# Спецификация веб-приложения «Workout Logger» — v0.2 (MVP)

> Обновлено 16 мая 2025 с учётом ответов заказчика.

---

## 1. Цель проекта

Создать автономное веб-приложение учёта силовых тренировок, развёртываемое на ноутбуке под **Linux**. Приложение доступно из браузера смартфона по Wi-Fi и в перспективе поддерживает многопользовательский режим.

## 2. Сценарии использования (User Stories)

1. **Регистрация и вход** — пользователь создаёт учётную запись и логинится.
2. **Начать тренировку** — выбрать одну или несколько групп мышц.
3. **Записать подход** — выбрать упражнение, указать повторения и вес, сохранить.
4. **Добавить ещё один подход** того же упражнения (автосуммирование N×reps×weight).
5. **Добавить другое упражнение** — выбор нового упражнения, предыдущие результаты видны.
6. **Завершить тренировку** — тренировка фиксируется и появляется в истории.
7. **Просмотреть историю** — список сеансов; внутри — детализация сетов.
8. **Посмотреть предыдущий результат упражнения** — на экране тренировки рядом с селектором упражнения показывается «последний вес × повторы».
9. **Редактировать базу упражнений** — CRUD-интерфейс упражнений.
10. **Выход из учётной записи**.

## 3. Функциональные требования

| ID | Требование |
|---|---|
| FR-01 | Регистрация/логин через форму (username + пароль). |
| FR-02 | Старт тренировок с выбором 1+ мышечных групп. |
| FR-03 | Запись неограниченного числа подходов. |
| FR-04 | Автоматическое агрегирование одинаковых подходов (N×reps×weight). |
| FR-05 | Кнопки: «Сохранить подход», «Другое упражнение», «Закончить тренировку». |
| FR-06 | История тренировок доступна по дате. |
| FR-07 | CRUD упражнений через UI. |
| FR-08 | Отображение **последнего результата** по выбранному упражнению. |
| FR-09 | Все данные сохраняются в SQLite немедленно после действия. |
| FR-10 | Возможность множественных пользователей (таблица User). |
| FR-11 | Экспорт/импорт, синхронизация — **в последующих версиях**. |

# 4. Нефункциональные требования

- **Локальная работа**, offline-first.
- **Linux**; запуск одним скриптом (`uvicorn app:app --host 0.0.0.0 --port 8000`).
- **Кросс-браузерность**, mobile-first UI.
- **Простота установки**: Python ≥3.11, `pip install -r requirements.txt`.
- **Безопасность**: пароли хранятся как Bcrypt-хеши.
- **Производительность**: латентность API <100 мс.

# 5. Архитектура и стек технологий

| Слой | Библиотека / фреймворк | Причины |
|---|---|---|
| Backend | **Python 3.12 + FastAPI** | Асинхронность, типизация, OpenAPI-документация. |

| | | |
|---|---|---|
| ORM | **SQLAlchemy 2 + aiosqlite** | Работа с SQLite, миграции Alembic. |
| Auth | **passlib[bcrypt] + fastapi-users** | Готовые эндпоинты регистрации/логина. |
| Frontend | **Vue 3 (CDN) + Pinia** | Небольшой bundle, реактивность. |
| UI | **Tailwind CSS** | Utility-first, быстрая адаптивная верстка. |
| Charts (later) | Chart.js | Для будущей статистики. |

# 6. Модель данных (ER-диаграмма)

User (1) ——— (N) Workout ——— (N) Set ─ (1) Exercise

- **User**: id, username, password_hash, created_at
- **Workout**: id, user_id, dt_start, dt_end, muscle_groups (JSON)
- **Exercise**: id, name, default_muscle_group
- **Set**: id, workout_id, exercise_id, reps, weight_kg

Доп. индекс: `(exercise_id, workout_id, reps, weight_kg)` для агрегирования.

# 7. API (MVP)

| Метод | URL | Request | Response |
|---|---|---|---|
| POST | /auth/register | {username, password} | 201 |
| POST | /auth/jwt/login | {username, password} | {access_token} |
| GET | /api/exercises | — | `[ {id, name, muscle_group} ]` |
| POST | /api/exercises | {name, muscle_group} | 201 |
| PUT | /api/exercises/{id} | … | 204 |
| DELETE | /api/exercises/{id} | — | 204 |

| | | | |
|---|---|---|---|
| POST | /api/workouts | {muscle_groups: []} | {workout_id} |
| POST | /api/workouts/{id}/sets | {exercise_id, reps, weight} | {set_id} |
| POST | /api/workouts/{id}/finish | — | 204 |
| GET | /api/workouts | — | список тренировок текущего пользователя |
| GET | /api/exercises/{id}/last | — | {date, reps, weight} |

JWT-токен передаётся в `Authorization: Bearer <token>`.

# 8. UI-потоки

## 8.1 Логин/Регистрация

Формы `/login`, `/register`.

## 8.2 Главная

- «Начать тренировку»
- «История»
- «Упражнения»
- Статус пользователя + «Выход»

## 8.3 Экран тренировки

- Заголовок: дата, группы мышц.
- **Форма сета**: выбор упражнения (autocomplete), «Последний: 8×80 кг» справа, поля Reps/Weight, «Сохранить подход».
- Таблица сетов ниже (агрегированная).
- Кнопки: «Другое упражнение», «Закончить тренировку».

## 8.4 История

Список: дата, длит., упражнения (count), кнопка «Открыть». Подробный просмотр — развёрнутый список сетов.

### 8.5 CRUD упражнений

Таблица + модальное окно «Добавить/Редактировать».

# 9. Развёртывание

sudo apt-get install python3.12 python3.12-venv git
git clone https://github.com/<repo>/workout-logger
cd workout-logger
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
uvicorn app:app --host 0.0.0.0 --port 8000

На телефоне: `http://<IP-ноутбука>:8000`.

# 10. Тестирование

- Pytest для моделей и API.
- Postman коллекция.
- Mobile DevTools.

# 11. План MVP

| Этап | Задачи | Длительность |
|---|---|---|
| 1 | Инициализация проекта, модели SQLAlchemy | 0.5 дня |
| 2 | Auth (fastapi-users) | 0.5 дня |
| 3 | Endpoints Workouts/Sets | 1 день |
| 4 | CRUD Exercises | 0.5 дня |
| 5 | Frontend Vue + Tailwind, страницы Login/Главная | 1 день |
| 6 | Экран тренировки + агрегация | 1 день |
| 7 | История + просмотр предыдущего результата | 0.5 дня |
| 8 | Тесты + деплой на ноутбуке | 0.5 дня |
| **Итого** | ~5.5 рабочих дней | |

## 12. Дальнейшее развитие

- Экспорт/импорт CSV/JSON.
- Графики прогресса (Chart.js).
- Синхронизация через облако (Postgres + Supabase).
- Роли и совместные тренировки.

---

**Статус**: все открытые вопросы закрыты. Принимаю комментарии и готов перейти к созданию репозитория и стартовому каркасу проекта.