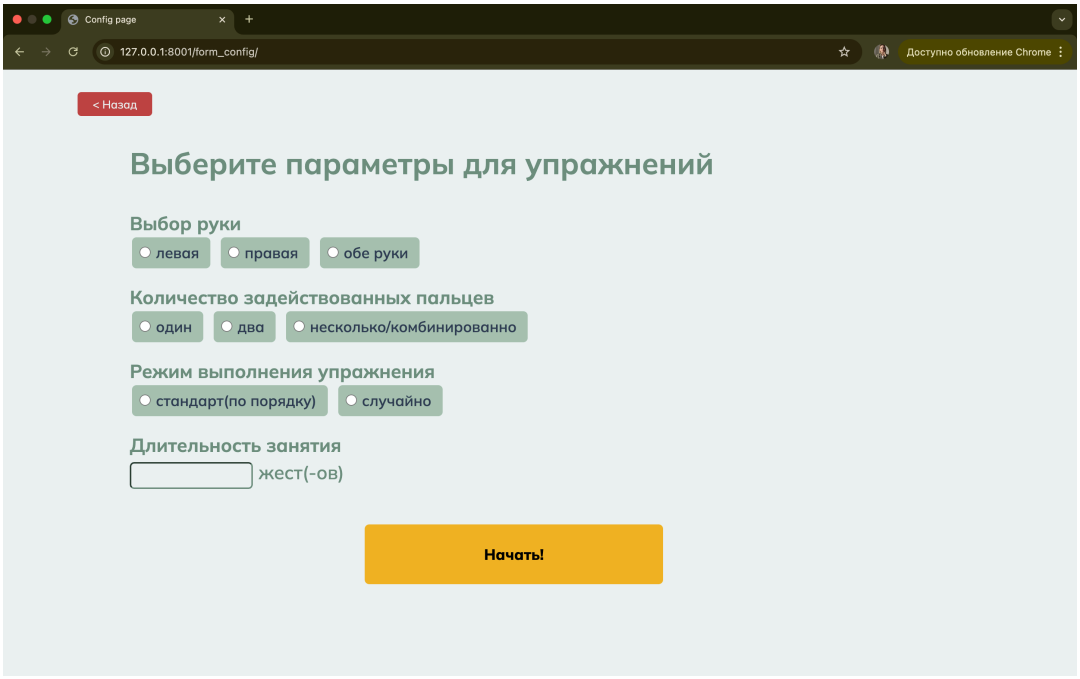

Рисунок 5. Форма конфигурации режима упражнений

3.2.3. Алгоритм генерации упражнений

После отправки формы параметры передаются на сервер методом POST и сохраняются в localStorage браузера. На основе этих параметров система формирует массив упражнений (exlist) по следующему сценарию.

Для режима одной руки (Left/Right):

– при выборе 1 пальца используются значения из массива one_fingers_list [16, 8, 4, 1, 4, 8, 16];
– для 2 пальцев – two_fingers_list [3, 6, 9, 12, 17, 20, 24];

– для 5 пальцев – more_fingers_list [16, 24, 28, 31] или all_fingers в случайном режиме.

Для режима двух рук (Multi):

– используется базовый набор fist_and_palm [0, 31];
– дополнительно формируется hands_location_list с чередованием ["Left", "Right"].

Логика заполнения массива (см. рис. 6):

– в стандартном режиме (std) упражнения повторяются циклически;
– в случайном режиме (`rnd`) выбираются произвольные элементы.

```
if (hand_location == "Left" || hand_location == "Right"){
    switch (fingers){
        case 1:
            switch (mode){
                case "std":
                    for(let i=0; i<exercises; i++){
                        exlist.push(one_fingers_list[i % one_fingers_list.length]);
                    }
                    break;
                case "rnd":
                    for (let i = 0; i < exercises; i++) {
                        let randomItem = one_fingers_list[Math.floor(Math.random() * one_fingers_list.length)];
                        exlist.push(randomItem);
                    }
                    break;
            }
            break;
```

Рисунок 6. Пример заполнения массива для упражнений

3.2.4. Примеры конфигураций

Далее приведены примеры сформированных списков упражнений.

1. Базовая реабилитация:
   – правая рука;
   – 1 палец;
   – стандартный режим;
   – 10 упражнений.

   Результат: [16, 8, 4, 1, 4, 8, 16, 8, 4, 1]
2. Продвинутый уровень:
   – обе руки;

– комбинированный режим;

– случайный порядок;

– 5 упражнений.

Результат: [31, "Left"], [0, "Right"], [31, "Right"], [0, "Left"], [31, "Left"]

Визуализация процесса представлена на рисунке 6.

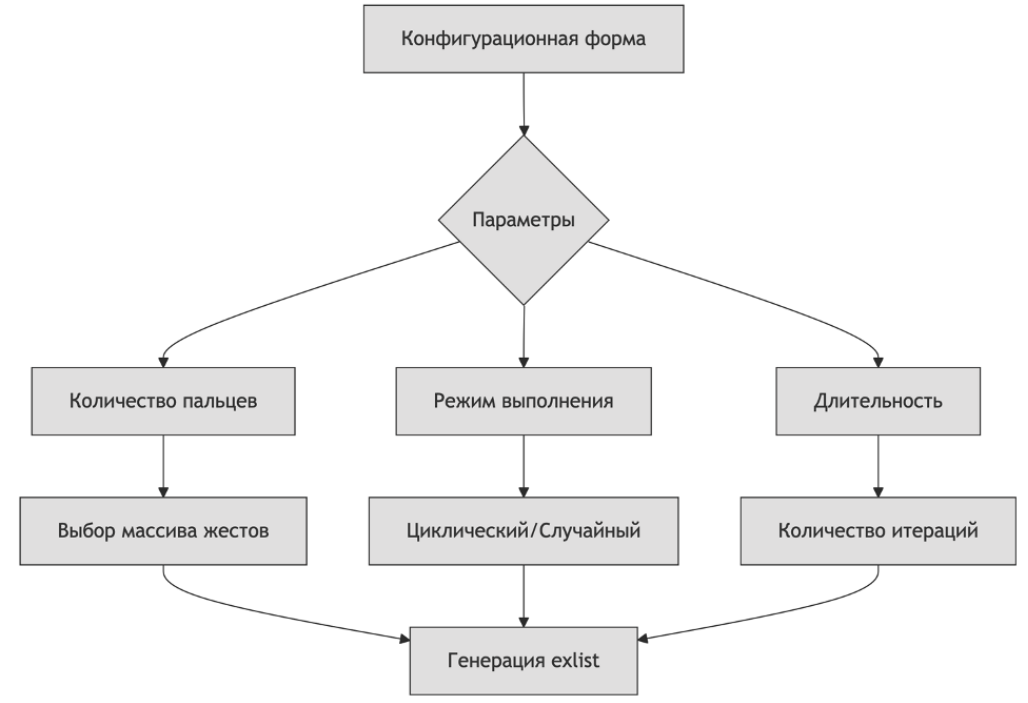

Рисунок 6. Визуализация процесса формирования списка упражнений

3.2.5. Особенности реализации

Среди особенностей реализации можно выделить следующие:

1. Динамическое обновление интерфейса:
   – изображение-образец меняется при переходе между упражнениями;
   – для левой руки применяется зеркальное отображение (см. рис. 7).

```
const imagePath = 'http://127.0.0.1:8001/static/hand_gesture/hands_img/'+exlist[0]+'.jpg';
const imageContainer = document.getElementById('image-container');
const img = document.createElement('img');
img.src = imagePath;
img.id = 'hand_image';
img.style.borderRadius = "6px";
if (hand_location == "Left" || hands_location_list[0] == "Left") {
    img.style.transform = "scale(-1, 1)";
}
imageContainer.appendChild(img);
```

Рисунок 7. Пример работы с картинками рук

2. Адаптивная обратная связь:
   – подсветка активной руки цветом (зеленый – верное поднята требуемая рука, красный – неверно) (см. рис. 8);
   – отображение счетчика оставшихся упражнений.

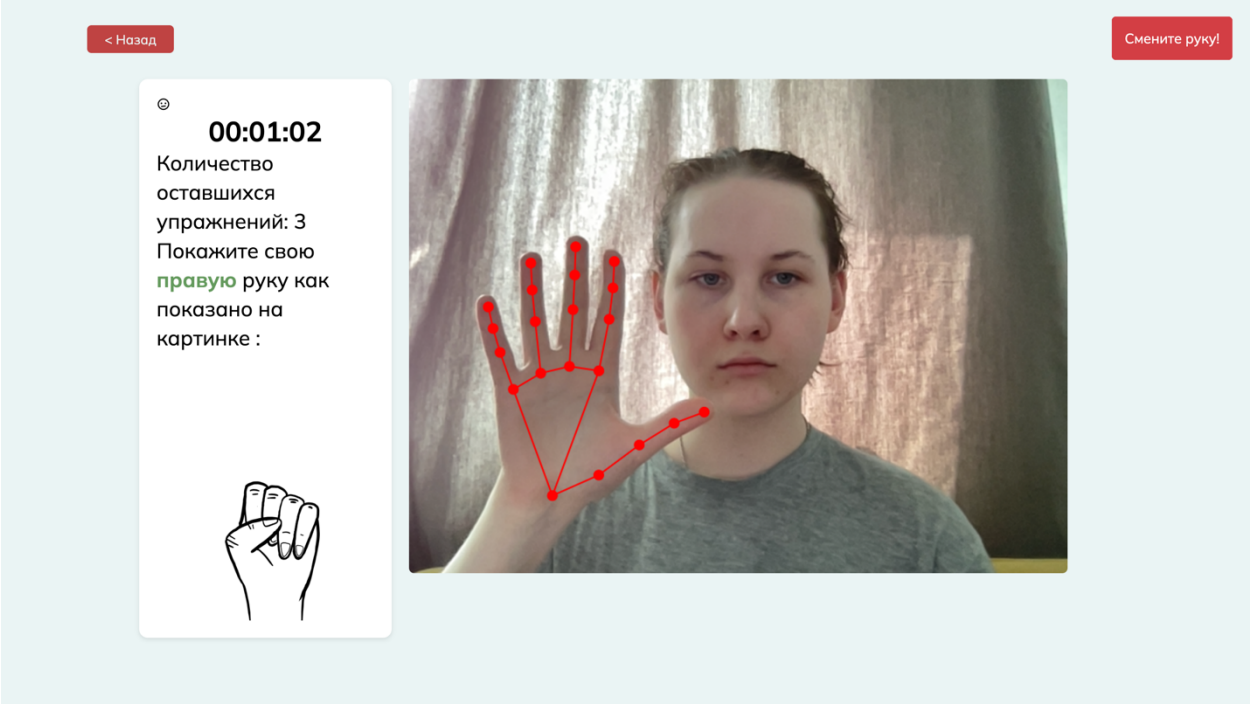

Рисунок 8. Пример отображения подсветки на руке

3. Гибкость системы:
   – возможность расширения путем добавления новых массивов жестов;
   – простота модификации логики генерации через изменение параметров формы.

Данный подход обеспечивает персонализированную настройку реабилитационных упражнений под индивидуальные потребности пациента, сохраняя при этом простоту интерфейса и гибкость системы.

3.3. Разработка пользовательского интерфейса

3.3.1. UX/UI-дизайн для пациентов

Пользовательский интерфейс системы разработан с учетом особых потребностей пациентов, проходящих реабилитацию мелкой моторики. В основе дизайна лежат принципы доступности (accessibility) и минимальной

когнитивной нагрузки. Цветовая схема построена на контрастных, но не агрессивных тонах, что обеспечивает комфортное восприятие для пользователей с возможными нарушениями зрения. Основные интерактивные элементы имеют размер не менее 48×48 пикселей, что соответствует рекомендациям WCAG 2.1 для пользователей с ограниченной моторикой [25, 26].

Интерфейс упражнений реализует принцип "одна задача — один экран", где все внимание пользователя фокусируется на выполнении текущего задания. Видеопоток с камеры отображается в зеркальном режиме, что упрощает процесс самоконтроля движений. Эталонное изображение жеста сопровождается анимацией плавного появления, что позволяет плавно переключать внимание между демонстрацией и собственными действиями.

Для обеспечения обратной связи используется многоуровневая система уведомлений:

- визуальные маркеры правильного положения пальцев;
- цветовая индикация распознанного жеста;
- текстовые подсказки о необходимости коррекции движений.

3.3.2. Взаимодействие с серверной частью

Архитектура взаимодействия клиентской части с серверными микросервисами построена по принципу "тонкого клиента". Основная бизнес-логика обработки жестов вынесена на клиентскую сторону, что минимизирует объем передаваемых данных. Обмен информацией с сервером осуществляется через REST API с использованием JSON-формата.

Особенностью реализации является гибридный подход к аутентификации:

1. первичная авторизация через JWT-токены;
2. хранение сессионных данных в HttpOnly cookies;
3. механизм автоматического обновления токенов.

Для обработки ошибок сети реализована система экспоненциальной повторной отправки запросов (exponential backoff), что особенно важно для пользователей с нестабильным интернет-соединением. Состояние интерфейса синхронизируется с сервером через механизм оптимистичных обновлений, когда изменения сначала отражаются в интерфейсе, а затем подтверждаются сервером.

3.4. Контейнеризация приложения

3.4.1. Контейнеризация сервисов (Docker)

Система развертывается в виде набора изолированных контейнеров, каждый из которых отвечает за определенный функциональный модуль [27, 28]. Для сервиса аутентификации используется образ на базе Python 3.9 с предустановленными зависимостями Django и DRF. Особое внимание уделено оптимизации размеров образов:

- многоэтапная сборка для исключения dev-зависимостей;
- использование alpine-образов для минимального размера;
- версионирование тегов для обеспечения воспроизводимости.

Контейнер с сервисом упражнений включает в себя:

- минимальную среду выполнения Python;
- предустановленные бинарные зависимости MediaPipe;
- конфигурационные файлы для работы с SQLite;
- скрипты инициализации базы данных.

3.4.2. Настройка docker-compose.yml

Конфигурация оркестрации контейнеров реализована через файл docker-compose.yml, который определяет три основных сервиса и общую сетевую инфраструктуру для их взаимодействия. Архитектура развертывания построена с учетом принципов изолированности сервисов и обеспечения надежного взаимодействия между ними.

Структура сервисов:

1. backend (Django):
   - собирается из контекста директории ./account-service/backend;
   - выполняет автоматические миграции базы данных при запуске;
   - запускает development-сервер на порту 8000;
   - монтирует рабочую директорию для hot-reload изменений;
   - работает в debug-режиме (DEBUG=1);
   - подключается к внутренней сети gesture-network.
2. frontend (React):
   - собирается из контекста ./account-service/frontend;
   - экспонирует порт 3000 для доступа к клиентскому приложению;
   - использует volumes для синхронизации кода и изоляции node_modules;
   - настроен для работы с hot-reload через polling (CHOKIDAR_USEPOLLING);
   - требует настройки tty и stdin_open для интерактивной работы.
3. hand-gesture (Django):
   - собирается из контекста ./hand_gesture_project;
   - выполняет миграции и запускает сервер на порту 8001;
   - монтирует рабочую директорию для разработки;
   - работает в debug-режиме;
   - подключается к общей сети gesture-network.

Сетевая конфигурация реализована через мостовую сеть bridge, которая обеспечивает:

- изолированное взаимодействие между контейнерами;
- автоматическое DNS-разрешение имен сервисов;
- возможность масштабирования отдельных компонентов.

Данная конфигурация обеспечивает гибкое развертывание всех компонентов системы одной командой docker-compose up, сохраняя при этом

возможность индивидуальной настройки каждого сервиса. Для production-среды рекомендуется дополнить конфигурацию настройками репликации, логгирования и мониторинга.

ЗАКЛЮЧЕНИЕ

Результаты работы

В ходе выполнения работы над проектом была успешно разработана веб-система для восстановления мелкой моторики рук, основанная на микросервисной архитектуре. Были решены следующие задачи.

1. Выполнен анализ современных методов реабилитации мелкой моторики и существующих программных решений, в ходе которого выявлены основные тенденции цифровизации в реабилитации, классифицированы существующие методы (традиционные, аппаратные, программные), а также выделены преимущества и ограничения каждого подхода. Сделан вывод о наибольшей перспективности программных решений, основанных на компьютерном зрении, за счёт их доступности, масштабируемости и потенциальной эффективности.
2. Проанализированы требования к приложению, на основе которых были сформулированы ключевые функциональные и нефункциональные характеристики системы. Установлена необходимость обеспечения работы в браузере без серверной обработки видеопотока, что повлияло на выбор архитектуры и технологий. В результате было принято решение о реализации микросервисной архитектуры с акцентом на клиентскую обработку данных, что позволило достичь высокой модульности, масштабируемости и эффективности системы. Выработана микросервисная архитектура приложения, обеспечивающая модульность и масштабируемость.
3. Разработана оригинальная методика классификации жестов:
   - бинарное кодирование положения пальцев;
   - адаптивный алгоритм подбора упражнений;
   - система визуальной обратной связи.

4. Создано приложение, включающее:
   - точное распознавание жестов с использованием MediaPipe Hands;
   - интуитивно понятный интерфейс для пациентов с ограниченными возможностями на основе React;
   - безопасную систему аутентификации на основе JWT.
5. Реализовано отказоустойчивое развёртывание, в основе которого:
   - изолированные микросервисы в Docker-контейнерах;
   - автоматизация развертывания на основе Docker-compose;
   - документированное API с поддержкой Swagger UI.

Перспективы развития проекта

Дальнейшее развитие системы предполагает следующие направления совершенствования.

1. Расширение функциональности:
   - интеграция с медицинскими CRM-системами;
   - добавление телемедицинских консультаций;
   - разработка мобильной версии приложения.
2. Улучшение алгоритмов распознавания:
   - внедрение 3D-анализа движений кисти;
   - использование нейросетевых моделей для сложных случаев;
   - добавление коррекции жестов в реальном времени.
3. Оптимизация производительности:
   - переход на gRPC для межсервисного взаимодействия;
   - кэширование результатов распознавания;
   - поддержка аппаратного ускорения.
4. Научно-исследовательские направления:
   - адаптация системы для других видов реабилитации;
   - интеграция с VR-технологиями и Smart-часами;
   - разработка персонализированных алгоритмов восстановления.

Разработанная система открывает новые возможности для цифровизации реабилитационного процесса, предлагая доступную альтернативу дорогостоящему оборудованию. Проект имеет значительный потенциал для внедрения в медицинских учреждениях и для домашнего использования, что может существенно улучшить качество реабилитации пациентов с нарушениями мелкой моторики.

Перспективным направлением является коммерциализация проекта через создание SaaS-решения для клиник и реабилитационных центров, а также разработка франшизной модели для тиражирования системы в регионах.

СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Можейко Е. Ю. Восстановление когнитивных нарушений и тонкой моторики после инсульта с использованием компьютерных программ и принципа биологической обратной связи: дис. ... д-ра мед. наук: 14.01.11 / Можейко Елена Юрьевна [КрасГМУ]. — Красноярск, 2014. — 238 с.
2. Екушева Е. В., Комазов А. А. Нарушение тонкой моторики кисти после латерализованного инсульта: процессы нейропластичности и сенсомоторной интеграции // Клиническая практика. — 2021. — № 1. — С. 35–42.
3. Воробьев А. А., Петрухин А. В., Засыпкина О. А., Кривоножкина П. В. Экзоскелет — состояние проблемы и перспективы внедрения в систему абилитации и реабилитации инвалидов (аналитический обзор) // Вестник Волгоградского государственного медицинского университета. — 2014. — № 1. — С. 20–26.
4. ЭкзоАтлет: история бизнеса по производству медицинских экзоскелетов // Официальный сайт компании «ЭкзоАтлет». — 2023.
5. В России начинается выпуск дешевых промышленных экзоскелетов // CNews. — 2020.
6. Шошина Е. А. Микросервисная архитектура: как её использование влияет на IT-решения // Молодой ученый. — 2025. — № 1 (552). — С. 8–9.
7. Селезнёв А. И., Селезнёв И. Л. Актуальность применения микросервисной архитектуры в системах обработки данных // Молодой ученый. — 2023. — № 48 (495). — С. 22–32.
8. Лазаренко Н.Н., Ильин В.С., Супова М.В. Многоканальная электростимуляция и лечебная физкультура в реабилитации пациентов с парезами мышц гортани // Санаторно-курортное оздоровление, лечение и реабилитация больных социально значимыми и профессиональными заболеваниями: материалы VI Междунар. конгр.,

Сочи, 07–10 окт. 2014 г. Сочи, 2014. С. 118–120. URL: https://repository.rudn.ru/ru/records/article/record/20737

9. Баранова Е.А., Магадяйка А.В., Киселёв А.В., Сидорова Н.В. Современные подходы к роботизированной механотерапии с элементами биоуправления и телемедицины для восстановления утраченных двигательных функций // Вестник Томского государственного университета. 2018. № 433. С. 127–134. URL: https://elibrary.ru/item.asp?id=36387785ТПУ Портал+1mgounb.ru+1
10. Белова А.Н., Клочкова А.С., Черникова Л.А., Пирадов М.А. Роботизированные устройства в нейрореабилитации: состояние вопроса // Вестник восстановительной медицины. 2023. Т. 22, № 1. С. 45–52. URL: https://journals.eco-vector.com/2078-1962/article/view/609457Eco-Vector Journals Portal+3Eco-Vector Journals Portal+3Eco-Vector Journals Portal+3
11. Петрова М.В., Рыжова О.В., Чебоксаров Д.В., Саенко И.В., Суева В.С., Петриков С.С. Перспективы применения VR-технологий в ранней реабилитации пациентов с острым нарушением мозгового кровообращения // Физическая и реабилитационная медицина, медицинская реабилитация. 2023. Т. 5, № 2. С. 157–166. URL: https://journals.eco-vector.com/2658-6843/article/view/405659Eco-Vector Journals Portal+3Eco-Vector Journals Portal+3Eco-Vector Journals Portal+3
12. Клочков А.С., Хижникова А.Е., Котов-Смоленский А.М., Супонева Н.А., Черникова Л.А., Пирадов М.А. Эффективность двигательной реабилитации при постинсультном парезе руки с помощью системы биологической обратной связи «Habilect» // Вестник восстановительной медицины. 2018. Т. 17, № 2. С. 41–45. URL: https://journals.eco-vector.com/2078-1962/article/view/609457Eco-Vector Journals Portal+1Eco-Vector Journals Portal+1

13. Салов Д.С., Седых Н.В. Адаптивная двигательная рекреация детей дошкольного возраста с нарушением интеллекта // Парадигмы аппроксимации данных в науке и практике: современное состояние и перспективы развития: сб. науч. ст. по итогам Междунар. межвуз. студенч. науч.-практ. конф. по результатам НИРС и магистрантов, Санкт-Петербург, 18–20 дек. 2020 г. Санкт-Петербург, 2020. С. 20–22. URL: https://elibrary.ru/item.asp?id=46150136mgounb.ru
14. Анализ и классификация реабилитационных робототехнических систем на базе параллельных манипуляторов // КиберЛенинка. URL: https://cyberleninka.ru/article/n/analiz-i-klassifikatsiya-reabilitatsionnyh-robototehnicheskih-sistem-na-baze-parallelnyh-manipulyatorovКиберЛенинка+1КиберЛенинка+1
15. Роботизированные технологии в физической реабилитации спортсменов с последствиями позвоночно-спинномозговых травм // КиберЛенинка. URL: https://cyberleninka.ru/article/n/robotizirovannye-tehnologii-v-fizicheskoy-reabilitatsii-sportsmenov-s-posledstviyami-pozvonochno-spinnomozgovyh-travmКиберЛенинка
16. Verma, A. R., Singh, G., Meghwal, K., Ramji, B., & Dadheech, P. K. (2024). *Enhancing Sign Language Detection through Mediapipe and Convolutional Neural Networks (CNN)*. arXiv preprint arXiv:2406.03729.
17. Barabanov A., Makrushin D. Authentication and authorization in microservice-based systems: survey of architecture patterns. arXiv preprint arXiv:2009.02114. 2020.
18. Петров С. А. Технические особенности Django Framework 3.0 при построении системы ритейлинга со встроенными модулями анализа и прогнозирования // Современные технологии. Системный анализ. Моделирование. – 2020. – № 3 (67). – С. 45–52. URL: https://elibrary.ru/item.asp?id=43808171
19. Чиганов Д. Р. Docker: ключ к контейнеризации и масштабируемости // Вестник науки. – 2023. – № 2. – С. 45–50.

URL: https://cyberleninka.ru/article/n/docker-klyuch-k-konteynerizatsii-i-masshtabiruemosti

20. Лазуков Д. А. Обзор методов аутентификации в REST WEB API // Безопасность информационного пространства — 2017: XVI Всероссийская научно-практическая конференция студентов, аспирантов, молодых ученых. Екатеринбург, 12 декабря 2017 года. — Екатеринбург: Изд-во Урал. ун-та, 2018. — С. 264–266. URL: http://elar.urfu.ru/handle/10995/65651
21. Рябов Н. В., Шавловский М. В., Подкопаев В. В. Управление доступом к защищенным ресурсам при помощи JSON Web Tokens // Системный анализ в науке и образовании. — 2021. — № 3. — С. 44–50. URL: https://sanse.ru/index.php/sanse/article/view/181.
22. Девицына С. Н., Пилькевич П. В., Удод Е. В. Способы улучшения защищённости сервисов, использующих JWT-токены // Экономика. Информатика. — 2023. — Т. 50, № 1. — С. 144–151. DOI: https://doi.org/10.52575/2687-0932-2023-50-1-144-151
23. Рындин Н. А. Технологии разработки клиентских WEB-приложений на языке JavaScript: учебное пособие / Н. А. Рындин. — Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2020. — 54 с. — ISBN 978-5-7731-0888-7.
24. Князев А. В., Баранов С. А. Распознавание жестов с использованием библиотеки MediaPipe // Сборник трудов Международной научной конференции студентов и аспирантов. — Муром: МИВлГУ, 2024. — С. 108–112.
25. Седельников И. А., Колтыгин Д. С. Разработка методики распознавания жестов для управления робототехническим комплексом // Научный вестник Братского государственного университета. — 2022. — № 72. — С. 54–59 DOI: 10.37539/221026.2022.72.54.001

26. Руководство по обеспечению доступности веб-контента (WCAG) 2.0 / Консорциум Всемирной паутины (W3C). — 2008. URL: https://www.w3.org/Translations/WCAG20-ru/
27. ГОСТ Р 52872-2019. Интернет-ресурсы. Требования доступности для инвалидов по зрению / Федеральное агентство по техническому регулированию и метрологии. — М.: Стандартинформ, 2019. URL: https://docs.cntd.ru/document/1200160584
28. Подорожный И. В., Светличный А. Н., Подлеснов А. В. Введение в контейнеры, виртуальные машины и Docker // Молодой ученый. — 2016. — № 19 (123). — С. 49–53. URL: https://moluch.ru/archive/123/33873/