Требования

к техническому заданию

на разработку ПО

"Фитнес-менеджер" (FitMan)

Оглавление

[Введение 3](#_Toc210197586)

[Цель проекта 3](#_Toc210197587)

[Общие сведения 3](#_Toc210197588)

[Краткое описание функциональных процессов 3](#_Toc210197589)

[Функциональные требования 5](#_Toc210197590)

[1. Основной функционал системы 5](#_Toc210197591)

[2. Роли пользователей и их функции в системе 7](#_Toc210197592)

[3. Сайт компании 9](#_Toc210197593)

[4. Онбординг (Lead Management) 11](#_Toc210197594)

[5. Файл настроек приложения 13](#_Toc210197595)

[6. Подсистема уведомлений (Notification Service) 14](#_Toc210197596)

[7. Подсистема чата (Messaging) 16](#_Toc210197597)

[8. Подсистема отчетности (Reporting & Analytics) 18](#_Toc210197598)

[9. Подсистема управления пользователями (User Management) 22](#_Toc210197599)

[10. Архитектура планов тренировок 39](#_Toc210197600)

[11. Расписание занятий 40](#_Toc210197601)

[12. Табели 41](#_Toc210197602)

[13. Подсистема рекомендаций по тренингу 42](#_Toc210197603)

[14. Команды (pattern Command) 42](#_Toc210197604)

[15. Каталоги (Catalogs) 42](#_Toc210197605)

[16. Оффлайн-режим (Offline-First Approach) 56](#_Toc210197606)

[17. Архивация и восстановление записей 58](#_Toc210197607)

[18. Аудит и журнал событий (Audit Log) 58](#_Toc210197608)

[Нефункциональные требования 60](#_Toc210197609)

[1. Требования к технологическому стеку 60](#_Toc210197610)

[2. Требования к таблицам 60](#_Toc210197611)

[3. Требования к списочным формам и карточкам 61](#_Toc210197612)

[4. Требования к логированию приложения 61](#_Toc210197613)

[5. Требования к безопасности 61](#_Toc210197614)

[6. Требования к тестированию 64](#_Toc210197615)

[7. Дополнительные требования 64](#_Toc210197616)

[Глоссарий 65](#_Toc210197617)

### Введение

#### Цель проекта

Создание мобильного и веб приложений для автоматизации процессов фитнес-центра с основным функционалом:

* ведение планов тренировок;
* контроль и учёт занятий;
* управление расписанием занятий:
* рекомендации по тренингу:
* контроль калорийного баланса клиентов;
* управление персоналом (тренеры, инструкторы, менеджеры, администраторы);
* ведение табелей сотрудников;
* ведение отчетности по клиентам, сотрудникам, занятиям, центру.

#### Общие сведения

| **Параметр** | **Значение** |
| --- | --- |
| Наименование | Фитнес-менеджер (FitMan) |
| Тип | Корпоративное приложение |
| Целевая аудитория | Фитнес-центры малого и среднего бизнеса |
| Платформы | iOS, Android, Web |
| Технологический стек | Flutter (Client) + Dart Frog/Shelf (Server) + PostgreSQL |
| Размещение | VPS без контейнеризации и baas |

#### Краткое описание функциональных процессов

**Клиенты** занимаются в фитнес-центре, согласно расписанию тренировок, под руководством назначенного тренера и инструктора.

**Тренер** составляет групповые и индивидуальные планы тренировок. Он проводит индивидуальные занятия, может задавать прогресс нагрузок на основе плана тренировок. Тренер осуществляет общее руководство занятиями прикрепленных клиентов.

**Инструктор** в основном проводит групповые занятия. Так же может помочь тренеру провести индивидуальное занятие, вести учет времени. Если он имеет права, может создавать планы тренировок, задавать прогресс и т.п., аналогично функционалу тренера.

У тренеров и инструкторов есть табель учета занятий, который составляется по факту проведения занятий.

**Менеджер** (куратор) осуществляет контроль за бизнес-процессами центра. Контролирует ведение журналов прикрепленных клиентов, инструкторов, тренеров. Также совместно с тренером ведет расписание тренировок. Назначенный дежурный менеджер ведет онбординг клиентов.

**Администратору** доступны все функции менеджера, также управляет системным функционалом, осуществляет контроль за системой.

### Функциональные требования

#### Основной функционал системы

* **Планирование и учет занятий.**

Составление расписания с учетом пожеланий клиентов. Клиент у себя в дашборде вводит желательные дни недели и время занятий. При запуске задачи составления общего нового расписания, учитываются данные предпочтений каждого клиента. Возможен вариант обновления расписания для отдельного клиента.

* **Учет и анализ полученных и затраченных калорий.**

План тренировок состоит из тренировочных дней или занятий. Занятие состоит из набора упражнений с повторами или временем.

У каждого упражнения есть среднее затрачиваемое количество калорий. У занятия, соответственно, сумма калорий упражнений. У плана тренировок - сумма калорий занятий.

Клиент, перед занятием, у себя в дашборде, вносит количество полученных калорий и свой вес. Полученные калории клиент высчитывает самостоятельно.

По завершению тренировки, система автоматически высчитывает затраченные калории, учитывая калорийность BMR (Basal Metabolic Rate, Основной обмен веществ) плюс калории, потраченные на занятии. Клиент может визуально сравнить приходящие калории и расходные калории в таблице в своей карточке. Клиент так же может открыть и посмотреть график зависимости калорий и веса от времени.

* **Система рекомендаций по плану тренировок на основе антропометрии, соматотипов, биоимпеданса.**

На основе значений антропометрии, соматотипии и биоимпеданса клиента, определяются его основные свойства – тип сложения, тип фигуры, качество тела. На основе свойств составляется его "портрет" и выдаются рекомендации по тренингу из базы данных. Так же рекомендации запрашиваются у ИИ в онлайн-режиме.

* **Индивидуальные планы тренировок на основе шаблонов.**

Система реализовывает модель разделения на **Шаблоны** и **Индивидуальные назначения** для поддержки кастомизации тренировок клиента. Такая архитектура позволяет тренеру гибко настраивать прогрессию нагрузок для каждого клиента, изменяя параметры в таблицах назначений, без изменения исходных шаблонов.

* **Подбор прогрессии нагрузок тренировок.**

При необходимости, тренером подбирается прогрессия тренировок: изменяются, вводятся или убираются отдельные упражнения в план, в зависимости от цели тренировок клиента. Например, вид плана тренировок – похудение, а у клиента наблюдается профицит калорий. Тренер может увеличить прогрессию тренировок путем увеличения подходов к снаряду, времени упражнений или замену упражнений.

* **Управление пользователями через роли.**

Роль представляет собой набор прав доступа и полномочий, позволяющих пользователю выполнять определенные действия в системе. Такой подход упрощает администрирование, обеспечивает единообразие и безопасность, так как позволяет назначать права группам пользователей, а не каждому отдельно.

В системе реализованы следующие роли: клиент, инструктор, тренер, менеджер, администратор.

Пользователь может иметь несколько ролей, что легко и гибко расширяет набор прав доступа. Например, для небольших фитнес-залов, где тренер является единственной штатной единицей, можно добавить ему роли менеджера и администратора. Или, как вариант, роли инструктора и менеджера отключить вообще.

#### Роли пользователей и их функции в системе

1. Клиент. Пользователь системы, занимающийся в фитнес-центре под руководством тренера.

Функции:

* Аутентификация;
* Ввод антропометрии. Ввод входящих калорий и веса. Ввод предпочтений занятий. Просмотр графиков калорий и веса, просмотр расписания.

1. Инструктор. Помощник тренера. Контролирует занятия, назначенных клиентов.

Функции:

* Аутентификация;
* Просмотр списка назначенных клиентов и открытие карточки клиента из списка;
* Ввод входящих калорий и входящего веса клиента. Просмотр графиков калорий и веса клиента;
* Просмотр расписания клиента, своего расписания и табеля;
* Коррекция плана занятия дня (списка упражнений);
* Фиксация начала и окончания занятия.
* Просмотр списка назначенных тренеров и открытие карточки из списка.

1. Тренер. Тренирует клиентов. Контролирует занятия, назначенных клиентов и инструкторов. Составляет планы тренировок, назначает планы клиентам.

Функции:

* Аутентификация;
* Просмотр списка назначенных клиентов и открытие карточки клиента из списка;
* Ввод входящих калорий и входящего веса клиента. Просмотр графиков калорий и веса клиента;
* Коррекция плана занятия дня (списка упражнений);
* Фиксация начала и окончания занятия.
* Доступ к кнопке "Подобрать прогрессии" в карточке клиента;
* Выбор прогрессии тренировок;
* Смена плана тренировок до первого занятия клиента;
* Просмотр списка прикрепленных инструкторов и открытие карточки из списка.
* Просмотр расписания клиента, своего расписания и табеля, расписание и табель закрепленного инструктора;

1. Менеджер. Контролирует тренировочные бизнес-процессы. Контроль за назначенными клиентами, инструкторами, тренерами.

Функции:

* Аутентификация;
* Ведение записей клиентов, инструкторов, тренеров.
* Ведение расписаний и табелей всех участников бизнес-процесса.
* Ведение обординга клиентов.

1. Администратор. Администрирует систему.

Функции:

* Аутентификация;
* Ведение (просмотр, добавление, удаление, изменение записей) каталогов.
* Ведение записей клиентов, инструкторов, тренеров, менеджеров.
* Ведение расписания работы центра.
* Редактирование настроек системы.

Управление доступом пользователя осуществляется на основе ролей.

В структуре проекта на каждую роль должен отводиться свой модуль-каталог.

#### Сайт компании

Сайт компании выполняет две ключевые функции:

* **Маркетинговую** для привлечение новых клиентов.
* **Служит точкой входа в веб-версию приложения** для существующих пользователей.

##### 1. Общие требования к дизайну и архитектуре

* **Стиль:** Строгий, лаконичный, корпоративный. Академичный подход с акцентом на надежность и профессионализм. Минимум декоративной графики.
* **Технологии:** Веб-сайт должен быть реализован как **Single Page Application (SPA) на Flutter Web**, что обеспечит единообразие дизайна и логики с мобильными приложениями.
* **Адаптивность:** Сайт должен корректно отображаться и функционировать на устройствах с разным разрешением экрана (десктоп, планшет, смартфон).
* **Навигация:** Интуитивно понятное главное меню, расположенное в шапке сайта. Меню должно быть доступно на всех страницах.

##### 2. Структура и содержание страниц

Сайт состоит из следующих обязательных страниц:

* **Главная страница («О компании»):**
  + Краткое описание фитнес-центра, его философия, преимущества.
  + Блок с ключевыми показателями (количество довольных клиентов, лет на рынке и т.д.).
  + Призыв к действию (CTA) "Оставить заявку".
  + Призыв к скачиванию apk мобильной версии на Андроид.
* **Контакты:**
  + Адрес центра с интерактивной картой (Google Maps/Яндекс.Карты).
  + Телефон, email, график работы.
  + Форма для обратной связи (упрощенный аналог блока "Составить заявку").
* **Расписание работы:** Информация о работе центра по дням недели. Данные должны подгружаться из каталога work\_schedules основной системы (только для чтения).
* **FAQ:** Страница с ответами на часто задаваемые вопросы.
* **Отзывы:** Модуль для отображения отзывов реальных клиентов.

##### 3. Функциональные блоки

###### 3.1. Блок «Составить заявку» (Форма лида)

Это ключевой маркетинговый элемент, доступный на главной странице и в виде всплывающего модального окна на других страницах.

* **Поля формы:**
  1. Имя (обязательное, текст).
  2. Пол (обязательное, выпадающий список: Мужской, Женский).
  3. Возраст (обязательное, число).
  4. Телефон (условно-обязательное, валидация формата).
  5. Email (условно-обязательное, валидация формата).
  6. Способ обратной связи (обязательное, радиокнопки: "По телефону", "По email"). Логика: если выбран телефон, обязательным становится поле "Телефон", и наоборот.
* **Логика работы:**
  1. **Валидация на стороне клиента:** При попытке отправить форму проверяется заполненность обязательных полей и корректность формата. Ошибки выводятся рядом с полями.
  2. **Отправка данных:** При нажатии кнопки "Отправить" данные формы отправляются методом POST на защищенный (HTTPS) API-эндпоинт бэкенда (например, /api/leads).
  3. **Обработка на сервере:** Бэкенд создает запись в таблице onboarding\_client и запускает процесс уведомления дежурного менеджера (как описано в разделе об онбординге).
  4. **Реакция на клиенте:** После успешной отправки форма скрывается, а пользователю показывается сообщение: "Спасибо! Ваша заявка принята. Наш менеджер свяжется с вами в ближайшее время.".

###### 3.2. Блок «Личный кабинет»

Это **веб-интерфейс основного приложения**, доступный после аутентификации.

* **Страница входа:**
  + Поля: "Логин (Email/Телефон)" и "Пароль".
  + Кнопка "Войти".
  + Ссылка "Забыли пароль?" (запускает стандартный процесс сброса через email).
* **Функциональность после входа:**
  + После успешной аутентификации пользователь перенаправляется на дашборд, соответствующий его роли (Клиент, Тренер, Менеджер, Администратор).
  + **Функциональность веб-дашбордов на 100% идентична функциональности мобильного приложения**, описанной в разделах "Функциональный модуль «Клиент»", «Тренер» и т.д.
  + Интерфейс должен быть оптимизирован для управления с помощью мыши и клавиатуры, но сохранять общую стилистику мобильного UI.

##### 4. Интеграция с основной системой

* **API:** Веб-сайт общается с бэкендом через те же **API и RESTful эндпоинты**, что и мобильное приложение.
* **Аутентификация:** Используется единая система аутентификации. Сессия, открытая в браузере, не должна конфликтовать с сессией в мобильном приложении.
* **Данные:** Данные для страниц "Расписание работы" и др. должны браться напрямую из БД основной системы, чтобы избежать дублирования и расхождения информации.

##### 5. Требования к контенту и SEO

* **Контент:** Все тексты на сайте должны быть уникальными, грамотными и отражать специфику центра.
* **Мета-теги:** Настройка title, description и keywords для каждой страницы.
* **Скорость загрузки:** Оптимизация изображений и кода для быстрой загрузки страниц, что критично для SEO и удержания пользователей.

#### Онбординг (Lead Management)

В данной системе онбординг - привлечение и активация клиентов.

Процесс превращения потенциального клиента в активного пользователя системы состоит из двух этапов: управления лидами (заявками) и активации учетной записи.

##### Этап 1: Управление лидами (Lead Management)

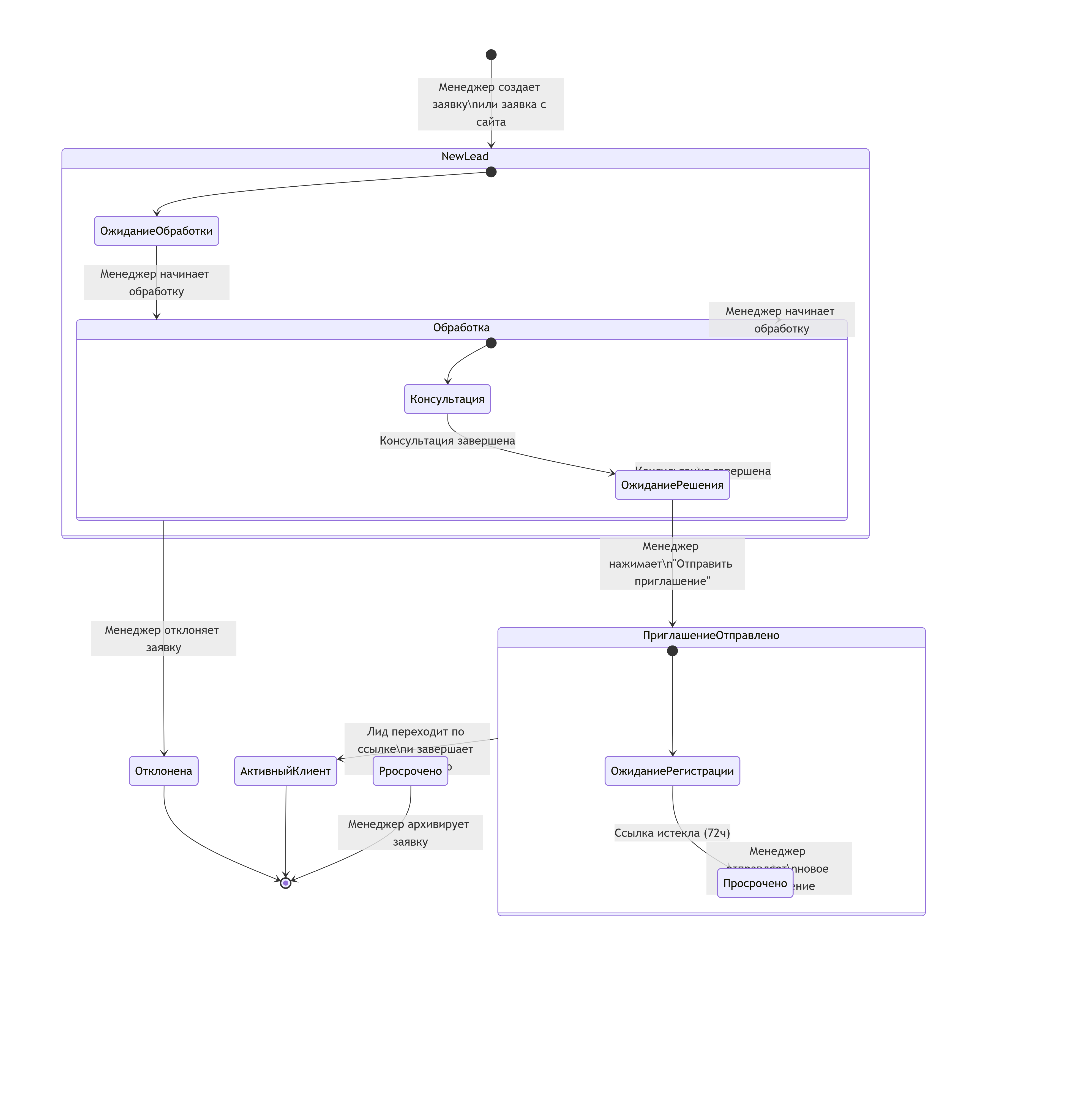
**1-й способ холодного контакта.** Потенциальный клиент оставляет заявку на сайте компании через контейнер "Составить заявку". Система автоматически создает запись в каталоге "Заявки (Лид-менеджмент)" (onboarding\_client) и уведомляет дежурного менеджера.  
**2-й способ холодного контакта.** Менеджер вручную создает запись в каталоге "Заявки" после телефонного разговора.  
Менеджер связывается с лидом для консультации. Результаты общения фиксируются в поле "Примечание менеджера" карточки лида.

##### Этап 2: Активация учетной записи клиента

После достижения договоренности менеджер в карточке лида нажимает кнопку **"Отправить приглашение"**.

1. **Генерация приглашения:** Система создает уникальную одноразовую ссылку-приглашение с JWT-токеном, содержащим id записи лида. Ссылка действительна **72 часа**.
2. **Отправка уведомления:** Ссылка отправляется лиду по тому каналу связи, который был указан как предпочтительный (email или SMS). Текст уведомления настраивается администратором.
3. **Завершение регистрации:** Лид переходит по ссылке, попадает на защищенную страницу, где должен задать свой пароль для системы. Пароль должен соответствовать политике безопасности.
4. **Создание клиента:** После успешной установки пароля система в рамках одной транзакции:
   * Создает полноценную учетную запись User с ролью "Клиент", используя данные из onboarding\_client.
   * Помечает запись в onboarding\_client как is\_client\_created = true.
   * Автоматически аутентифицирует пользователя в системе и перенаправляет его в личный кабинет.
5. **Обработка просрочки:** Если ссылка истекла, клиент должен обратиться к менеджеру для генерации новой.

##### Диаграмма состояний для онбординга



**Пояснения к диаграмме и процессу**

**Ключевые состояния:**

1. **NewLead (Новая заявка):** Начальное состояние. Заявка создана (через сайт или менеджером), но еще не взята в работу.
   * *Подсостояние «ОжиданиеОбработки»*: Заявка в очереди на менеджера.
   * *Подсостояние «Обработка»*: Менеджер активно работает с лидом (звонок, консультация).
2. **Обработка:** Менеджер связался с лидом, идет этап консультации.
   * *Подсостояние «Консультация»*: Идет общение.
   * *Подсостояние «ОжиданиеРешения»*: Менеджер ждет ответа от клиента или принимает решение.
3. **Отклонена (Archived):** Конечное состояние. Заявка неактивна (клиент не заинтересовался, не вышел на связь).
4. **ПриглашениеОтправлено:** Ключевое состояние. Менеджер подтвердил согласие клиента, система сгенерировала и отправила ссылку-приглашение.
   * *Подсостояние «ОжиданиеРегистрации»*: Ссылка активна, система ждет действия от клиента.
   * *Подсостояние «Просрочено»*: Ссылка истекла (прошло >72 часов). Требуется действие менеджера. Возможен вариант повторной отправки приглашения: создается новая ссылка с новым TTL, а старые инвалидируются.
5. **АктивныйКлиент (Client Created):** Конечное состояние, цель процесса. Лид успешно зарегистрировался в системе, учетная запись пользователя создана.

**Ключевые события (переходы):**

* **"Менеджер отправляет новое приглашение"** из состояния Просрочено позволяет повторно активировать заявку, не создавая новую запись.
* **"Лид переходит по ссылке"** — это автоматическое событие, инициируемое клиентом, которое запускает процесс регистрации и финального перехода в статус АктивныйКлиент.
* **"Менеджер архивирует заявку"** — это ручное действие менеджера для очистки базы от неактивных просроченных заявок.

#### Файл настроек приложения

Файл настроек приложения должен быть типовым средством фреймворка технологического стека. В файле должны содержаться следующие ключи:

Контактный телефон (contactPhone).

* E-mail (contactEmail).
* Посылать уведомления (sendNotification).
* Уведомление до начала занятий (часов) (hourNotification).
* SMS-шлюз (smsGate).
* Роль инструктора (roleInstructor). Включение и отключение (по умолчанию) роли инструктора.
* Роль менеджера (roleManager). Включение (по умолчанию) и отключение роли менеджера.
* Учет калорий (trackCalories). Включение (по умолчанию) и отключение функционала учета калорий клиентов.
* Текст приглашения для клиента (invitationText).
* Время жизни ссылки-приглашения (часы) (invitationLinkTTL).
* SMTP-хост (smtpHost).
* SMTP-порт (smtpPort).
* SMTP-логин (smtpLogin).
* SMS-шлюз URL (smsGateUrl).

Файл настроек доступен только администратору.

#### Подсистема уведомлений (Notification Service)

Подсистема уведомлений отвечает за своевременную отправку сообщений пользователям через различные каналы связи (Email, SMS, Push) в соответствии с их настройками и бизнес-событиями системы.

**1. Архитектура и общие требования**

* **Централизованный сервис:** Уведомления должны обрабатываться единым сервисом (отдельным Dart-модулем), который получает события от системы и управляет очередью отправки.
* **Шаблоны сообщений:** Тексты уведомлений должны храниться в базе данных в виде шаблонов с возможностями подстановки переменных (например, {{client\_name}}, {{training\_time}}). Это позволит администратору изменять тексты без изменения кода.
* **Очередь отправки (Queue)**: Для надежности отправка должна использовать очередь сообщений, чтобы избежать потери уведомлений при сбоях в работе внешних сервисов (SMTP, SMS-шлюз).
* **Логирование:** Отправка каждого уведомления должна фиксироваться в отдельной таблице.

**2. Типы уведомлений (Бизнес-события)**

Система должна отправлять уведомления при наступлении следующих событий:

1. **Приглашение клиента:**
   * **Событие:** Менеджер нажал "Отправить приглашение" в карточке лида.
   * **Канал:** Email или SMS (в зависимости от данных лида).
   * **Получатель:** Потенциальный клиент (запись в onboarding\_client).
   * **Содержание:** Приветствие, уникальная ссылка для активации, инструкция.
2. **Напоминание о занятии:**
   * **Событие:** За hourNotification часов до начала занятия.
   * **Триггер:** Система проверяет расписание каждые 15 минут.
   * **Канал:** Push-уведомление в приложении (основной), дублирование на Email (опционально, по настройкам пользователя).
   * **Получатель:** Клиент, Тренер, Инструктор (привязанные к занятию).
   * **Содержание:** Время занятия, название плана тренировок, ссылка на карточку занятия.
3. **Изменение в расписании:**
   * **Событие:** Тренер или менеджер изменил время/дату занятия, отменил его.
   * **Канал:** Push-уведомление, Email.
   * **Получатель:** Клиент, Тренер, Инструктор.
   * **Содержание:** Информация об изменении, старое и новое время.
4. **Новое сообщение в чате:**
   * **Событие:** Пользователь отправил сообщение в чат (Peer-to-Peer или Broadcast).
   * **Канал:** Push-уведомление.
   * **Получатель:** Участник(и) чата, которые в данный момент не в приложении.
   * **Содержание:** Имя отправителя и начало текста сообщения.
5. **Системные уведомления для персонала:**
   * **Событие:** Поступление новой заявки с сайта (onboarding\_client).
   * **Канал:** Push-уведомление, Email.
   * **Получатель:** Дежурный менеджер.
   * **Содержание:** "Поступила новая заявка от [Имя]".

**3. Управление подписками (Пользовательские настройки)**

Каждый пользователь (в дашборде "Настройки") должен иметь возможность независимо настраивать получение уведомлений:

* **Общий переключатель:** sendNotification (Вкл/Выкл все уведомления).
* **Время напоминания о занятии:** hourNotification (в часах).
* **Каналы для разных событий:** Отдельные переключатели для получения уведомлений по каналам:
  + Push-уведомления (всегда Вкл, если не выключены на уровне ОС).
  + Email-уведомления (опционально).
  + SMS-уведомления (опционально, может быть платной услугой).

**4. Технологический стек**

- \*\*Очередь задач:\*\* Redis Queue или пакет dart\_redis

- \*\*Шаблоны сообщений:\*\* PostgreSQL + Handlebars-подобный движок

- \*\*Email отправка:\*\* SMTP сервер (Postfix или внешний провайдер)

- \*\*SMS отправка:\*\* REST API российских SMS-шлюзов

- \*\*Push-уведомления:\*\* Firebase Cloud Messaging (FCM) для мобильных приложений

- \*\*Веб-уведомления:\*\* WebSocket + Service Worker

* 1. **Компоненты системы**

notification\_service/

├── lib/

│ ├── notification\_queue.dart # Очередь уведомлений

│ ├── email\_sender.dart # Отправка email

│ ├── sms\_sender.dart # Отправка SMS

│ ├── push\_sender.dart # Push-уведомления

│ ├── template\_engine.dart # Шаблонизатор

│ └── webhook\_sender.dart # Webhook-уведомления

├── config/

│ ├── smtp\_config.dart # Настройки SMTP

│ └── sms\_providers.dart # Конфигурация SMS-шлюзов

└── workers/

├── email\_worker.dart # Воркер для email

└── sms\_worker.dart # Воркер для SMS

#### Подсистема чата (Messaging)

Чат является критически важным инструментом коммуникации между всеми участниками процесса (клиентами, тренерами, инструкторами, менеджерами).

##### 1. Функциональные требования

###### 1.1. Типы чатов

Система должна поддерживать следующие типы бесед:

1. **Личные сообщения (Peer-to-Peer):**
   * Между любыми двумя пользователями системы, у которых есть права на общение (например, клиент может писать своему тренеру, но не может писать другому клиенту, если это не разрешено настройками).
2. **Групповые чаты (Broadcast / Group):**
   * **Тренер → Его клиенты:** Создается автоматически для каждого тренера с его клиентами.
   * **Инструктор → Его клиенты:** Создается автоматически для каждого инструктора с его клиентами.
   * **Менеджер → Группа сотрудников/клиентов:** Для объявлений.
   * Возможность создания групповых чатов вручную (например, менеджером) с выбором участников.
3. **Системные уведомления:**
   * Уведомления от системы (напоминания, изменения расписания)
   * Broadcast сообщения от администрации

###### 1.2. Базовый функционал

* **Отправка сообщений:**
  + Текстовые сообщения.
  + Поддержка вложений.
* **История сообщений:** Полная история переписки должна храниться и подгружаться при открытии чата (с пагинацией).
* **Статусы сообщений:**
  + sent (отправлено), delivered (доставлено получателю), read (прочитано).
  + В групповых чатах статус read должен отображаться в виде списка прочитавших (или количества).
* **Индикаторы активности:**
  + Индикатор "онлайн" (зеленая точка) рядом с аватаром пользователя.
  + Индикатор "печатает..." (is typing...) в реальном времени.
* **Уведомления:**
  + Push-уведомления на устройство при новом сообщении, если пользователь не в активном чате.
  + Звуковое оповещение (опционально, настраивается пользователем).
  + Бейджи (цифры) на иконке чата в интерфейсе, указывающие на количество непрочитанных сообщений.

###### 1.3. Управление чатами

* **Список чатов:** Интерфейс должен отображать список всех чатов пользователя, отсортированный по времени последнего сообщения.
  + Для каждого чата в списке отображается: аватар/название, последнее сообщение (или вложение), время и бейдж непрочитанных.
* **Поиск по чатам и сообщениям:** Возможность быстрого поиска по названиям чатов и тексту сообщений.
* **Архивация чатов:** Возможность скрыть чат из основного списка без удаления истории.

###### 1.4 Автоматическое создание чатов

* При назначении клиента тренеру → создается личный чат
* При создании группового занятия → создается групповой чат участников
* При найме нового сотрудника → добавляется в общие чаты филиала

##### 2. Архитектура и техническая реализация

###### 2.1. Технологический стек

* **Backend:** Dart сервер на Shelf/Dart Frog
* **База данных:** PostgreSQL 15+
* **Real-time:** WebSocket + Redis Pub/Sub
* **Хранение файлов:** Локальная файловая система
* **Клиент:** Flutter WebSocket client

###### 2.2. Компоненты системы

// Структура серверной части чата

chat\_server/

├── lib/

│ ├── websocket\_server.dart # WebSocket обработчик

│ ├── chat\_service.dart # Бизнес-логика чата

│ ├── message\_repository.dart # Работа с БД

│ └── redis\_service.dart # Pub/Sub для реального времени

├── config/

│ └── database.dart # Конфигурация БД

└── bin/

└── server.dart # Точка входа

###### 2.2. Таблицы чата

-- Таблица чатов

-- Таблица участников чата

-- Таблица сообщений

-- Таблица статусов сообщений (для отслеживания прочтения)

###### 2.3. Алгоритмы и процессы

* **Создание личного чата:** Происходит автоматически при первой попытке отправить сообщение пользователю, если чата еще не существует.
* **Доставка и прочтение:**
  1. Сообщение сохраняется в БД, ему присваивается статус sent.
  2. Через Realtime канал оно мгновенно доставляется всем онлайн-участникам чата. В БД для них проставляется статус delivered.
  3. Когда пользователь открывает чат, клиентское приложение отправляет команду на сервер о прочтении всех сообщений в этом чате. В БД проставляется статус read.
* **Индикатор "печатает...":** При начале ввода текста в поле сообщения клиент отправляет через Realtime событие typing\_start в канал чата. При остановке или отправке сообщения — событие typing\_stop.

##### 3. Требования к интерфейсу (UI/UX)

* **Экран списка чатов:** Должен быть доступен с главного навигационного меню каждого дашборда.
* **Экран чата:**
  + Шапка с названием чата и списком участников (для групповых).
  + Область сообщений с группировкой по датам.
  + Панель ввода сообщения с кнопкой отправки.
  + *На втором этапе: кнопка прикрепления файла.*
* **Контекстные меню:** Должен быть реализован функционал "долгого нажатия" на сообщение для его копирования или пересылки (для администраторов/менеджеров).

##### 4. Безопасность и контроль доступа

* **Row Level Security (RLS):** Должны быть настроены строгие политики RLS на все таблицы чатов. Пользователь может видеть и писать только в те чаты, участником которых он является.
* **Валидация:** Проверка прав пользователя на отправку сообщения в конкретный чат должна выполняться на стороне сервера перед сохранением сообщения.
* **Модерация:** *На втором этапе рассмотреть возможность удаления сообщений для ролей Менеджер/Администратор.*

#### Подсистема отчетности (Reporting & Analytics)

Подсистема отчетности предоставляет руководству и менеджерам инструмент для анализа эффективности фитнес-центра, сотрудников и бизнес-процессов на основе накопленных данных.

##### 1. Общие принципы

* **Доступ на основе ролей:** Отчеты доступны только пользователям с ролями **Администратор** и **Менеджер**. Уровень детализации данных может различаться (например, менеджер видит данные только по своим клиентам/тренерам, а администратор — по всему центру).
* **Интерактивность:** Отчеты должны быть интерактивными: с возможностью фильтрации, сортировки и детализации (drill-down). Например, клик на цифру "Количество занятий" открывает список этих занятий.
* **Визуализация:** Данные должны представляться в виде сводных таблиц, графиков (столбчатые, линейные, круговые) и дашбордов с KPI.
* **Экспорт:** Все отчеты должны поддерживать экспорт данных в форматы **PDF** (для печати и презентаций) и **XLSX/CSV** (для дальнейшего анализа в Excel).

##### 2. Виды отчетов

###### 2.1. Отчетность по клиентам

* **2.1.1. Отчет «Динамика клиентской базы»**
  + **Цель:** Анализ притока и оттока клиентов.
  + **Параметры:** Период (месяц, квартал, год).
  + **Метрики:**
    - Новые клиенты (зарегистрировались за период).
    - Активные клиенты (посетили хотя бы 1 занятие за период).
    - Ушедшие клиенты (не были на занятиях последние N дней).
    - Общее количество клиентов на конец периода.
    - **График:** Динамика изменения этих показателей over time.
* **2.1.2. Отчет «Посещаемость и прогресс клиентов»**
  + **Цель:** Оценка вовлеченности и эффективности тренировок.
  + **Параметры:** Период, конкретный клиент (опционально), тренер (опционально).
  + **Метрики:**
    - Общее количество проведенных занятий.
    - Процент посещаемости от запланированных занятий.
    - Среднее количество занятий в неделю на одного клиента.
    - Динамика изменения веса, обхватов тела.
    - Динамика потребленных и затраченных калорий (дефицит/профицит).
    - Достижение целей (количество клиентов, достигших целевого веса за период).

###### 2.2. Отчетность по сотрудникам (тренеры, инструкторы)

* **2.2.1. Отчет «Загрузка и эффективность тренеров»**
  + **Цель:** Оценка продуктивности и загруженности тренерского состава.
  + **Параметры:** Период, конкретный тренер (опционально).
  + **Метрики:**
    - Общее количество проведенных занятий.
    - Общее количество рабочих часов (по табелю).
    - Количество прикрепленных клиентов (на начало и конец периода).
    - Средний доход с клиента (если интегрировано с финансовой системой. Это вынесено за рамки данной системы и будет интегрироваться via API).
    - Рейтинг тренера на основе отзывов клиентов (если функционал реализован).
    - **График:** Сравнительная диаграмма загрузки тренеров.
* **2.2.2. Отчет «Табель учета рабочего времени» (Расширенная версия)**
  + **Цель:** Финальное подведение итогов для payroll (расчета заработной платы).
  + **Параметры:** Расчетный период (например, с 1 по последнее число месяца).
  + **Метрики:** По каждому сотруднику:
    - Отработанные часы (суммировано из занятий).
    - Количество проведенных индивидуальных/групповых занятий.
    - Простои/опоздания (если ведется учет).
    - **Экспорт:** Обязательный вывод в XLSX для бухгалтерии.

###### 2.3. Отчетность по занятиям и расписанию

* **2.3.1. Отчет «Статистика занятий»**
  + **Цель:** Анализ популярности направлений и загрузки зала.
  + **Параметры:** Период, тип занятия (индивидуальное/групповое), зал (если несколько).
  + **Метрики:**
    - Распределение занятий по типам (индив./групп.).
    - Распределение занятий по планам тренировок (похудение, масса и т.д.).
    - Процент отмененных/перенесенных занятий.
    - Среднее количество участников в групповых занятиях.
    - **График:** Загрузка зала по времени суток и дням недели.

###### 2.4. Финансовая отчетность по центру (если интегрировано с кассой/оплатами)

* **2.4.1. Отчет «Выручка и платежи»**
  + **Цель:** Контроль финансовых потоков.
  + **Параметры:** Период.
  + **Метрики:**
    - Общая выручка за период.
    - Выручка по типам услуг (абонементы, разовые занятия, персональные тренировки).
    - Количество оформленных абонементов.
    - Динамика выручки (сравнение с предыдущим периодом).
    - **График:** Круговая диаграмма структуры выручки.

##### 3. Техническая реализация

###### 3.1. Архитектура

* **Источник данных:** Все отчеты строятся на основе основной базы данных PostgreSQL (таблицы users, lessons, track\_calories, anthropometry\_start/finish и др.).
* **Генерация отчетов:** Для сложных отчетов с агрегацией данных рекомендуется использовать **представления (Views)** или **материализованные представления (Materialized Views)** в PostgreSQL для оптимизации производительности.
* **API:** Бэкенд должен предоставлять API-эндпоинты для получения данных отчетов в формате JSON, которые затем визуализируются на фронтенде.
  + Пример эндпоинта: GET /api/reports/client\_dynamics?start\_date=2023-01-01&end\_date=2023-12-31

###### 3.2. Интерфейс генератора отчетов

* **Дашборд «Аналитика»** в интерфейсе Администратора/Менеджера.
* **Левая панель:** Древовидный список всех доступных отчетов.
* **Центральная часть:**
  1. **Панель параметров:** Выбор периода, фильтров (тренер, клиент, тип занятия).
  2. **Область визуализации:** Здесь отображаются графики и таблицы.
  3. **Панель инструментов:** Кнопки "Обновить", "Экспорт в PDF", "Экспорт в Excel".

##### 4. Пример технического задания для одного отчета

**Отчет: «Динамика клиентской базы за период»**

* **Эндпоинт API:** GET /api/reports/client\_dynamics
* **Параметры запроса (Query Parameters):**
  + start\_date (обязательный)
  + end\_date (обязательный)
* **Ответ (Response Body):**

json

{

"period": {"start": "2023-01-01", "end": "2023-12-31"},

"metrics": {

"new\_clients": 150,

"active\_clients": 400,

"churned\_clients": 45,

"total\_clients\_end": 1200

},

"chart\_data": [

{"month": "2023-01", "new": 10, "active": 350, "churned": 5, "total": 1000},

{"month": "2023-02", "new": 15, "active": 370, "churned": 8, "total": 1007},

...

]

}

* **Логика расчета на бэкенде:**
  + new\_clients: COUNT пользователей с ролью 'client', у которых created\_at входит в период.
  + active\_clients: COUNT уникальных client\_id из таблицы lessons, где start\_fact\_at входит в период и complete = 'completed'.
  + churned\_clients: COUNT клиентов, у которых последнее занятие (MAX(start\_fact\_at)) было раньше, чем end\_date - 30 days (или другой порог "ухода").
  + total\_clients\_end: COUNT пользователей с ролью 'client' на дату end\_date.

#### Подсистема управления пользователями (User Management)

Подсистема управления пользователями является фундаментом всей системы, обеспечивая безопасную аутентификацию, гибкую авторизацию на основе ролей и централизованное управление учетными записями.

##### 1. Модель данных и ролевая система

###### 1.1. Сущность «Пользователь» (User)

Универсальная сущность, хранящая общие данные для всех участников системы.

* **Базовые поля:**
  + id (UUID, Primary Key)
  + login (String, Unique) // Логин для входа (может совпадать с email/телефоном)
  + password\_hash (String) // Хеш пароля (алгоритм bcrypt)
  + email (String, Unique, Nullable)
  + phone (String, Unique, Nullable)
  + last\_name (String)
  + first\_name (String)
  + middle\_name (String, Nullable)
  + gender (SMALLINT) //значение перечисления в коде
  + age (Integer)
  + photo\_url (String, Nullable) // Ссылка на аватар в хранилище
  + created\_at (Timestamp)
  + updated\_at (Timestamp)
  + archived\_at (Timestamp)
  + archived\_by (BIGINT FK → users(id))
* **Настройки уведомлений (хранятся в таблице user\_settings):**
  + send\_notifications (Boolean, default: true) // Разрешить уведомления
  + hour\_notification (Integer) // За сколько часов уведомлять о занятии
  + notification\_email (Boolean) // Получать на email
  + notification\_push (Boolean) // Получать push-уведомления

###### 1.2. Ролевая модель (Role-Based Access Control - RBAC)

Гибкая система, позволяющая назначать пользователю несколько ролей.

* **Таблица roles:**
  + id (Primary Key)
  + name (String, Unique: 'client', 'instructor', 'trainer', 'manager', 'admin') // Системное имя
  + title (String) // Человекочитаемое название ('Клиент', 'Тренер')
  + icon (String) // Иконка роли для UI
* **Таблица user\_roles (связь многие-ко-многим):**
  + user\_id (Foreign Key -> users)
  + role\_id (Foreign Key -> roles)
  + assigned\_at (Timestamp)
* **Иерархия и комбинации ролей:**
  + Пользователь может иметь несколько ролей.
  + **Пример:** В небольшом клубе тренер может иметь роли trainer, manager и admin.
  + Наличие роли определяется бизнес-логикой, а не техническими ограничениями.

###### 1.3. Профили конкретных ролей (Profile Tables)

Дополнительные данные, специфичные для роли, хранятся в отдельных таблицах с связью One-to-One к основной таблице users.

* **Таблица client\_profiles (для клиентов):**
  + user\_id (Primary Key, Foreign Key -> users)
  + goal\_training\_id (Foreign Key -> goals\_training)
  + level\_training\_id (Foreign Key -> levels\_training)
  + track\_calories (Boolean, default: true) // Включить учет калорий
  + coeff\_activity (Double, default: 1.2) // Коэффициент активности
* **Таблица instructor\_profiles (для инструкторов):**
  + user\_id (Primary Key, Foreign Key -> users)
  + specialization (String, Nullable) // Специализация
  + work\_experience (Integer, Nullable) // Стаж в годах
  + is\_duty (Boolean, default: false) // Дежурный (логика на основе employees\_duty)
  + can\_replace\_trainer (Boolean, default: false) // Может замещать тренера
  + can\_create\_plan (Boolean, default: false) // Может создавать планы тренировок
* **Таблица trainer\_profiles (для тренеров):**
  + user\_id (Primary Key, Foreign Key -> users)
  + specialization (String, Nullable) // Специализация
  + work\_experience (Integer, Nullable) // Стаж в годах
* **Таблица manager\_profiles (для менеджеров):**
  + user\_id (Primary Key, Foreign Key -> users)
  + specialization (String, Nullable) // Специализация
  + work\_experience (Integer, Nullable) // Стаж в годах
  + is\_duty (Boolean, default: false) // Дежурный (логика на основе employees\_duty)

##### 2. Функциональные требования

###### 2.1. Жизненный цикл учетной записи

* **Создание пользователя:**
  + **Сценарий А (Администратором):** Через интерфейс "Создание пользователя" с ручным назначением ролей.
  + **Сценарий Б (Самостоятельная регистрация):** Через процесс онбординга (пригласительная ссылка).
* **Активация/Деактивация:** Возможность временно отключить учетную запись без удаления данных (is\_active = false).
* **Удаление:** *Строго логическое удаление.* Запись помечается как удаленная (is\_deleted = true), данные сохраняются для аудита.

###### 2.2. Аутентификация и безопасность

* **Вход в систему:** По паре login (email/телефон) и password.
* **Политика паролей:**
  + Минимальная длина: 8 символов.
  + Обязательные категории: заглавные и строчные буквы, цифры, специальный символ.
  + Запрет на использование простых паролей (проверка against словаря).
* **Защита от брутфорса:** Блокировка учетной записи после 5 неудачных попыток входа за 15 минут.
* **Сброс пароля:** Стандартный flow через email с одноразовой ссылкой.
* **Смена пароля:** Обязательна при первом входе, если пароль был установлен администратором.

###### 2.3. Управление пользователями (CRUD)

* **Просмотр списка пользователей:**
  + **Интерфейс:** Единый дашборд с возможностью фильтрации по ролям, активности, ФИО.
  + **Отображение:** Аватар, ФИО, основные роли, контакты, статус активности.
* **Создание/Редактирование пользователя:**
  + **Универсальная форма:** Основные поля (ФИО, контакты, логин).
  + **Динамические блоки:** В зависимости от выбранных ролей появляются дополнительные поля (например, блок "Данные клиента" при назначении роли client).
  + **Назначение ролей:** Выбор из списка с чекбоксами.
* **Управление доступом:** Только пользователи с ролями admin и manager имеют полный доступ к CRUD. Тренер может просматривать и редактировать только своих клиентов.

##### 3. Техническая реализация

###### 3.1. API Endpoints (Примеры)

* POST /api/auth/login - Вход в систему.
* POST /api/auth/logout - Выход.
* POST /api/auth/reset-password - Запрос на сброс пароля.
* GET /api/users - Получить список пользователей (с фильтрами).
* POST /api/users - Создать нового пользователя.
* PUT /api/users/{id} - Обновить данные пользователя.
* PUT /api/users/{id}/roles - Обновить роли пользователя.

###### 3.2. Безопасность (Row Level Security - RLS)

* **Для всех таблиц (users, client\_profiles, etc.) должны быть настроены политики RLS.**
* **Пример политики для users:**
  + Администратор (role\_admin) видит всех пользователей.
  + Менеджер (role\_manager) видит пользователей, которые являются его клиентами/тренерами (через таблицы связей manager\_client, manager\_trainer).
  + Тренер (role\_trainer) видит только себя и своих клиентов/инструкторов.
  + Клиент (role\_client) видит только свою учетную запись.

###### 3.3. Процессы и бизнес-логика

* **Создание клиента из лида:** При нажатии "Создать клиента" в карточке лида система:
  1. Создает запись в users на основе данных из onboarding\_client.
  2. Назначает пользователю роль client.
  3. Создает запись в client\_profiles.
  4. Устанавливает связь trainer\_client с тренером-менеджером.
* **Проверка уникальности:** При создании/редактировании обязательна проверка на уникальность login, email, phone.

##### Функциональный модуль "Администратор"

###### Дашборд "Администратор"

Обозначение: AdminDashboard.

Заголовок: "Администратор".

Меню дашборда:

* Подменю каталогов. Доступны для создания/редактирования записей всех каталогов.
* Параметры -> Настройки. Открывается popup-форма "Настройки".

Дашборд включает ListView пользователей с панелью инструментов. На панели инструментов находятся кнопки: "Создать" (активная), "Изменить", " Архивировать", "Сброс пароля" (активные при выборе строки в списке). Так же имеется поле поиска по ФИО/телефону и кнопкой сброса содержимого поля. На каждое действие определяется своя команда.

Отдельно размещен виджет SegmentedButton со значениями: Все | Администраторы | Менеджеры | Тренеры | Инструкторы | Клиенты. При клике вызывается соответствующая команда фильтрации списка.

1. **Popup-окно "Создание пользователя"**

Параметр: строка (userRole).

Заголовок: "Создание: " + userRole.

Поля:

* id (string). //Идентификатор.
* login (string). //Логин.
* password (string). //Пароль.
* lastName (string). //Фамилия.
* firstName (string). //Имя.
* middleName (string). //Отчество, необязательное.
* fullName (string). //Полное ФИО.
* shortName (string). //Краткое ФИО.
* phone (string). //Телефон.
* email (string). //E-mail, необязательное.
* photo (URL). //Фото, необязательное.
* sendNotification (bool). //Посылать уведомления. Значение берется из файла настроек.
* hourNotification (duration). //Время уведомления до занятий (в часах). Значение берется из файла настроек.
* List<Role> roles. //Список ролей. По умолчанию – userRole.

В случае создания клиента, отображается дополнительные поля:

* gender (string). //Пол.
* age (int). //Возраст.
* trackCalories (bool) //Учет калорий. Значение берется из файла настроек.
* double coeffActivity = 1.2; //Минимальный коэффициент активности.

Поле "Пароль" имеет отображаемый формат в виде звездочек.

Поля fullName и shortName вычисляются автоматически после ввода lastName, firstName, middleName.

Для выбора пола сформировать выпадающий список со значениями "мужской", "женский".

Кнопки: "Генерировать пароль", "Создать", "Отмена".

По кнопке "Генерировать пароль", запускается команда генерации пароля с солью и значение в поле Пароль.

По кнопке "Создать", запускается команда создания пользователя с параметром userRole.

По кнопке "Отмена" закрывает окно.

**4.1.2 Окно "Сброс пароля"**

Параметр: строка (userLogin).

Лэйбл: "При сбросе пароля будет установлен новый пароль, который нужно будет сообщить пользователю. Продолжить?"

Поле: "Пароль".

Кнопки: "Сбросить пароль" и "Отменить".

При открытии формы запускается Команда "Генерация пароля" и возвращаемое значение подставляется в поле "Пароль".

При нажатии на кнопку "Сбросить пароль" запускается Команда "Сохранение пароля " с параметрами: userLogin и паролем.

**4.1.3 Окно "Дежурства сотрудников"**

**Фрейм "Дежурный менеджер"**

Содержит элементы:

* ФИО дежурного менеджера; //Статус "дежурный" определяется по последней незавершенной записи в employees\_duty (где duty\_finish\_at IS NULL)'
* Дежурство с: ' + время последней записи duty\_start\_at из таблицы employees\_duty по id из таблицы users;
* ListView пользователей, отфильтрованных по роли Менеджер.
* Кнопка с заголовком "Назначить", если ФИО пусто, или с заголовком "Сменить", если ФИО не пусто.

**Логика:** После выбора строки из ListView, и нажатия кнопки, запускается команда назначения дежурного сотрудника с параметрами: типа роли и id сотрудника. У нее есть 2 варианта:

1. id == null. В этом случае выдается предупреждающее сообщение "Назначить имярек дежурным менеджером?" В случае "Да" в таблице employees\_duty создается новая запись с текущем временем в поле duty\_start\_at. На выходе команды заполняется поле ФИО и 'Дежурство с: ' + время duty\_start\_at;
2. id != null. В этом случае выдается предупреждающее сообщение "Сменить дежурного менеджера имярек1 на имярек2?" В случае "Да" в таблице employees\_duty по id менеджера1 устанавливается текущем временем поле duty\_finish\_at. Далее создается новая запись с текущем временем в поле duty\_start\_at. На выходе команды заполняется поле ФИО и 'Дежурство с: ' + время dutyManagerStartAt;

**4.1.4 Окно "Настройки"**

Имеет поля из файла настроек.

Кнопка: "Сохранить". Становится доступной при изменении любого поля окна.

**4.1.5 Окно "Расписание работы центра"**

Содержит редактируемую ListView из таблицы work\_schedules.

**4.1.6 Окно "Расписание занятий"**

Содержит все виды расписаний занятий, с выборкой по тренеру, инструктору, клиенту.

Содержит функционал составления общего расписания, обновления по клиенту.

**4.1.7 Окно "Табели"**

Окно содержит дашборд "Табель" в групповом режиме.

##### Функциональный модуль "Клиент"

###### Таблицы индивидуальных назначений

* Таблица: client\_training\_plans
  + id (PK)
  + client\_id (FK -> clients)
  + training\_plan\_template\_id (FK -> training\_plan\_templates) // Исходный шаблон
  + assigned\_by (FK -> users) // Кто назначил (тренер)
  + assigned\_at (TIMESTAMP)
  + is\_active (BOOLEAN) // Активный план у клиента
  + goal (VARCHAR) // Цель может быть переопределена для этого клиента
  + notes (TEXT) // Персональные заметки тренера
* Таблица: client\_set\_exercises
  + id (PK)
  + client\_training\_plan\_id (FK -> client\_training\_plans) // Часть какого ИНДИВИДУАЛЬНОГО плана
  + set\_exercise\_template\_id (FK -> set\_exercise\_templates) // Основано на каком шаблоне
  + order (INT) // Порядок в плане для этого клиента
  + is\_active (BOOLEAN)
  + repeats (INT) // Индивидуальное кол-во повторений всего сета
  + rest\_after\_set (REAL) // Индивидуальное время отдыха после сета
* Таблица: client\_exercises
  + id (PK)
  + client\_set\_exercise\_id (FK -> client\_set\_exercises) // Часть какого ИНДИВИДУАЛЬНОГО сета
  + exercise\_template\_id (FK -> exercise\_templates) // Основано на каком шаблоне
  + order (INT) // Порядок в сете для этого клиента
  + custom\_repeat\_qty (INT) // Индивидуальное кол-во повторов
  + custom\_duration\_exec (REAL) // Индивидуальная длительность
  + custom\_duration\_rest (REAL) // Индивидуальный отдых
  + custom\_notes (TEXT) // Персональные указания (например, "ставить ноги шире")

Таблица "lessons"

* schedule\_id (BIGINT); //определяет запланированный слот,
* client\_training\_plan\_id (BIGINT); //конкретный план, который был выполнен в этом слоте
* set\_exercises\_id (BIGINT);
* client\_id (BIGINT);
* instructor\_id (BIGINT);
* trainer\_id (BIGINT);
* start\_plan\_at (TIMESTAMP);
* start\_fact\_at (TIMESTAMP);
* finish\_plan\_at (TIMESTAMP);
* finish\_fact\_at (TIMESTAMP);
* complete (INT);
* note (VARCHAR(100));

Таблица lessons заполняется на основе расписания.

Таблица "track\_сalories"

* client\_id (BIGINT);
* lesson\_id (BIGINT);
* weight (double); //вес
* calories\_in (REAL); //входящие калории
* calories\_out (REAL); //исходящие калории

Таблица хранения формулы расчета BMR "bmr\_formulas"

* name (VARCHAR(100));
* formula (TEXT), -- например: "66 + (13.7 \* weight) + (5 \* height) - (6.8 \* age)"
* for\_men (BOOLEAN),
* for\_women (BOOLEAN),
* is\_active (BOOLEAN DEFAULT true)

###### Таблицы антропометрии

Таблица "anthropometry\_fix" (фиксированные значения, вводятся 1 раз).

* client\_id (BIGINT);
* date\_time (TIMESTAMP);
* height (int); //рост в см
* wrist\_circ (Обхват запястья, см);
* ankle\_circ (Обхват лодыжки, см);

Таблица "anthropometry\_start" (антропология начало).

* client\_id (BIGINT);
* date\_time (TIMESTAMP);
* photo (URL); //фото
* weight (double); //вес в кг
* shoulders\_circ (int); //обхват в плечах
* breast\_circ (int); //обхват в груди
* waist\_circ (int); //обхват в талии
* hips\_circ (int); //обхват бедер
* bmr (int); //коэффициент BMR

Таблица "anthropometry\_finish" " (антропология окончание). По структуре повторяет таблицу "anthropometry\_start".

Таблицы anthropometry\_ обновлять могут только сотрудники (не роль Клиент).

###### Таблицы биоимпеданса

Таблица "bioimpedance\_start"

* client\_id (BIGINT);
* date\_time (TIMESTAMP);
* fat\_percentage (Процент жира);
* muscle\_mass (Мышечная масса, кг);
* water\_percentage (Процент воды);
* visceral\_fat (Висцеральный жир, уровень);
* bmc (Минеральная масса костей);
* bmi (Индекс массы тела, часто считает сам аппарат);
* metabolism (Скорость основного обмена, ккал);

Таблица "bioimpedance\_finish". По структуре повторяет таблицу "bioimpedance\_start".

Таблицы bioimpedance\_ обновлять могут только сотрудники (не роль Клиент).

###### Таблица предпочтений

Таблица "client\_schedule\_preferences" (предпочтения клиента по расписанию).

* id\_client (FK -> Client). Ссылка на клиента.
* day\_of\_week (Integer, NOT NULL). День недели: 1-7.
* preferred\_start\_time (Time, NOT NULL). Желаемое время начала занятия.
* preferred\_end\_time (Time, NOT NULL). Желаемое время окончания занятия.

###### Списочная форма

ListView клиентов должна отображаться в дашбордах инструкторов, тренеров, менеджеров в окне "Клиенты".

Отображаемые поля:

* Фото клиента. При выборе поля открывается дашборд клиента.
* Фамилия, Имя, Отчество. При выборе поля открывается дашборд клиента.
* Занятие ближайшее. При выборе поля открывается дашборд занятия
* Инфо.

Поле Инфо представляет собой строку информации о ближайшем занятии клиента в зависимости от значения hourNotification клиента:

* Текущее время меньше разницы времени начала ближайшего занятия и hourNotification:

отображается строка: "До занятия " + (разница времени начала ближайшего занятия и текущего) + " час."

* Текущее время в пределах разницы времени начала ближайшего занятия и hourNotification:

отображается строка на желтом фоне: "До занятия " + (разница времени начала ближайшего занятия и текущего) + " час."

* Текущее время больше времени начала ближайшего занятия и меньше времени окончания занятия:

отображается строка на зеленом фоне: "Идет занятие".

Функционал поля Инфо является задачей бэкенд-API с периодом 1 минута.

###### Дашборд "Клиент"

Дашборд "Клиент" должен быть доступен как клиенту, так и всем остальным ролям при открытии дашборда из списка клиентов.

Обозначение: ClientDashboard.

Дашборд должен включать следующие виджеты:

1. **Виджет "Следующая тренировка":**
   * Отображает информацию о ближайшем предстоящем занятии.
   * Включает динамический обратный отсчет до начала занятия.
   * Содержит кнопку-ссылку для перехода к деталям занятия.
2. **Виджет "Прогресс тренировок":**
   * Отображает общую статистику: количество завершенных занятий за период, общее количество сожженных калорий, процент посещаемости.
   * Визуализирует данные с помощью индикаторов выполнения (progress bar) и компактных графиков активности.
3. **Виджет "Прогресс по цели":**
   * Отображает текущую цель клиента (напр., целевой вес).
   * Показывает прогресс в достижении цели в виде численных индикаторов и тренда на мини-графике.
   * Отображает средний недельный дефицит или профицит калорий.
4. **Виджет "Достижения":**
   * В геймифицированном виде отображает последние полученные клиентом достижения.
   * Представляет собой сетку иконок с кратким описанием.
5. **Быстрое меню:** Доступ к основным разделам ("Занятия", "Учет калорий") должен быть реализован в виде панели быстрого доступа или виджетов-карточек на самом дашборде.

*Примечание: Виджеты должны быть адаптивными и перестраиваться в зависимости от размера экрана устройства. Порядок и наличие виджетов можно настроить.*

**Меню дашборда:**

* Мой тренер.
* Мой инструктор (при включенном roleInstructor из файла настройки).
* Мой менеджер (при включенном roleManager из файла настройки).
* Антропометрия.
* Занятия.
* Учет калорий (при включенном trackCalories из таблицы client).
* Прогресс.

**5.6.1 Окно "Мой тренер"**

Содержит фото тренера, полное ФИО, телефон, фрейм чата.

**5.6.2 Окно "Мой инструктор"**

Содержит фото инструктора, полное ФИО, телефон, фрейм чата.

**5.6.3 Окно "Мой менеджер"**

Содержит фото менеджера, полное ФИО, телефон, фрейм чата.

**5.6.4 Окно "Антропометрия"**

Состоит из:

* фрейма вверху по центру с ListView таблицы anthropometry\_fix;
* 2-х одинаковых фреймов "Начало" и "Окончание", расположенных горизонтально, с ListView таблиц anthropometry\_start и anthropometry\_finish.

**Функция "Сравнить":**

* При нажатии кнопки "Сравнить" система открывает полноэкранный режим для визуального сравнения фотографий "до" и "после".
* Реализовать режим **"Слайдер"** (split-view), позволяющий пользователю интерактивно перекрывать одно изображение другим для наглядной оценки изменений.
* *На втором этапе разработки рассмотреть возможность добавления функционала* ***ручной разметки*** *ключевых точек на фотографиях для автоматического расчета и сравнения объемов тела.*
* *Технологии автоматического замера объемов с помощью Computer Vision (AI) являются опциональными и требуют отдельного технико-экономического обоснования.*

**5.6.5 Окно "Биоимпеданс"**

Состоит из 2-х одинаковых фреймов "Начало" и "Окончание", расположенных горизонтально, с ListView таблиц bioimpedance\_start и bioimpedance\_finish.

**5.6.6 Окно "Занятия"**

Окно содержит ListView "Lessons" c расписанием занятий.

Поля:

* № занятия.
* Наименование набора упражнений.
* Начало занятия план.
* Конец занятия план.
* Начало занятия факт.
* Конец занятия факт.
* Длительность.
* Инструктор.
* Тренер.
* Выполнение.
* Примечание.

Логика отображения:

* Если у клиента есть активные занятия в расписании: отображается таблица с расписанием.
* Если активных занятий нет: таблица скрывается. Вместо нее отображается виджет с текстом: "Вам еще не назначены занятия. Заполните, пожалуйста, предпочтения по времени." и кнопка "Заполнить предпочтения".

Логика кнопки "Заполнить предпочтения":

* При нажатии открывается окно "Предпочтения по расписанию" ClientPreferenceSchedule.

После тапа-клика строки ListView "Lessons", открывается карточка "Занятие". Редактирование полей карточки возможно только последней записи.

Окно содержит кнопку "Создать", после чего создается запись в таблице lessons на основе расписания и открывается карточка "Занятие".

**Карточка "Занятие"**

Поля:

* № занятия (недоступно).
* Наименование набора упражнений (недоступно).
* Начало занятия план (недоступно).
* Конец занятия план (недоступно).
* Начало занятия факт.
* Конец занятия факт.
* Длительность (недоступно).
* Инструктор.
* Тренер.
* Выполнение.
* Примечание.

Примечания:

По умолчанию поля Начало занятия факт = Начало занятия план, Конец занятия факт = Конец занятия план, и их можно изменить.

Длительность рассчитывается после обновления поля "Выполнение".

По умолчанию поле Выполнение пустое. Поле Выполнение выбирается из  
enum CompleteLesson {completed, postponed, canceled, frozen} (выполнено, перенесено, отменено, заморожено). В случае выбора из последних двух, открывается inputBox для обязательного ввода примечания. Нельзя сохранить запись с пустым полем Выполнение.

В полях Инструктор и Тренер можно выбрать из выпадающего списка произвольного инструктора и тренера.

**5.6.7 Окно "Предпочтения по расписанию"**

Класс ClientPreferenceSchedule.

* Форма загружает данные из таблицы work\_schedules.
* Для каждого рабочего дня недели (is\_day\_off = false) отображается строка с:
* Наименованием дня недели.
* Диапазоном доступного времени (взят из work\_schedules для этого дня). *Например: "Понедельник (доступно с 09:00 до 20:00)"*. *(минус 1 час на последнее занятие, как вы и предложили: end\_time - 1h)*.
* Полями для ввода или выбором времени (TimePicker) "Желаемое время начала" и "Желаемое время окончания" в пределах доступного диапазона.
* Клиент может заполнить предпочтения для одного или нескольких дней.
* Кнопки "Сохранить" и "Отмена".

**5.6.8 Окно "Учет калорий"**

Окно представляет собой дашборд, включающий ListView "Таблица учета калорий".

Редактирование разрешено только последней записи.

**Поля ListView:**

№ занятия (недоступно).

Наименование набора упражнений (недоступно).

Начало занятия факт (недоступно).

Длительность (недоступно).

Входящий вес (in-line).

Полученные калории (in-line).

Затраченные калории (недоступно).

*Примечания:*

Клиент, после создания занятия, вводит свой вес и полученные калории в последнюю запись в режиме in-line. Полученные калории рассчитываются за время в часах, прошедшее с окончания предыдущего занятия до начала текущего**.** После ввода калорий рассчитывается поле "Затраченные калории" по формуле

**Затраченные калории** = (BMR / 24 \* coeffActivity \* ΔT) + calories\_out

где:

* BMR - Основной обмен веществ (ккал/день).
* 24 - количество часов в сутках.
* coeffActivity - Коэффициент активности.
* ΔT (Delta T) - Время в часах между занятиями, прошедшее **с окончания** предыдущего занятия **до начала текущего.**
* calories\_out - Расход на занятии. Рассчитывается как сумма caloriesOut всех упражнений в плане, скорректированная на длительность и интенсивность.

Если занятие первое, то клиент вводит полученные калории за последние сутки и рассчитывается за последние сутки.

**Пример 1: Тренировки день через день (короткий интервал)**

* **Данные:**
  + BMR = 1600 ккал/день
  + coeffActivity = 1.55
  + Предыдущее занятие закончилось вчера в 19:00.
  + Текущее занятие началось сегодня в 10:00.
  + ΔT = 15 часов (с 19:00 до 10:00 следующего дня).
  + Расход на занятии = 350 ккал.
* **Расчет:**
  1. **Часовой BMR:** 1600 / 24 ≈ 66.67 ккал/час
  2. **Расход за период между тренировками:** 66.67 ккал/час \* 1.55 \* 15 часов ≈ 1550 ккал
  3. **Общий расход:** 1550 ккал + 350 ккал = 1900 ккал

**Пример 2: Тренировки 2 раза в неделю (длинный интервал)**

* **Данные:**
  + ΔT = 3 дня = 72 часа (с понедельника 19:00 до четверга 19:00).
  + Расход на занятии = 400 ккал.
* **Расчет:**
  1. **Расход за период:** 66.67 ккал/час \* 1.55 \* 72 часа ≈ 7440 ккал
  2. **Общий расход:** 7440 ккал + 400 ккал = 7840 ккал

Расчет затраченных калорий на занятии является задачей бэкенд-API.

**5.6.9 Окно "Прогресс"**

Комплекс визуализации прогресса должен включать:

* График зависимости веса и калорий.
* График дефицита/профицита калорий.
* Индикаторы и KPI.
* Выводы системы рекомендаций.
* Интерактивные элементы.
* Адаптивный дизайн.
* Интеграция с планами тренировок.

##### Функциональный модуль "Инструктор"

###### **Списочная форма**

ListView инструкторов должна отображаться в дашбордах тренера, менеджера в окне "Инструкторы".

Отображаемые поля:

* Фото. При выборе поля открывается дашборд инструктора.
* Фамилия, Имя, Отчество. При выборе поля открывается дашборд инструктора.
* Может создавать планы; //can\_create\_plan из таблицы **instructor\_profiles**
* Может замещать тренера; //can\_replace\_trainer из таблицы **instructor\_profiles**

###### **Дашборд "Инструктор"**

Дашборд должен быть доступен как инструктору, так и тренеру, и менеджеру, и администратору.

Обозначение: InstructorDashboard.

Главные поля:

* Фото.
* Фамилия, Имя, Отчество.
* Может создавать планы; //can\_create\_plan из таблицы **instructor\_profiles**
* Может замещать тренера; //can\_replace\_trainer из таблицы **instructor\_profiles**

Меню дашборда:

* Клиенты
* Мой тренер.
* Мой менеджер (при включенном roleManager из файла настройки).
* Расписание.
* Табель.
* Подменю каталогов. Если инструктор имеет право, (can\_create\_plan) может создавать/редактировать только шаблоны упражнений и планов, все остальное – только просмотр.

**6.2.1 Окно "Клиенты"**

Содержит ListView привязанных клиентов, фрейм группового чата. При выборе клиента открывается дашборд клиента.

**6.2.2 Окно "Мой тренер"**

Содержит фото тренера, полное ФИО, телефон, фрейм чата.

**6.2.3 Окно "Мой менеджер"**

Содержит фото менеджера, полное ФИО, телефон, фрейм чата.

**6.2.4 Окно "Расписание"**

Окно содержит персональное расписание инструктора.

**6.2.5 Окно "Табель"**

Окно содержит дашборд "Табель" в одиночном режиме.

##### Функциональный модуль "Тренер"

###### **7.1 Списочная форма**

ListView тренеров должна отображаться в дашборде менеджера, в окне "Тренеры".

###### **7.2 Дашборд "Тренер"**

Дашборд должен быть доступен как тренеру, так и менеджеру, и администратору.

Обозначение: TrainerDashboard.

Меню дашборда:

* Клиенты
* Мой менеджер (при включенном roleManager из файла настройки).
* Расписание.
* Табель.
* Подменю каталогов. Тренер может создавать/редактировать только шаблоны упражнений и планов, все остальное – только просмотр.

**7.2.1 Окно "Клиенты"**

Содержит ListView привязанных клиентов, фрейм группового чата. При выборе клиента открывается дашборд клиента.

**7.2.2 Окно "Инструкторы"**

Содержит ListView привязанных инструкторов, фрейм группового чата. При выборе инструктора открывается дашборд инструктора.

**7.2.3 Окно "Мой менеджер"**

Содержит фото менеджера, полное ФИО, телефон, фрейм чата.

**7.2.4 Окно "Расписание"**

Окно содержит персональное расписание тренера.

**7.2.5 Окно "Табель"**

Окно содержит дашборд "Табель" в одиночном режиме.

##### Функциональный модуль "Менеджер"

###### **8.1 Дашборд "Менеджер"**

Дашборд должен быть доступен как менеджеру, так и администратору.

Обозначение: ManagerDashboard.

Меню дашборда:

* Клиенты
* Расписание.
* Табели.
* Подменю каталогов. Доступны для создания/редактирования записей всех каталогов, кроме "Расписание работы центра" и Дежурства сотрудников, они только для просмотра.

**Окно "Клиенты"**

Содержит ListView привязанных клиентов, фрейм группового чата. При выборе клиента открывается дашборд клиента.

**Окно "Инструкторы"**

Содержит ListView привязанных инструкторов, фрейм группового чата. При выборе инструктора открывается дашборд инструктора.

**Окно "Тренеры"**

Содержит ListView привязанных тренеров, фрейм группового чата. При выборе тренера открывается дашборд тренера.

**Окно "Расписание"**

Содержит все виды расписаний занятий, с выборкой по тренеру, инструктору, клиенту.

Содержит функционал составления общего расписания, обновления по клиенту.

**Окно "Табели"**

Окно содержит дашборд "Табель" в групповом режиме.

#### Архитектура планов тренировок

Система должна реализовывать модель разделения на **Шаблоны** и **Индивидуальные назначения** для поддержки кастомизации тренировок для каждого клиента.

1. **Таблицы шаблонов (Templates).** Являются общей библиотекой сущностей, доступной для создания и редактирования тренерам.
   * training\_plan\_templates
   * set\_exercise\_templates
   * exercise\_templates
2. **Таблицы индивидуальных назначений (Assignments).** Связывают клиента с его персональной версией плана тренировок, созданной на основе шаблонов. Поля в этих таблицах позволяют переопределить любые параметры шаблона (повторы, длительность, порядок, заметки).
   * client\_training\_plans - Связь "клиент - план". Хранит активный план и историю назначений.
   * client\_set\_exercises - Индивидуальные наборы упражнений в рамках плана клиента.
   * client\_exercises - Индивидуальные упражнения в рамках набора клиента.
3. **Связь с занятиями.** Таблица lessons должна быть связана с таблицей client\_training\_plans через foreign key, чтобы однозначно идентифицировать, какая персональная версия плана отрабатывалась на занятии.

#### Расписание занятий

##### Виды и шаблоны расписаний

Система должна поддерживать следующие виды расписаний и их отображение:

1. **Общее расписание зала:** Сводная таблица всех групповых занятий на неделю с привязкой ко времени, залу и тренеру. Должна поддерживать фильтрацию по залу и типу занятия.
2. **Персональное расписание тренера и инструктора:** Детальное расписание конкретного тренера с указанием клиентов, планов тренировок и статусов занятий на каждый день. Должно отображать свободные и занятые слоты.
3. **Индивидуальное расписание клиента:** Упрощенный вид, отображающий только занятия конкретного клиента с указанием статуса (Завершено, Запланировано, Отменено, Заморожено).
4. **При коллизии предпочтений клиентов** (см. класс ClientPreferenceSchedule), отдавать приоритет по дате-времени регистрации.
5. **Поддержка сложных сценариев:**
   * *Повторяющиеся события:* Создание расписания по шаблону (ежедневно, еженедельно) на указанный период.
   * *Исключения:* Возможность отменить одно повторяющееся событие в серии.
   * *Перенос:* Возможность изменить время или дату одного события в серии.
   * *Контроль конфликтов:* Система не должна позволять назначать клиенту или тренеру два занятия на одно и то же время.

##### Процесс составления расписания

1. Выбор плана тренировок.
2. Проверка конфликтов.
3. Генерация расписания на период.
4. UI для составления расписания.
5. Интеграция с каталогами.

##### Процесс работы:

1. Менеджер/Тренер выбирает дату, время, тренера.
2. Система предлагает подходящие планы тренировок based on цели клиентов.
3. Проверяются конфликты в расписании.
4. Рассчитывается длительность и калорийность.
5. Создается расписание с привязкой ко всем каталогам.

##### Таблицы расписания

1. Основная таблица расписания schedules.
2. Таблица связи расписания с клиентами schedule\_clients.
3. Таблица повторяющихся расписаний recurring\_schedules.
4. Таблица исключений расписания schedule\_exceptions.
5. Таблица уведомлений о занятиях schedule\_notifications.
6. Выходная таблица занятий lessons.

#### Табели

Табели составляются на сотрудников, имеющих время проведения занятий – тренерам и инструкторам.

Табели могут отображаться как на одного сотрудника, так и на группы (роли) и имеет 2 режима: одиночный и групповой.

Табель представляет собой дашборд "Табель" с элементами управления отображения и ListView времени занятий.

В дашборде «Табель» фильтр по умолчанию: archived\_at IS NULL.  
Администратор может включить «Показать архивных сотрудников».

* 1. **Элементы управления:**
* **Выбор интервала времени.** По умолчанию отображается начало текущего месяца и текущее число.
* **Комбобокс "Вид сетки".** Выбор разреза времени: день, неделя, месяц. По умолчанию – день. В зависимости от выбора вида сетки, формируются колонки с соответствующими заголовками, уточняющими время.
* **Комбобокс "Сотрудники".** Отображается только в групповом режиме. Осуществляет выбор групп сотрудников: все, тренеры, инструкторы. По умолчанию – все.
* **Экспорт.** Осуществляется выбор экспорта в pdf, xlsx или csv и по кнопке "Экспорт", проводит экспорт в указанный каталог.
  1. **Режимы отображения табеля.**
* **Групповой**. В левой колонке отображаются сотрудники, выбранные по значению комбобокса "Сотрудники", с общем временем по каждому и общем временем в последней строке "Всего". Далее по горизонтали отображаются колонки в зависимости от выбора комбобокса "Вид сетки". После клика по сотруднику, открывается одиночный режим.
* **Одиночный**. В этом режиме колонка сотрудника скрывается, ФИО сотрудника добавляется в шапку, так же добавляется колонка со всеми ФИО прикрепленными клиентами, с общем временем по каждому и общем временем в последней строке "Всего". После клика по клиенту, открывается окно "Занятия" клиента за выбранный интервал времени.

В клетках сетки табеля, по колонкам выбранного разреза, отображаются суммирующее фактическое время, полученное функцией подсчета времени занятий.

#### Подсистема рекомендаций по тренингу

##### Архитектура алгоритма рекомендаций

**Входные данные (от пользователя):**

1. **Константы:** Пол, Возраст, Рост.
2. **Генетические замеры (не меняются):** Обхват запястья (wrist\_circ), Обхват лодыжки (ankle\_circ).
3. **Текущие динамические замеры:** Обхват плеч (shoulders\_circ), груди (breast\_circ), талии (waist\_circ), бедер (hips\_circ).
4. **Данные биоимпеданса (если есть):** Процент жира (fat\_percentage), мышечная масса (muscle\_mass), скорость основного обмена (BMR).

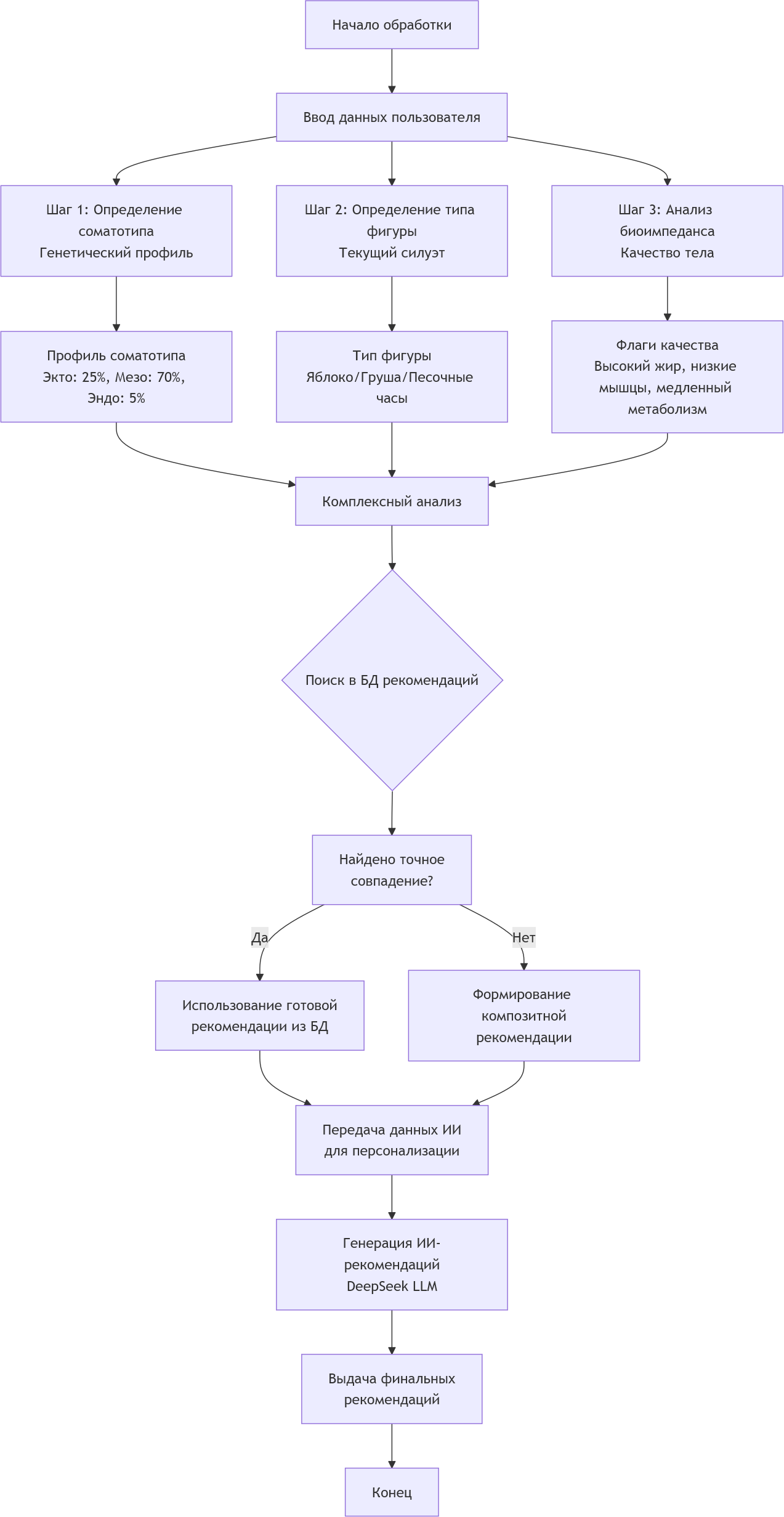


Рисунок алгоритма рекомендаций

**Пояснения к алгоритму**

**Шаг 1: Определение соматотипа (Вероятностный, генетический профиль).**

* **Что анализируется:** Обхват запястья и лодыжки + пол пользователя
* **Как работает:** Система сравнивает замеры с таблицей types\_body\_build
* **Результат:** Вероятностный профиль (например: "Эктоморф: 15%, Мезоморф: 80%, Эндоморф: 5%")
* **Особенность:** Учитывает генетическую предрасположенность, которая не меняется при тренировках
* **Основа:** Данные клиента (пол, wrist\_circ, ankle\_circ) сверяются с правилами в каталоге "Типы телосложения " (таблица types\_body\_build).
* **Логика:**
  1. Для каждого типа телосложения (Эктоморф, Мезоморф, Эндоморф) из каталога, подходящего по полу клиента, система рассчитывает **балл близости**.
  2. **Расчет балла для запястья:**
     + Если wrist\_circ меньше rule\_wrist\_min, балл = 0.
     + Если wrist\_circ между rule\_wrist\_min и rule\_wrist\_max, балл = 100 (полное соответствие).
     + Если wrist\_circ больше rule\_wrist\_max, балл линейно убывает от 100 до 0 на интервале от rule\_wrist\_max до (rule\_wrist\_max + 2 см).
  3. **Аналогично рассчитывается балл для лодыжки** на основе ankle\_circ, rule\_ankle\_min, rule\_ankle\_max.
  4. **Итоговый балл по типу** = среднее арифметическое баллов для запястья и лодыжки. Если данные для лодыжки отсутствуют, используется только балл для запястья.
  5. Система выполняет п.2-4 для всех трех типов.
  6. **Результат:** Формируется профиль вида { "Эктоморф": 25, "Мезоморф": 70, "Эндоморф": 5 }, где значения - процент принадлежности (сумма процентов равна 100 после нормализации).

**Шаг 2: Определение Типа Фигуры (Текущий, динамический силуэт).**

* **Что анализируется:** Соотношение обхватов плеч, талии и бедер
* **Как работает:** Математический расчет пропорций по четким правилам
* **Результат:** Один из типов: "Яблоко", "Груша", "Песочные часы", "Прямоугольник", "Перевернутый треугольник"
* **Особенность:** Показывает текущее состояние тела, которое можно изменить
* **Основа:** Обхват плеч (shoulders\_circ), талии (waist\_circ), бедер (hips\_circ).
* **Логика:** Алгоритм вычисляет соотношения.
  + IF (waist\_circ / hips\_circ >= 0.85 AND waist\_circ > shoulders\_circ AND waist\_circ > hips\_circ) THEN bodytype = 'Яблоко' (Apple)
  + IF (hips\_circ > shoulders\_circ \* 1.05) THEN bodytype = 'Груша' (Pear)
  + IF (ABS(shoulders\_circ - hips\_circ) < 0.05 AND (waist\_circ / shoulders\_circ) < 0.75) THEN bodytype = 'Песочные часы' (Hourglass)
  + IF (ABS(shoulders\_circ - hips\_circ) < 0.05 AND (waist\_circ / shoulders\_circ) >= 0.75) THEN bodytype = 'Прямоугольник' (Rectangle)
  + IF (shoulders\_circ > hips\_circ \* 1.05) THEN bodytype = 'Перевернутый треугольник' (Inverted Triangle)

**Шаг 3: Уточнение через Биоимпеданс (Качество состава тела)**.

* **Что анализируется:** Процент жира, мышечная масса, скорость метаболизма
* **Как работает:** Оценка качества тела относительно здоровых норм
* **Результат:** Флаги качества ("высокий жир", "низкие мышцы", "медленный метаболизм")
* **Особенность:** Дает качественную оценку состава тела
* **Основа:** Процент жира (fat\_percentage), мышечная масса (muscle\_mass), BMR.
* **Логика:** Алгоритм добавляет "качественную" оценку.
  + **Фактор ожирения:** IF (fat\_percentage > max\_healthy\_for\_age\_gender) THEN flag = 'high\_fat'
  + **Фактор худобы:** IF (fat\_percentage < min\_healthy\_for\_age\_gender AND muscle\_mass < average) THEN flag = 'low\_muscle'
  + **Фактор метаболизма:** IF (BMR > calculated\_average\_BMR \* 1.1) THEN metabolism = 'fast' ELSE IF (BMR < calculated\_average\_BMR \* 0.9) THEN metabolism = 'slow'

**Шаг 4: Комплексный анализ и вывод рекомендаций**

**Процесс объединения:**

* Система собирает данные всех трех этапов
* Создает комплексный "портрет" пользователя
* Учитывает как генетику, так и текущее состояние

**Пример портрета:**

* *Генетика:* Преимущественно мезоморф (70%) с небольшой склонностью к эндоморфу (25%)
* *Фигура:* "Яблоко" (скопление жира в области талии)
* *Качество:* Высокий процент жира, нормальные мышцы, медленный метаболизм
* **Основа:** Комбинация всех полученных данных: **профиль соматотипа** + bodytype + flags + metabolism.
* **Логика поиска в БД:**
  1. Система ищет в таблице training\_recommendations записи, соответствующие bodytype, goal\_training\_id, level\_training\_id.
  2. Если найдено несколько записей, приоритет отдается той, которая лучше всего подходит для **преобладающего соматотипа** из профиля клиента.
  3. Если точного совпадения нет, система формирует **композитную рекомендацию**. Например, если профиль клиента {"Мезоморф": 60, "Эндоморф": 40}, итоговая рекомендация может быть: "На 60% используй стратегию для мезоморфа, на 40% - стратегию для эндоморфа, уделяя внимание ...".
* **Учет смешанных типов для ИИ:** Сформированный профиль соматотипа и все остальные данные передаются ИИ как есть. Это позволяет ИИ генерировать truly персонализированные рекомендации, учитывающие смешанную природу телосложения пользователя.

**Таблица training\_recommendations (Рекомендации по тренировкам)**

* body\_type// тип фигуры
* goal\_training\_id; // из таблицы Цели тренировок;
* level\_trainig\_id; //из таблицы Уровень подготовки.
* recommendation\_text\_trainer // Текст рекомендации для тренера.
* recommendation\_text\_client // Текст рекомендации для клиента.

##### Гибридная система рекомендаций с ИИ

**Алгоритмы и таблица:** Формируют **структурированный, проверенный и безопасный каркас** рекомендаций. **Новая вероятностная модель** обеспечивает гораздо более высокую точность для пользователей со смешанным типом телосложения.

**ИИ-модель (LLM DeepSeek):** Выступает как **"креативный ассистент-эксперт"**, который:

* **Персонализирует** стандартные рекомендации под уникальный, в том числе смешанный, кейс на основе переданного профиля.
* **Генерирует** конкретные примеры, учитывающие нюансы смешанного типа (например, "так как у вас есть склонность к эндоморфу, несмотря на мезоморфную основу, рекомендуем...").
* **Объясняет** сложные концепции простым языком.

**Обновленный Пример Промпта (Запроса) для ИИ:**

Ты - опытный персональный тренер и диетолог с 10-летним опытом.

Генерируй персонализированные рекомендации по тренировкам и питанию.

ОГРАНИЧЕНИЯ И ПРАВИЛА:

1. НЕ давай медицинских рекомендаций и диагнозов

2. НЕ предлагай добавки и фармакологию

3. НЕ создавай экстремальные диеты (менее 1200 ккал/день)

4. Призывай консультироваться с врачом при проблемах со здоровьем

5. Используй только научно-доказанные методы

6. Учитывай российские пищевые привычки и доступность продуктов

**ДАННЫЕ ПОЛЬЗОВАТЕЛЯ:**  
- Пол: {пол}  
- Возраст: {возраст}  
- Цель: {цель\_из\_таблицы}  
- Опыт тренировок: {уровень}  
- **Генетический профиль телосложения (соматотип):** {рассчитанный\_профиль\_соматотипа} // Напр.: "Эктоморф: 15%, Мезоморф: 80%, Эндоморф: 5%"  
- Актуальный тип фигуры: {тип\_фигуры}  
- Биоимпеданс: процент жира - {процент\_жира}, мышечная масса - {мышцы}, скорость метаболизма - {BMR}.  
- Основная рекомендация из нашей базы данных: "{базовая\_рекомендация\_из\_БД}"

**ОСОБОЕ ВНИМАНИЕ:** Учти, что у пользователя **смешанный тип телосложения**. Сгенерируй рекомендации, которые объединят подходы для его преобладающего типа, но скорректируют их с учетом влияния второстепенных типов.

**СФОРМУЛИРУЙ РЕКОМЕНДАЦИИ:**

1. **Тренировки:** Предложи 2-3 конкретных упражнения, которые особенно эффективны для этого сочетания типов и цели. Объясни, почему они подходят именно для его смешанного профиля.
2. **Питание:** Дай совет по балансу БЖУ и калорийности, учитывающий его метаболизм и генетические склонности.
3. **Образ жизни:** Дай 1-2 совета по восстановлению.
4. **Предупреждение:** Укажи на главную ошибку, которую ему стоит избегать, исходя из его профиля.

Тон: поддерживающий, мотивирующий, профессиональный, но без сложного жаргона.

#### Команды (pattern Command)

Команда реализует паттерн Command. Каждая команда должна иметь спецификацию. Команда должна иметь полную логику действий – в случае нескольких операций, не разбивать их на подкоманды, а выполнять все операции в одной команде, заключая их в блоки try-catch.

Каждая команда реализует интерфейс Command<T> с методом  
Future<Result<T, String>> execute();

* Команда не знает о слое UI и не возвращает виджеты.
* Все входные данные передаются через конструктор и не изменяются после создания.
* Команда логирует начало, успешное завершение и ошибку уровня INFO / ERROR.
* В случае исключения, команда перехватывает его и возвращает Failure.

Все команды, имеющие бизнес-сценарии, должны быть задачами бэкенд-API.

#### Каталоги (Catalogs)

* Все каталоги являются сущностями.
* Все каталоги должны иметь списочную форму и карточку или дашборд. В списке должна быть панель с элементами управления: создание, клонирование, сохранение, удаление, поиск. В карточке аналогично должна быть панель элементов (без поиска), а также возможность изменения, удаления, выбора полей сущности.
* Если действия (создание, сохранение и т.д.) требуют внутреннюю логику, обращение к нескольким таблицам, то создавать отдельные команды.
* В списочной форме чек-бокс "Показать архивные" добавляет фильтр archived\_at IS NOT NULL. Архивные строки выделяются серым фоном.
* Все таблицы каталогов должны иметь поле замечаний: note (VARCHAR(100));
* Работа со каталогами в полном объеме (CRUD) доступна ролям "Администратор" и "Менеджер". Роль "Тренер" имеет разрешение создавать записи, изменять и удалять. Изменять и удалять записи может только свои. Роль "Инструктор" имеет разрешение на просмотр и переход. Роли "Клиент" каталоги недоступны.

##### Каталог "Виды активности клиента "

Необходим для определения коэффициента активности для расчета калорийности BMR.

###### Таблица kinds\_activity\_client

Поля:

* name (VARCHAR(100));
* coeffActivity (REAL); //Коэффициент активности.

Начальные записи:

1. Очень низкая активность (сидячий образ жизни, минимум физических нагрузок); 1,2.
2. Низкая активность (легкие физические нагрузки 1-3 раза в неделю); 1,375.
3. Средняя активность (физические нагрузки 3-5 раз в неделю); 1,55.
4. Высокая активность (интенсивные тренировки 6-7 раз в неделю); 1,725.
5. Очень высокая активность (экстремальные нагрузки, профессиональный спорт, тяжелая физическая работа); 1,9.

##### Каталог "Уровни фитнес-подготовки"

Необходим для сопоставления с планами тренировок.

###### Таблица levels\_training

Поля:

* name (VARCHAR(20));

Начальные записи:

1. Начальный.
2. Продвинутый.
3. Экспертный.

###### Класс LevelTrainingCatalog

Поля:

* name.

###### Списочная форма

Заголовок: Уровни фитнес-подготовки

Отображаемые поля:

* Наименование (name).

##### Каталог "Цели тренировок"

###### Таблица goals\_training

Поля:

* name (VARCHAR(20));

Начальные записи:

1. Похудение.
2. Набор массы.

###### Класс GoalTrainingCatalog

Поля:

* name.

###### Списочная форма

Заголовок: Цели тренировок

Отображаемые поля:

* Наименование (name).

##### Каталог "Типы телосложения "

Необходим для составления рекомендаций.

###### Таблица types\_body\_build

Поля:

* name (VARCHAR(20)); //Название: "Эктоморф", "Мезоморф", "Эндоморф";
* description (VARCHAR(200)); //Описание;
* rule\_gender (VARCHAR(20)); //Пол, для которого правило актуально: 'M', 'Ж', 'ALL';
* rule\_wrist\_max (REAL); //Максимальное значение обхвата запястья для этого типа);
* rule\_wrist\_min (REAL); //Минимальное значение обхвата запястья для этого типа;
* rule\_ankle\_max (REAL); //Максимальное значение обхвата лодыжки для этого типа;
* rule\_ankle\_min (REAL); //Минимальное значение обхвата лодыжки для этого типа;

Начальные записи:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Эктоморф | M | - | 17.5 | - | 21.5 |
| 2 | Мезоморф | M | 17.5 | 19.5 | 21.5 | 23.5 |
| 3 | Эндоморф | M | 19.5 | - | 23.5 | - |
| 4 | Эктоморф | Ж | - | 15.5 | - | 21.5 |
| 5 | Мезоморф | Ж | 15.5 | 17.5 | 21.5 | 23.5 |
| 6 | Эндоморф | Ж | 17.5 | - | 23.5 | - |

###### Класс TypeBodyBuildCatalog

Поля:

* name;
* description;
* ruleGender;
* ruleWristMax;
* ruleWristMin;
* ruleAnkleMax;
* ruleAnkleMin;

##### Каталог "Виды оборудования"

###### Таблица kinds\_equip

Поля:

* name (VARCHAR(100));

Начальные записи:

1. Снаряд.
2. Тренажер.

###### Класс KindEquipCatalog

Поля:

* name;

###### Списочная форма

Заголовок: Виды оборудования

Отображаемые поля: Наименование (name).

Способ ввода – in-line.

##### Каталог "Типы оборудования"

###### Таблица type\_equips

Поля:

* photo (URL);
* name (VARCHAR(100));
* kind\_equip\_id (BIGINT);

Начальные записи:

1. Гири, 1.
2. Гантели, 1.
3. Беговая дорожка, 2.
4. Велотренажер, 2.
5. Тренажер-платформа, 2.

###### Класс TypeEquipCatalog

Поля:

* photo;
* name;
* kindEquip;

Предусмотреть загрузку фото из ресурса или локального файла.

###### Списочная форма

Заголовок: Типы оборудования

Отображаемые поля:

* Фото (photo).
* Наименование (name).
* Вид оборудования (kindEquip.name).

###### Карточка

Заголовок: Тип оборудования

Отображаемые поля:

* Фото (photo).
* Наименование (name).
* Вид оборудования (kindEquip.name).

К полю "Вид оборудования" привязана кнопка с действием выбора вида оборудования из списка в popup-окне.

##### Каталог "Виды Упражнений"

###### Таблица kinds\_exercis

Поля:

* name (VARCHAR(100));

Начальные записи:

1. Аэробное,
2. Растягивающее,
3. Силовое,
4. Аэробное-силовое,
5. Весовое,
6. Балансировочное,
7. Дыхательное,
8. Суставное,
9. Функциональное.

###### Класс KindExercisCatalog

Поля:

* name;

###### Списочная форма

Заголовок: Виды упражнений

Отображаемые поля: Наименование (name).

###### Карточка

Заголовок: Вид упражнения

Отображаемые поля: Наименование (name).

##### Каталог "Типы Упражнений"

###### Таблица types\_exercis

Поля:

* name (VARCHAR(100));
* kind\_exercis\_id (BIGINT);
* type\_equip\_id (BIGINT);

Начальные записи:

1. Упражнения с гирями, 3, 1.
2. Упражнения с гантелями, 3, 2.
3. Упражнения на беговой дорожке, 1, 3.
4. Упражнения на велотренажере, 1, 4.
5. Упражнения на тренажере-платформе, 4, 5.
6. Приседания, 5, null
7. Отжимания, 5, null
8. Бег, 1, null.

###### Класс TypeExercisCatalog

Поля:

* name;
* kindExercis;
* typeEquip;

###### Списочная форма

Заголовок: Типы упражнений

Отображаемые поля:

* Наименование (name)
* Вид упражнения (kindExercis.name)
* Фото оборудования (typeEquip.photo).

###### Карточка

Заголовок: Тип упражнения

Отображаемые поля:

* Наименование (name)
* Вид упражнения (kindExercis.name)
* Фото оборудования (typeEquip.photo).

К полю kindExercis.name привязана кнопка с действием выбора вида упражнения из списка в popup-окне.

##### Каталог "Упражнения"

###### Таблица exercises\_templates

Поля:

* name (VARCHAR(100));
* repeatQty (INT);
* duration\_exec (REAL);
* duration\_rest (REAL);
* caloriesOut (REAL);
* is\_group (BOOL, false);
* type\_exercis\_id (BIGINT);

Начальные записи:

1. Становая тяга с гирей, 100, 5, 2, 120, false, 1.
2. Бег на дорожке, null, 20, 1, 50, false, 3.
3. Отжимания, 50, 5, 1, 100, false, 7.
4. Приседания, 50, 5, 1, 100, false, 7.
5. Становая тяга с гантелями, 100, 5, 2, 120, false, 1.
6. Велотренажер, null, 30, 1, 70, false, 4.
7. Жим ногами, 50, 60, 2, 100, false, 5.
8. Групповой бег, null, 30, 2, 120, true, 8.

###### Класс ExercisCatalog

Поля:

* name;
* repeatQty; //nullable (Количество повторов).
* durationExec; //Длительность проведения в минутах.
* durationRest; //Длительность отдыха после упражнения в минутах.
* caloriesOut; //Расход калорий в кал.
* isGroup; //Групповое упражнение.
* typeExercis; //Тип.

###### Списочная форма

Заголовок: Упражнения - шаблон

Отображаемые поля:

* Фото (typeExercis.typeEquip.photo).
* Наименование (name).
* Количество повторов (repeatQty).
* Время проведения (durationExec).
* Время отдыха (durationRest).
* Расход калорий в калориях (caloriesOut).
* Групповое (isGroup).

###### Карточка

Заголовок: Упражнение - шаблон

Отображаемые поля:

* Фото (typeExercis.typeEquip.photo).
* Наименование (name).
* Количество повторов (repeatQty).
* Время проведения (durationExec).
* Время отдыха (durationRest).
* Расход калорий в калориях (caloriesOut).
* Групповое (isGroup).

##### Каталог "Набор упражнений"

###### Таблица sets\_exercises\_templates

Поля:

* name (VARCHAR(100));
* level\_training\_id; //уровень фитнес-подготовки

###### Таблица отношений "Набор упражнений - Упражнения"

Обозначение: set\_exercises\_templates\_exercis\_templates

Поля: set\_exercises\_template\_id, exercis\_template\_id.

Начальные записи:

В таблице set\_exercises\_templates:

1. Ноги, 0.
2. Торс, 2
3. Кардио, 1.

В таблице set\_exercises\_exercis:

1, 2. 1, 4. 1, 7. 2, 1. 2, 3. 2, 5. 3, 8. 3, 6, 3, 2.

###### Класс SetExercisCatalog

Поля:

* name;
* exercisQty (int); //расчетное, кол-во упражнений.
* durationExecSum (double); //расчетное, длительность проведения всех упражнений.
* durationRestSum(double); //расчетное, длительность отдыха упражнений.
* durationSum(double); //расчетное, durationExecSum + durationRestSum.
* caloriesOutSum (double); //расчетное, расходуемые калории.

###### Списочная форма

Заголовок: Наборы упражнений - шаблон

Отображаемые поля:

* Наименование (name).
* Уровень подготовки (level\_training\_id).
* Количество (exercisQty).
* Длительность упражнений (durationExec).
* Отдых (durationRestSum).
* Длительность набора (durationSum).
* Расход калорий (caloriesOutSum).

###### Дашборд

Заголовок: Набор упражнений - шаблон

Отображаемые поля:

* Наименование (name).
* Уровень подготовки (level\_training\_id).
* Количество (exercisQty).
* Длительность упражнений (durationExec).
* Длительность отдыха между упражнениями(durationRestSum).
* Длительность набора упражнений (durationSum).
* Расход калорий (caloriesOutSum).

Элемент дашборда представляет собой фрейм с наименованием "Упражнения". Во фрейме сверху размещается панель инструментов с активной кнопкой "Добавить" и неактивной кнопкой "Удалить". Так же находится список упражнений, привязанный к данному набору. При выделении строки из списка, кнопка "Удалить" становится активной.

При нажатии на кнопку "Добавить" отображается popup-окно со списком всех упражнений, с неактивной кнопкой "Ок" и активной кнопкой "Отмена". По кнопке "Отмена", окно закрывается. При выделении строки из списка упражнений, кнопка "Ок" становится активной. По кнопке "Ок", окно закрывается и выбранное упражнение добавляется к списку упражнений.

При нажатии на кнопку "Удалить" удаляется id выбранного упражнения по id набора упражнений из таблицы set\_exercises\_exercis.

После изменения списка упражнений, все поля должны обновиться.

##### Каталог "Планы тренировок"

Каталог имеет записи плана тренировок – наборы упражнений в отношении один ко многим.

###### Таблица training\_plan\_templates

Поля:

* name (VARCHAR(100));
* goal\_training\_id (BIGINT). //из таблицы Цели тренировок

###### Таблица отношений "Планы тренировок – Наборы упражнений"

Обозначение: training\_plan\_\_templates\_set\_exercises\_templates

Поля: training\_plan\_template\_id, set\_exercises\_template\_id.

Начальные записи:

В таблице training\_plan\_templates:

1. Для похудения.
2. Набор мышечной массы.
3. Поддержание формы.

В таблице training\_plan\_\_templates\_set\_exercises\_templates:

1, 3. 2, 1. 2, 2. 3, 1. 3, 3.

###### Класс TrainingPlanCatalog

Поля:

* name;
* kindTrainingPlan; //вид плана тренировок.
* setExercisQty (int); //расчетное, кол-во наборов упражнений.
* exercisQty (int); //расчетное, кол-во упражнений.
* durationExecSum (double); //расчетное, длительность всех упражнений.
* durationRestSum(double); //расчетное, длительность отдыха упражнений.
* durationSum(double); //расчетное, durationExecSum + durationRestSum.
* caloriesOutSum (double); //расчетное, расходуемые калории.

Начальные записи:

Для похудения, 1, 10.

Набор мышечной массы, 2, 12.

Поддержание формы, 3, 15.

Для женщин, 3, 20.

###### Списочная форма

Заголовок: Планы тренировок – шаблон

Отображаемые поля:

* Наименование (name).
* Вид плана тренировок (kindTrainingPlan.name).
* Длительность упражнений (durationExec).
* Длительность отдыха между упражнениями (durationRestSum).
* Длительность набора упражнений (durationSum).
* Расход калорий (caloriesOutSum).

###### Дашборд

Заголовок: План тренировок - шаблон

Отображаемые поля:

* Наименование (name).
* Вид плана тренировок (kindTrainingPlan.name).
* Длительность упражнений (durationExec).
* Длительность отдыха между упражнениями (durationRestSum).
* Длительность набора упражнений (durationSum).
* Расход калорий (caloriesOutSum).

Элемент дашборда представляет собой фрейм с наименованием "Наборы упражнений". Во фрейме сверху размещается панель инструментов с активной кнопкой "Добавить" и неактивной кнопкой "Удалить". Так же находится список наборов упражнений, привязанный к данному плану. При выделении строки из списка, кнопка "Удалить" становится активной.

При нажатии на кнопку "Добавить" отображается popup-окно со списком всех наборов упражнений, с неактивной кнопкой "Ок" и активной кнопкой "Отмена". По кнопке "Отмена", окно закрывается. При выделении строки из списка упражнений, кнопка "Ок" становится активной. По кнопке "Ок", окно закрывается и выбранный набор упражнений добавляется к списку.

При нажатии на кнопку "Удалить" удаляется id выбранного набора упражнений по id плана из таблицы training\_plan\_set\_exercises.

После изменения списка упражнений, все поля должны обновиться.

##### Каталог "Заявки (Лид-менеджмент)"

###### Таблица onboarding\_client

Поля:

* first\_name (VARCHAR(100));//Имя
* last\_name (VARCHAR(100));//Фамилия
* middle\_name (VARCHAR(100); //Отчество
* gender (VARCHAR(20)); //'М', 'Ж'
* age (int); //Возраст
* phone (VARCHAR(20)); //Телефон
* email (VARCHAR(100); //E-mail
* feedback\_method (VARCHAR(20)); //способ обратной связи: 'телефон' или 'почта'
* is\_contact\_on\_website (bool); //по умолчанию – false, признак холодного контакта клиента через вебсайт
* is\_client\_build (bool); //признак, что на основе записи создан клиент
* client\_build\_at (TIMESTAMP); //дата-время создания клиента

###### Класс OnboardingClientCatalog

Поля:

* firstName (string);//Имя
* lastName (string);//Фамилия
* middleName (string); //Отчество
* gender (string); //'М', 'Ж'
* age (int); //Возраст
* phone (string); //Телефон
* email (string); //E-mail
* feedbackMethod(string); //способ обратной связи: 'телефон' или 'почта'
* isContactOnWebsite (bool); //признак холодного контакта клиента через вебсайт
* isClientBuild (bool); //признак создания клиента
* clientBuildAt (datetime); // дата-время создания клиента
* archived\_at (Timestamp)
* archived\_by (BIGINT)
* note (VARCHAR(100)); //примечание

###### Списочная форма

Заголовок: Заявки (Лид-менеджмент)

Отображаемые поля:

* Имя (firstName)
* Фамилия (lastName)
* Отчество (middleName)
* Пол (gender)
* Возраст (age)
* Телефон (phone)
* E-mail (email)
* Cпособ обратной связи (feedbackMethod)
* Активное (isActive)

Форма содержит панель инструментов с поиском по имени, фамилии, телефону, почте. Так же содержит чекбокс по полю isActive, по умолчанию true.

###### Карточка

Заголовок: Заявка (Лид-менеджмент)

Отображаемые поля:

* Имя (firstName)
* Фамилия (lastName)
* Отчество (kindEquip.name)
* Пол (gender)
* Возраст (age)
* Телефон (phone)
* E-mail (email)
* Зашел через сайт (isContactOnWebsite)
* Cпособ обратной связи (feedbackМethod)
* Активное (isActive)
* Примечание менеджера (note)

Обязательность полей при создании записи:

* firstName;
* gender;
* age;
* phone или email. Обязательное хотя бы одно из двух.

Карточка содержит кнопку "Создать клиента", которая вызывает функцию-команду создания клиента. При успешном создании, заполняются поля isClientBuild, ClientBuildAt и isActive.

##### Каталог "Расписание работы центра"

###### Таблица work\_schedules

Поля:

* day\_of\_week (Integer, NOT NULL, UNIQUE). День недели: 1 (Понедельник) - 7 (Воскресенье).
* start\_time (Time, NOT NULL). Время начала работы (например, 09:00).
* end\_time (Time, NOT NULL). Время окончания работы (например, 21:00).
* is\_day\_off (Boolean, default: false). Флаг, указывающий, что этот день недели по умолчанию нерабочий (например, Воскресенье).

Начальные записи:

1. (1, '09:00', '21:00', false)
2. (2, '09:00', '21:00', false)
3. (3, '09:00', '21:00', false)
4. (4, '09:00', '21:00', false)
5. (5, '09:00', '21:00', false)
6. (6, '09:00', '21:00', false)
7. (7, '09:00', '21:00', false)

###### Класс WorkSchedule

Поля:

* dayOfWeek;
* startTime;
* endTime;
* IsDayOff

##### Каталог "Дежурства сотрудников"

Каталог имеет записи дежурства сотрудников по центру.

###### Таблица employees\_duty

Поля:

* user\_id(BIGINT). //из таблицы users
* duty\_start\_at (TIMESTAMP); //время начала дежурства
* duty\_finish\_at (TIMESTAMP); //время окончания дежурства

###### Класс EmployeeDuty

Поля:

* userId; //сотрудник
* dutyManagerStartAt (DateTime); //время начала дежурства
* dutyManagerFinishAt (DateTime); //время окончания дежурства

###### Списочная форма

Заголовок: "Дежурства сотрудников"

Отображаемые поля:

* ФИО (userId).
* Время начала (dutyManagerStartAt).
* Время окончания (dutyManagerFinishAt).

###### Дашборд

Заголовок: "Дежурство сотрудника" + ФИО

Отображаемые элементы:

* ФИО (userId).
* ListView по userId с полями Время начала, Время окончания.

##### Каталог "Квалификации"

Каталог имеет записи квалификаций тренеров и инструкторов.

###### Таблица employees\_qualifications

Поля:

* user\_id (BIGINT); //из таблицы users
* education\_level (SMALLINT); //enum employeeEducation ('высшее', 'среднее', 'курсы')
* specialization (SMALLINT); //enum employeeSpec ('фитнес', 'йога', 'реабилитация')
* qualification\_date (TIMESTAMP); //дата присвоения квалификации
* education\_document (URL); //скан диплома/сертификата

###### Класс employeeQualification

Поля:

* userId; //сотрудник
* educationLevel (Enum); //enum employeeEducation
* specialization (Enum); //enum employeeSpec
* qualificationDate (DateTime);
* educationDocument (Image); //скан документа

###### Списочная форма

Заголовок: "Квалификации"

Отображаемые поля:

* ФИО (userId);
* Уровень (educationLevel );
* Специализация (specialization);
* Дата квалификации (qualificationDate);

###### Дашборд

Заголовок: "Квалификация" + ФИО

Отображаемые элементы:

* ФИО (userId);
* Уровень (educationLevel );
* Специализация (specialization);
* Дата квалификации (qualificationDate);
* Скан (educationDocument);

#### Оффлайн-режим (Offline-First Approach)

##### 1. Цель и Область применения

Система должна быть способна функционировать с ограниченным или полностью отсутствующим интернет-соединением для выполнения критически важных бизнес-процессов. Основной фокус — на мобильном приложении для ролей **Тренер** и **Инструктор**, которые работают непосредственно в зале, где связь может быть нестабильной.

##### 2. Архитектура оффлайн-работы

* **Стратегия:** Offline-First. Приложение по умолчанию пытается сохранять данные локально, а затем синхронизировать их с сервером при наличии соединения.
* **Локальное хранилище:** Использование **Hive** или **Sembast** для быстрого и надежного локального хранения структурированных данных.
* **Механизм синхронизации:** Фоновая синхронизация при восстановлении соединения с использованием очереди (queue) отложенных запросов.

##### 3. Функциональность, доступная оффлайн

**Для Тренера/Инструктора:**

* **Просмотр расписания:** Доступ к расписанию на текущий и следующий день.
* **Просмотр карточек клиентов:** Основная информация, актуальные планы тренировок, последние замеры.
* **Фиксация занятий:**
  + Отметка начала и окончания занятия (start\_fact\_at, finish\_fact\_at).
  + Ввод фактических параметров занятия (примечания, изменение упражнений "на лету").
  + Ввод weight и calories\_in для клиента после занятия.
* **Работа с чатом:** Просмотр истории чатов, отправка сообщений (доставляются при появлении сети).

**Для Клиента:**

* Просмотр своего расписания и деталей ближайшего занятия.
* Просмотр своего дневника калорий и прогресса.
* Ввод данных о потребленных калориях и весе.

##### 4. Процесс синхронизации

1. **Обнаружение сети:** Приложение автоматически определяет восстановление стабильного интернет-соединения.
2. **Выполнение очереди:**
   * Данные, созданные или измененные оффлайн, помещаются в специальную таблицу pending\_sync\_operations (локально).
   * При появлении сети система последовательно отправляет эти данные на сервер.
3. **Разрешение конфликтов:**
   * **Стратегия:** "Последнее изменение побеждает" (Last Write Wins - LWW) на основе меток времени updated\_at.
   * Если конфликт не может быть разрешен автоматически (например, изменение одних и тех же данных разными пользователями оффлайн), запись помечается как требующая ручного разрешения администратором/менеджером.

##### 5. Ограничения оффлайн-режима

* **Невозможно оффлайн:**
  + Первоначальная регистрация и вход в систему (требует аутентификации).
  + Составление нового расписания.
  + Генерация сложных отчетов.
  + Получение онлайн-рекомендаций от ИИ.
  + Работа с каталогами (создание/редактирование шаблонов упражнений).
* **Уведомления:** Push-уведомления не доставляются оффлайн. Локальные уведомления (напоминания о начале занятия) работают.

##### 6. Требования к данным

* **Кэширование при запуске:** При наличии сети приложение заранее кэширует данные, необходимые для работы в течение дня:
  + Активное расписание пользователя.
  + Карточки прикрепленных клиентов (тренеру).
  + Активные планы тренировок.
  + Последние сообщения в чатах.
* **Политика устаревания кэша:** Локальные данные считаются актуальными в течение 24 часов. При попытке работы с устаревшим кэшем система показывает предупреждение.

##### 7. Пользовательский интерфейс

* **Индикатор статуса:** В верхней части интерфейса всегда отображается значок:
  + Зеленый / "Online" — стабильное соединение.
  + Желтый / "Синхронизация..." — идет процесс синхронизации.
  + Красный / "Оффлайн" — соединение отсутствует.
* **Уведомления:** При отправке данных в оффлайне появляется сообщение "Сообщение будет отправлено при подключении к сети". После успешной синхронизации — "Все данные синхронизированы".

##### 8. Техническая реализация

dart

// Примерная структура очереди синхронизации

class PendingSyncOperation {

String id;

String tableName; // 'lessons', 'track\_calories', etc.

String operation; // 'INSERT', 'UPDATE'

Map<String, dynamic> data;

DateTime timestamp;

int retryCount;

}

#### Архивация и восстановление записей

* 1. В каждой таблице обязательно присутствуют два служебных поля:

archived\_at timestamp with time zone NULL – дата/время архивации;

archived\_by bigint NULL – ссылка на users(id), кто архивировал.  
NULL в обоих полях означает «запись активна».

* 1. Архивация выполняется командой ArchiveEntityCommand (паттерн Command), которая:

проверяет право «Архивировать» у текущей роли;

заполняет archived\_at = now(), archived\_by = current\_user\_id;

пишет запись в audit\_log (действие = ‘ARCHIVE’).

* 1. Восстановление – командой RestoreEntityCommand, которая:

сбрасывает archived\_at и archived\_by в NULL;

пишет в audit\_log действие ‘RESTORE’.

* 1. На всех таблицах создаётся RLS-политика (или эквивалентный механизм БД):

sql

USING (archived\_at IS NULL)

Политика действует для ролей «Клиент», «Инструктор», «Тренер», «Менеджер».  
Роль «Администратор» видит записи без ограничений.

* 1. Списочные формы по умолчанию не выводят архивные строки. Чек-бокс «Показать архивные» добавляет условие archived\_at IS NOT NULL.
  2. Экспорт/отчёты/API-методы, не предназначенные для администратора, автоматически фильтруют archived\_at IS NULL.
  3. Физическое удаление (DELETE) запрещено; используется только логическое удаление через архивацию.

#### Аудит и журнал событий (Audit Log)

* 1. **Цель**  
     Обеспечить полную прослеживаемость изменений любых данных системы (клиенты, сотрудники, расписания, справочники, архивация/восстановление) без физического хранения версий строк.
  2. **Единая таблица аудита**  
     Создаётся одна таблица на всю БД:

sql

CREATE TABLE audit\_log (

id bigserial PRIMARY KEY,

table\_name text NOT NULL, -- имя таблицы-источника

row\_pk bigint NOT NULL, -- PK изменённой строки

action text NOT NULL CHECK (action IN (

'INSERT','UPDATE','DELETE','ARCHIVE','RESTORE')),

changed\_at timestamptz DEFAULT now(),

changed\_by uuid REFERENCES users(id),

old\_values jsonb, -- предыдущие значения (NULL при INSERT)

new\_values jsonb -- новые значения (NULL при DELETE)

);

* 1. **Триггеры**  
     Для каждой таблицы автоматически создаются триггеры AFTER INSERT, AFTER UPDATE, AFTER DELETE (через миграцию или генератор кода).  
     Триггер:

заполняет table\_name = TG\_TABLE\_NAME,

row\_pk = NEW.id / OLD.id,

action по типу операции,

changed\_by = current\_user\_id() (из RLS-сессии),

old\_values = to\_jsonb(OLD), new\_values = to\_jsonb(NEW).  
При архивации/восстановлении (команды ArchiveEntityCommand / RestoreEntityCommand) триггер фиксирует action = 'ARCHIVE' или 'RESTORE'.

* 1. **Доступ к журналу**
  + Просмотр аудита доступен только роли Администратор.
  + В карточке любой сущности кнопка «История» открывает список записей audit\_log отфильтрованных по table\_name + row\_pk, отсортированных по changed\_at DESC.
  + Для удобства чтения поля old\_values/new\_values отображаются в виде сворачиваемого JSON-дерева; значения дат/чисел форматируются по текущей локали.
  1. **Хранение и ротация**

Строки хранятся 36 месяцев.

Ежедневным cron-Job (pg\_cron или внешний планировщик) удаляются записи старше 36 мес:

sql

DELETE FROM audit\_log WHERE changed\_at < now() - interval '36 months';

* 1. **Производительность**

Индексы:  
(table\_name, row\_pk, changed\_at DESC) – быстрая выборка истории строки.  
(changed\_at) – быстрая чистка старых данных.

Триггеры объявляются FOR EACH ROW, но только при первичной записи (без каскадных UPDATE-ов) чтобы избежать рекурсии.

* 1. **Интерфейс**
  + Списочная форма «Аудит» (доступна только администратору): фильтры по периоду, таблице, пользователю, типу действия; экспорт в CSV/XLSX.
  + Виджет «Последние изменения» на дашборде администратора: 10 последних записей системы.
  1. **Безопасность**
  + Таблица audit\_log не подлежит архивации (archived\_at не добавляется).
  + Прямое изменение/удаление строк audit\_log запрещено триггером BEFORE DELETE / UPDATE (только SELECT и INSERT).
  + Маскировка чувствительных данных: перед записью в new\_values/old\_values поля password\_hash, sms\_code, jwt\_refresh\_token обнуляются ('::text = '\*\*\*').

### Нефункциональные требования

#### Требования к технологическому стеку

Технологический стек должен обеспечивать полную независимость от внешних BaaS-провайдеров, отсутствие контейнеризации, соответствие российским требованиям безопасности и возможность развертывания на собственной инфраструктуре заказчика.

##### Серверная часть

###### **Backend Framework**

\*\*Основной стек:\*\*

- \*\*Язык:\*\* Dart 3.0+

- \*\*Фреймворк:\*\* Dart Frog или Shelf

- \*\*HTTP сервер:\*\* Dart native (dart:io)

- \*\*Реальное время:\*\* WebSocket + Redis Pub/Sub

###### **База данных**

\*\*Основная БД:\*\*

- \*\*Система:\*\* PostgreSQL 15+

- \*\*Хостинг:\*\* Отдельный сервер или VPS

- \*\*Миграции:\*\* Dart-пакет `moor` или собственные скрипты

\*\*Расширения PostgreSQL:\*\*

- pg\_trgm - для полнотекстового поиска

- btree\_gin - для составных индексов

- uuid-ossp - для генерации UUID

\*\*Репликация:\*\*

- Настройка master-slave репликации

- Автоматические бэкапы

###### **Кэширование и реальное время**

\*\*Redis сервер:\*\*

- \*\*Назначение:\*\* Pub/Sub для реального времени, кэширование

- \*\*Конфигурация:\*\* Отдельный инстанс Redis 7+

- \*\*Использование:\*\*

- WebSocket сообщения через Pub/Sub

- Кэш сессий пользователей

- Временное хранение токенов

- Очереди задач

##### Клиентская часть

###### **Мобильные приложения**

\*\*Фреймворк:\*\* Flutter 3.0+

- \*\*Язык:\*\* Dart 3.0+

- \*\*Состояние:\*\* Riverpod 2.0

- \*\*Навигация:\*\* Go Router

- \*\*Локальное хранилище:\*\* Hive 2.0

\*\*Поддерживаемые платформы:\*\*

- Android 8.0+ (API 26+)

- iOS 13.0+

- \*\*Сборка:\*\* Ручная компиляция APK/IPA

###### **Веб-приложение**

\*\*Фреймворк:\*\* Flutter Web

- \*\*Режим сборки:\*\* CanvasKit для производительности

- \*\*Роутинг:\*\* Browser history API

- \*\*PWA:\*\* Service Worker для оффлайн-работы

\*\*Хостинг:\*\*

- Статические файлы на nginx

- NAS хранилище для медиа-файлов

##### Инфраструктура

###### **Серверное окружение**

\*\*Операционная система:\*\*

- \*\*Основная:\*\* Ubuntu 22.04 LTS

- \*\*Альтернативная:\*\* Debian 11+

\*\*Веб-сервер:\*\*

- \*\*Прокси:\*\* Nginx 1.20+

- \*\*SSL:\*\* Let's Encrypt или коммерческий сертификат

- \*\*Балансировка:\*\* На уровне nginx (при масштабировании)

\*\*Управление процессами:\*\*

- \*\*Инит-система:\*\* Systemd

- \*\*Логирование:\*\* Journald + ротация логов

- \*\*Мониторинг:\*\* Prometheus + Grafana (опционально)

###### **Файловое хранилище**

\*\*Локальное хранилище:\*\*

- \*\*Структура:\*\*

`/opt/fitman/storage/`

├── uploads/ # Пользовательские файлы

├── avatars/ # Фотографии профилей

├── documents/ # Сканы документов

└── backups/ # Резервные копии

##### Установка и развертывание

###### **Способ дистрибуции**

\*\*Серверная часть:\*\*

- \*\*Формат:\*\* Самодостаточный инсталлятор (.run)

- \*\*Содержимое:\*\*

- Бинарные файлы Dart приложения

- Скрипты установки PostgreSQL/Redis

- Конфигурационные файлы nginx

- Systemd unit files

\*\*Мобильные приложения:\*\*

- \*\*Android:\*\* APK файлы для скачивания с сайта

- \*\*iOS:\*\* TestFlight для тестирования, App Store для продакшена

##### Процесс установки

bash

# Пример установочного скрипта

#!/bin/bash

# fitman-installer.run

echo "Установка Фитнес-Менеджера..."

# Автоматическая установка зависимостей

apt-get install -y postgresql postgresql-contrib redis-server nginx

# Настройка БД

sudo -u postgres psql -c "CREATE USER fitman\_user WITH PASSWORD '${DB\_PASSWORD}';"

sudo -u postgres psql -c "CREATE DATABASE fitman OWNER fitman\_user;"

# Развертывание приложения

cp -r application /opt/fitman/

chown -R www-data:www-data /opt/fitman

# Запуск сервисов

systemctl enable fitman.service

systemctl start fitman.service

##### Безопасность

###### **Аутентификация и авторизация**

\*\*Механизм аутентификации:\*\*

- \*\*Токены:\*\* JWT (JSON Web Tokens)

- \*\*Алгоритм:\*\* HS256 или RS256

- \*\*Время жизни:\*\* Access token - 24 часа, Refresh token - 7 дней

\*\*Защита API:\*\*

- \*\*HTTPS:\*\* Обязательно для всех endpoints

- \*\*CORS:\*\* Настройка для доменов клиента

- \*\*Rate Limiting:\*\* Ограничение запросов по IP/пользователю

##### Защита данных

\*\*Шифрование:\*\*

- \*\*Пароли:\*\* bcrypt с salt rounds = 12

- \*\*Чувствительные данные:\*\* AES-256 в базе данных

- \*\*Транспорт:\*\* TLS 1.3 для всех соединений

\*\*Резервное копирование:\*\*

- \*\*Частота:\*\* Ежедневные полные бэкапы

- \*\*Хранение:\*\* Отдельный защищенный сервер

- \*\*Шифрование:\*\* GPG для резервных копий

##### Мониторинг и логирование

###### **Система мониторинга**

\*\*Встроенные метрики:\*\*

- Нагрузка CPU, памяти, диска

- Количество активных пользователей

- Время ответа API endpoints

- Статус WebSocket соединений

\*\*Внешний мониторинг:\*\*

- \*\*Доступность:\*\* Zabbix или UptimeKuma или UptimeRobot

- \*\*Логи:\*\* Centralized logging через rsyslog

- \*\*Оповещения:\*\* Telegram bot для критических ошибок

###### **Логирование**

dart

// Пример системы логирования

class AppLogger {

static final Logger \_logger = Logger();

static void setupLogging() {

Logger.root.level = Level.INFO;

Logger.root.onRecord.listen((record) {

// Запись в файл с ротацией

\_writeToFile(record);

// Отправка в syslog для критических ошибок

if (record.level >= Level.SEVERE) {

\_sendToSyslog(record);

}

});

}

}

##### Производительность и масштабирование

###### **Оптимизация базы данных**

sql

-- Пример индексов для производительности

CREATE INDEX CONCURRENTLY idx\_users\_email ON users(email);

CREATE INDEX CONCURRENTLY idx\_chats\_company ON chats(company\_id);

CREATE INDEX CONCURRENTLY idx\_messages\_chat\_date ON messages(chat\_id, created\_at DESC);

-- Частичные индексы для архивных данных

CREATE INDEX idx\_active\_users ON users(id) WHERE archived\_at IS NULL;

###### **Стратегия масштабирования**

\*\*Вертикальное масштабирование:\*\*

- Увеличение ресурсов VPS (CPU, RAM, SSD)

- Настройка репликации PostgreSQL

- Кэширование часто запрашиваемых данных

\*\*Горизонтальное масштабирование (при росте):\*\*

- Добавление application-серверов

- Балансировка нагрузки через nginx

- Шардирование базы данных

##### Резервные копии и восстановление

###### **Процедуры бэкапа**

\*\*Автоматические бэкапы:\*\*

- \*\*База данных:\*\* pg\_dump ежедневно в 02:00

- \*\*Файлы приложения:\*\* rsync раз в неделю

- \*\*Конфигурации:\*\* Git-репозиторий с историей изменений

\*\*Восстановление:\*\*

- Скрипт автоматического восстановления из бэкапа

- Пошаговые инструкции для ручного восстановления

- Регулярные тесты восстановления

#### Требования к таблицам

База данных системы – PostgreSQL.

* **В каждой таблице** должно быть поле id (BIGSERIAL) с primary key.  
  Имена foreign key полей должны соответствовать шаблону: <имя\_таблицы\_в\_единственном\_числе>\_id.
* **В каждой таблице** должно быть поле company\_id (BIGINT) для мультитенантности. По умолчанию равно -1.
* **Индексы** должны быть добавлены на все поля, используемые в условиях JOIN и WHERE, в частности, на все foreign key поля.
* Имена и поля таблиц должны быть в snake\_case нотации.
* **Способ удаления записей** -- логическая архивация: заполняются поля archived\_at, archived\_by. Физический DELETE запрещён.
* Поле для фото – тип URL, ссылка на файл с фото в хранилище
* **Не использовать встроенный тип ENUM**. Вместо него использовать SMALLINT, в нем значение enum в коде.
* **Не использовать строковый тип как перечисления**. Вместо него использовать SMALLINT, в нем значение enum в коде.
* **Для оперативного аудита,** все таблицы должны иметь поля:
* "Дата Создания" (created\_at (TIMESTAMP))
* "Дата Изменения" (updated\_at (TIMESTAMP))
* "Создан пользователем" (created\_by (BIGINT))
* "Изменен пользователем" (updated\_by (BIGINT)).
* **Поля** created\_at, created\_by, необходимо учитывать при добавлении записи. Поля updated\_at, updated\_by, необходимо учитывать при обновлении запросов.
* **Для архивации и восстановления записей**, все таблицы должны включать поля:
* archived\_at (TIMESTAMP NULL)
* archived\_by (BIGINTNULL FK → users(id))

#### Требования к списочным формам и карточкам

Все ListView должны быть пагинированы.

В списке и карточке в панели инструментов должна быть кнопка "Инфо", по клику которой в popup-окне выводится диагностическая информация: имя таблицы сущности, имя представления, id записи таблицы. Так же выводятся поля: created\_at, updated\_at, updated\_at, updated\_by. Данная кнопка доступна только тренеру и администратору.

#### Требования к логированию приложения

Логирование должно осуществляться средствами технологического стека. Логирование должно иметь 4 уровня важности сообщения: отладочное, информационное, предупреждение, ошибка. Максимальный уровень, по которому сообщения будут записываться, записывается в файл настроек приложения. Сообщения должны записываться в суточный файл формата "EFT\_YYYY\_MM\_DD", где YYYY\_MM\_DD – год, месяц и день. Сообщение, кроме отладочного, должно выводиться на экран.

#### Требования к безопасности

Данный раздел описывает меры, направленные на обеспечение конфиденциальности, целостности и доступности данных системы, а также на защиту от несанкционированного доступа.

##### Аутентификация и авторизация

* Многофакторная аутентификация (MFA). Система должна предоставлять возможность включения MFA для ролей Администратор, Менеджер и Тренер.
* Политика паролей. Пароли пользователей должны храниться в базе данных исключительно в виде хешей с использованием современных алгоритмов (например, bcrypt/scrypt/Argon2).
* Минимальная длина пароля — 8 символов.
* Пароль должен содержать как минимум одну заглавную букву, одну строчную букву, одну цифру и один специальный символ.
* Запрещено использовать простые или скомпрометированные пароли.
* Защита от brute-force. Система должна блокировать учетную запись или вводить задержку после 5 неуспешных попыток ввода пароля в течение 15 минут.
* JWT Tokens. Время жизни Access Token должно быть ограничено (например, 15-30 минут).
* Обязательно использование Refresh Tokens с ограниченным временем жизни (например, 7 дней) и механизмом их отзыва.
* Tokens должны быть подписаны с использованием надежного алгоритма (например, RS256).
* Управление сессиями. При выходе из системы или по истечении времени неактивности (30 минут) сессия пользователя должна быть полностью завершена на стороне клиента и сервера.

##### Контроль доступа (Authorization)

* Принцип наименьших привилегий. Каждая роль в системе должна иметь доступ только к тем данным и функциям, которые абсолютно необходимы для выполнения ее задач.
* Row Level Security (RLS) Для базы данных PostgreSQL \*\*обязательно\*\* должна быть настроена политика RLS на всех таблицах, содержащих персональные или бизнес-данные. Доступ к данным должен предоставляться на основе роли пользователя и его идентификатора.
* Валидация на стороне сервера. Все запросы к API, включая проверки прав доступа к объектам, должны быть валидированы на стороне бэкенда. Недостаточно проверять права только на фронтенде.

##### Защита данных

* Все данные должны передаваться по защищенному протоколу \*\*HTTPS/TLS 1.2+\*\*.
* Чувствительные данные (например, пароли, персональные данные клиентов) должны шифроваться при хранении в базе данных или использовать механизмы шифрования.
* Маскирование данных. При отображении конфиденциальной информации (например, телефона, email) в интерфейсах для некоторых ролей следует применять маскирование (например, `+7 \*\*\* \*\*\* \*\* 12`).
* Безопасное хранение секретов. Ключи API, секреты для подписи JWT, учетные данные для внешних сервисов (SMS-шлюз) не должны храниться в коде или репозитории. Необходимо использовать защищенные механизмы хранения секретов.
* Доступ к архивным записям возможен только роли «Администратор». Остальные роли видят только активные записи (RLS-политика archived\_at IS NULL).

##### Безопасность API

* Защита от перебора (Rate Limiting). API должен быть защищен от DDoS-атак и перебора паролей путем ограничения количества запросов с одного IP-адреса или для одной учетной записи в единицу времени.
* Валидация входных данных. Все входные данные, поступающие в API (параметры запросов, тела запросов), должны быть строго валидированы на предмет типа, длины, диапазона и наличия потенциально опасных конструкций (SQL-инъекции, XSS).

##### Защита на клиенте (Frontend).

* Защита от XSS. Данные, предоставляемые пользователями (например, комментарии, имена), должны быть экранированы перед отображением в интерфейсе для предотвращения межсайтового скриптинга.
* Безопасные заголовки HTTP. Приложение должно использовать security headers (например, Content-Security-Policy, X-Frame-Options, X-Content-Type-Options) для защиты от различных видов атак.

##### Аудит и мониторинг

* Логирование событий безопасности. В логи приложения должны записываться все критичные события:
* Неудачные попытки входа в систему.
* Попытки доступа к запрещенным ресурсам (ошибки 403).
* Изменение критичных настроек системы, прав пользователей.
* Создание, изменение и удаление учетных записей пользователей.
* Мониторинг. Система должна иметь возможность мониторинга подозрительной активности, с уведомлением ответственных лиц (Администратор).

##### Резервное копирование и восстановление

* Регулярное резервное копирование. Должна быть обеспечена процедура регулярного (ежедневного) резервного копирования базы данных и критичных файлов.
* План восстановления. Должен быть документирован и протестирован план восстановления работы системы после сбоя в течение не более 4 часов.

#### Требования к тестированию

* Реализовать юнит-тесты по всем командам.
* Реализовать интеграционные тесты по каждой роли.

#### Дополнительные требования

##### Добавить диаграмму БД.

* ER-диаграмму отношений между таблицами
* Схему связей и индексов

##### 2. Уточнить API endpoints.

* REST API методы для каждого модуля
* Форматы запросов/ответов

### Архитектурные требования

#### Редакции продукта

##### 1. Редакция Basic

Эта редакция является текущей.

**Основной функционал:**

* Для будущей мультитенантности вводится поле company\_id во все таблицы.
* Система лицензий и ограничений.
* Весь функционал в Basic-качестве.

**Для:** малых фитнес-залов, студий, персональных тренеров  
**Ограничения:** 1 филиал, до 100 клиентов, до 3 тренеров

##### 2. Редакция Pro

**Основной функционал:**

* Мультитенантная архитектура.
* Расширенная аналитика.
* White-label.
* API для интеграций.

**Для:** средних клубов, сетей в одном городе, мультибрендовых операторов  
**Ограничения:** мультитенантность, до 500 клиентов, до 10 тренеров

##### Редакция Enterprise

**Основной функционал:**

* + Филиальная и мультитенантная архитектура.
  + Корпоративная отчетность.
  + Сложные интеграции.

**Для:** федеральных сетей, крупных клубов с филиалами  
**Ограничения:** неограниченно, филиальная структура

##### Архитектура с учетом трех версий

**Базовая версия:**

- Единая база данных

- Простая схема tenant\_id = 1

- Ограничения через application logic

**Pro версия:**

- Schema-based мультитенантность

- Отдельные схемы для каждого tenant

- Общие справочники

**Enterprise версия:**

- База данных на филиал + общая сводная

- Репликация между филиалами

- Единый каталог пользователей

### Глоссарий

* Списочная форма – виджет-экран со списочным представлением (ListView).
* Карточка – виджет-экран с детальным представлением (DetailView).
* Дашборд – виджет-экран с вложенными виджетами.
* Фрейм – виджет-контейнер для группировки других виджетов.
* BMR - Basal Metabolic Rate (Основной обмен веществ).
* Дефицит/профицит калорий - разница между потребленными и затраченными калориями.
* Мультитенантность - архитектура для работы с несколькими независимыми клиентами в одной системе
* Филиальность - иерархическая структура подразделений с единым управлением