ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ
(АКТ (ф) СПбГУТ)

КУРСОВОЙ ПРОЕКТ

НА ТЕМУ

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

«ФИТНЕС-ТРЕНЕР»

Л109. 25КП01. 001 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

Студент	ИСПП-21		08.12.2025	Е.А. Баранов
	(Группа)	(Подпись)	(Дата)	(И.О. Фамилия)
Преподаватель			09.12.2025	Ю.С. Маломан
		(Подпись)	(Дата)	(И.О. Фамилия)

Архангельск 2025

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ
(АКТ (ф) СПбГУТ)

УТВЕРЖДАЮ
Зав. отделением
_____ Ю.В. Солодкая
(Подпись) (И.О. Фамилия)
24 октября 2025

Задание
для курсового проектирования по МДК.02.01
Технология разработки программного обеспечения

студенту группы ИСПП-21 **,** 4 **курса**

Фамилия, имя, отчество Баранову Егору Алексеевичу

1 Тема курсового проекта
Разработка мобильного приложения «Фитнес-тренер»

2 Исходные данные к проекту
Разработать мобильное приложение, автоматизирующее контроль прохождения тренировок пользователя. Приложение должно обеспечивать возможность создания профиля пользователя, формирования и редактирования программ тренировок, фиксирования выполненных упражнений и просмотра истории занятий.

3 Содержание пояснительной записки
Введение
1 Анализ и разработка требований
2 Проектирование программного обеспечения
3 Разработка и интеграция модулей программного обеспечения
4 Тестирование и отладка программного обеспечения
5 Инструкция по эксплуатации программного обеспечения
Заключение
Список использованных источников

4 Перечень графического материала

5 Календарный график работы над проектом на весь период проектирования
24.10-31.10.2025 – анализ поставленной задачи; 01.11-07.11.2025 – проектирование ПО; 08.11-28.11.2025 – разработка и интеграция модулей ПО; 29.11-05.12.2025 – тестирование и отладка ПО; 24.10-07.12.2025 – написание и проверка программной документации, оформление пояснительной записки; 08.12.2025 – сдача курсового проекта на проверку; 09.12.2025 – защита курсового проекта

6 Срок сдачи студентом законченного курсового проекта 08 декабря 2025

7 Рекомендуемая литература, интернет источники

7.1 Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

7.2 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2025. – 400 с. – URL: https://znanium.ru/catalog/product/2178802. – Режим доступа: по подписке. – Текст : электронный.

7.3 Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ : ИНФРА-М, 2024. – 368 с. – URL: https://znanium.ru/catalog/product/2096940. – Режим доступа: по подписке. – Текст : электронный.

7.4 Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – URL: https://ibooks.ru/bookshelf/386796/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

7.5 Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. – Москва : КУРС : ИНФРА-М, 2024. – 336 с. – URL: https://znanium.ru/catalog/product/2083407. – Режим доступа: по подписке. – Текст : электронный.

Преподаватель и руководитель проектирования _____ Ю.С. Маломан
(Подпись) (И.О. Фамилия)

24 октября 2025

Задание принял к исполнению 24 октября 2025

Студент группы ИСПП-21, 4 курса, очной формы обучения _____
(Подпись)

Заключение по курсовому проекту

Преподаватель и руководитель проектирования _____ Ю.С. Маломан
(Подпись) (И.О. Фамилия)

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем курсовом проекте применяются следующие сокращения и обозначения:

БД – база данных

ЗОЖ – здоровый образ жизни

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

API – интерфейс проектирования интерфейса

ERD – диаграмма сущность-связь

HTTP – протокол передачи гипертекста

JSON – формат данных, основанный на JavaScript

JWT – интернет-стандарт для безопасной передачи информации в виде ключей-токенов в формате JSON

MVVM – архитектурный шаблон

ORM – объектно-реляционное отображение

PDF – универсальный формат электронных документов

SDK – набор инструментов для разработки программного обеспечения

SQL – язык структурированных запросов

ВВЕДЕНИЕ

Развитие мобильных технологий привело к тому, что смартфон стал универсальным инструментом, сопровождающим человека в повседневной деятельности, включая занятия спортом и контроль физической активности. На фоне роста интереса к здоровому образу жизни особое значение приобретают приложения, которые помогают пользователю планировать тренировки, выполнять упражнения корректно и отслеживать собственный прогресс. В условиях увеличивающегося числа пользователей, ориентированных на самостоятельные занятия, возрастает потребность в удобных цифровых инструментах, обеспечивающих доступность тренинга независимо от места и времени.

Создание современного фитнес-приложения требует не только технической реализации, но и продуманного пользовательского интерфейса, чтобы процесс взаимодействия был удобным, понятным и максимально приближённым к реальному тренировочному процессу. Многие существующие приложения страдают перегруженностью интерфейса, платными подписками, отсутствием гибкости или неудачной визуальной структурой, что снижает их практическую ценность. Поэтому разработка собственного мобильного приложения с оптимальной логикой работы и современным интерфейсом является актуальной задачей. Также важным аспектом является обеспечение стабильной работы приложения на различных устройствах, что увеличивает его доступность для широкого круга пользователей.

Целью курсового проекта является разработка мобильного приложения «Фитнес-тренер», предназначенного для сопровождения персональных тренировок. Приложение предоставляет пользователю возможность просматривать упражнения, выполнять тренировку по шагам, фиксировать подходы, контролировать время и менять структуру тренировки в удобном интерактивном формате.

Для достижения поставленной цели требуется решить следующие задачи:

- проанализировать предметную область;
- определить требования к разрабатываемому программному средству;
- спроектировать структуру БД для хранения информации необходимой для функционирования мобильного приложения;
- разработать интуитивно понятный пользовательский интерфейс;
- обеспечить удобство взаимодействия пользователя с приложением за счёт оптимизации навигации и логики интерфейса;
- реализовать серверную часть, обеспечивающую выполнение бизнес-логики приложения и разграничение прав доступа;
- провести тестирование разработанного программного продукта;
- выполнить адаптацию интерфейса под различные разрешения и версии Android;

Реализация указанных задач позволит создать удобный инструмент, который помогает организовать тренировочный процесс и делает выполнение упражнений более структурированным и наглядным.

1 Сбор и анализ требований

1.1 Назначение и область применения

Разрабатываемое мобильное приложение предназначено для автоматизации и упрощения процесса планирования и отслеживания тренировок, контроля физической активности и прогресса пользователя, что повысит эффективность занятий и мотивирует к достижению целей в фитнесе.

Областью применения является сфера фитнеса и ЗОЖ, например, тренажерные залы, фитнес-клубы и люди, занимающиеся индивидуально. Основными категориями пользователей будут являться люди, желающие повысить качество тренировок и питания.

1.2 Постановка задачи

Необходимо разработать мобильное приложение, предоставляющее доступ к следующей функциональности:

- авторизация и регистрация пользователей;
- просмотр, создание и редактирование плана тренировок;
- просмотр, создание и редактирование данных о блюдах;
- экспорт данных о пользователе в формате *.pdf;
- фильтрация и сортировка упражнений;
- добавление, редактирование, удаление цели для достижения;
- просмотр и изменение данных о пользователе.

В целях безопасности в приложении требуется проходить обязательную авторизацию перед использованием приложения. При открытии мобильное приложение отобразиться страница авторизации, после авторизации определиться роль пользователя.

Гость должен иметь доступ к просмотру списка упражнений и прохождению авторизации или регистрации.

Пользователь должен иметь доступ к просмотру, добавлению и удалению целей для достижения, просмотру профиля и его изменению, а также экспорт данных о профиле в PDF, просмотра блюд и упражнений.

Тренер должен иметь доступ к редактированию, добавлению и удалению упражнений, а также ко всем действиям, доступным авторизованному пользователю.

Администратор должен иметь доступ к админ-панели, в которой он имеет полный доступ ко всем функциям в системе.

На рисунке 1 представлена диаграмма вариантов использования приложения различными категориями пользователей.

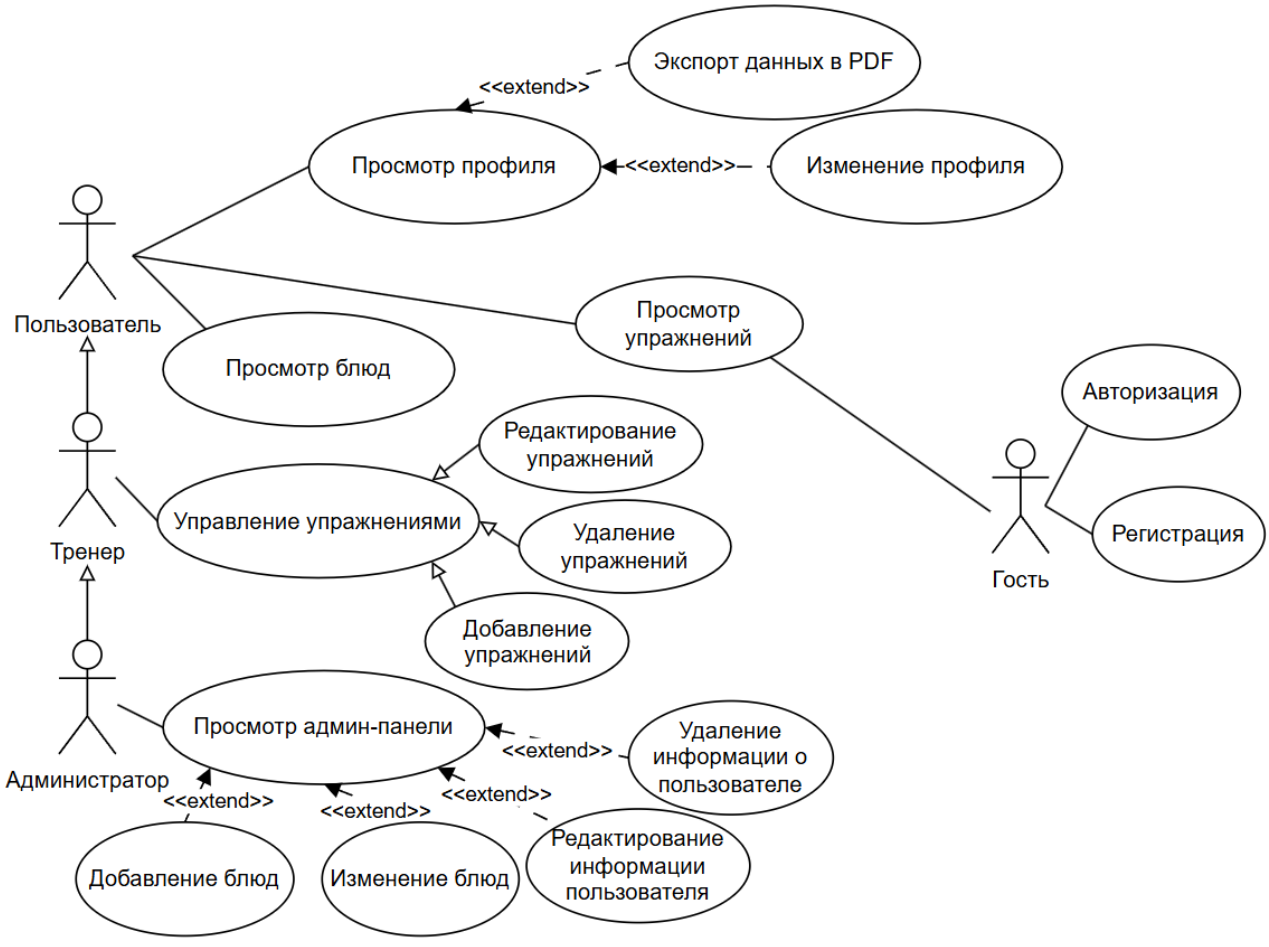

Рисунок 1 – Диаграмма вариантов использования

1.3 Выбор состава программных и технических средств

Приложение будет написано на языке программирования Kotlin с использованием архитектурного шаблона MVVM, что обеспечивает логическое разделение кода, удобство сопровождения и расширяемость функционала.

Для разработки будет использован Android Studio 2024.2.2 (Ladybug), так как среда обеспечивает удобную отладку, эмуляцию Android-устройств и поддержку актуальных версий SDK и Gradle.

Для реализации поставленной цели необходимо разработать БД, обеспечивающую хранение основной информации системы.

В качестве СУБД выбрана MySQL, так как она имеет высокую производительность, обладает кроссплатформенностью и легко масштабируется.

Для функционирования системы на стороне сервера необходимы следующие программные и технические средства:

- ОС Windows x86 64-бит или Linux x86 64-бит;
- MySQL Server не ниже 8.0;
- доступная оперативная память 3 ГБ;
- процессор с частотой не менее 1 ГГц и не менее 2 ядер;
- минимальный объем дискового пространства 10 ГБ.

Для функционирования приложения необходимы следующие программные и технические средства:

- ОС Android 11.0 (API 30) или выше;
- процессор с частотой не менее 1,8 ГГц;
- оперативная память не менее 2 ГБ;
- доступное место в памяти устройства – не менее 500 МБ;
- стабильное подключение к сети Интернет.

2 Проектирование ПО

2.1 Проектирование интерфейса пользователя

В рамках разработки ПО с использованием Figma спроектирован пользовательский интерфейс в виде wireframe, который демонстрирует структуру приложения, его основные элементы и доступную функциональность. На рисунке 2 в виде wireframe представлены следующие страницы ПО: главная страница, страница упражнений и страница подробностей об упражнении.

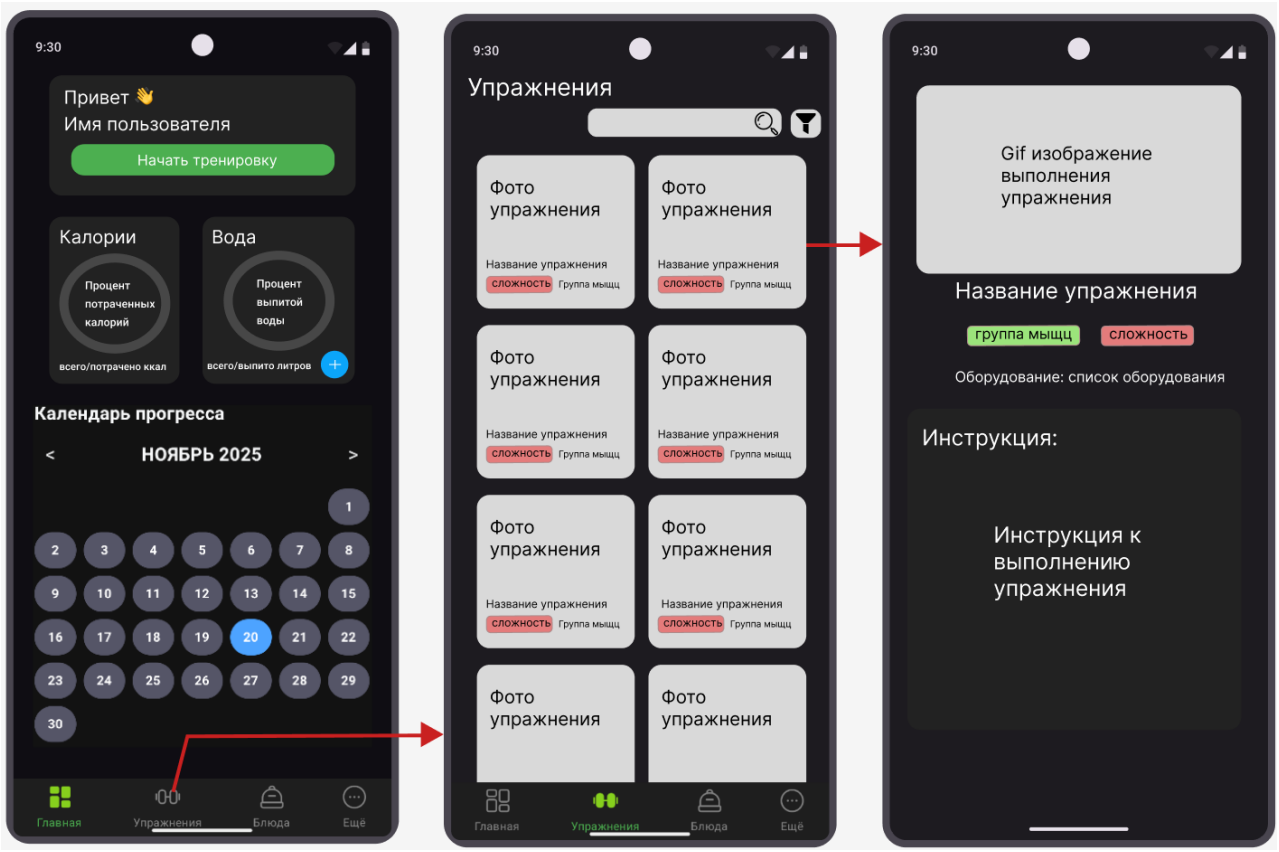

Рисунок 2 – Wireframe основных страниц

Для интерфейса мобильного приложения выбрана следующая цветовая палитра:

- #121212 – цвет фона
- #F8F9FA – вторичный цвет фона
- #222222 – цвет контейнеров
- #5C5C6C – вторичный цвет контейнеров
- #F8F9FA – цвет подписей кнопок и основного текста
- #212121 – вторичный цвет текста
- #81C784 – цвет кнопок
- #4CAF50 – вторичный цвет кнопок

2.2 Разработка архитектуры программного обеспечения

Архитектура ПО построена на основе клиент-серверной модели. Диаграмма развертывания компонентов представлена на рисунке 3.

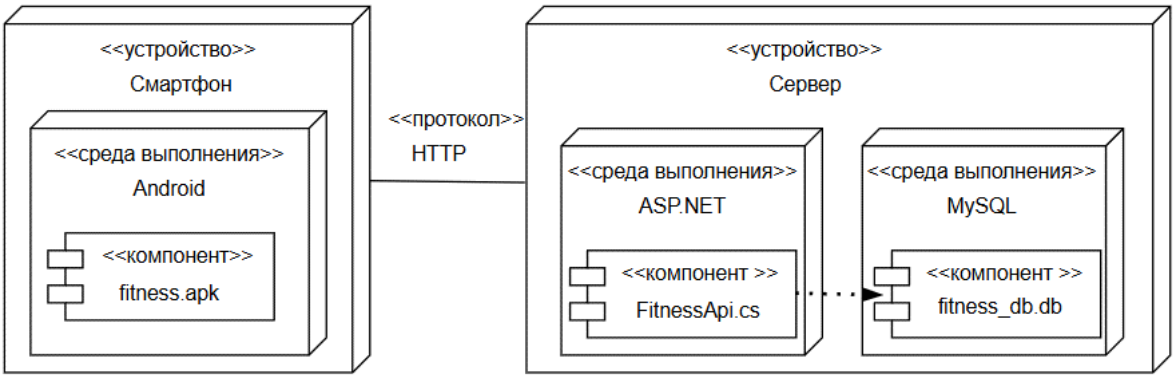

Рисунок 3 – Диаграмма развертывания компонентов

2.3 Проектирование БД

В рамках курсового проектирования требуется разработать БД для хранения данных о тренировках и питании пользователей [2]. На рисунке 4 в виде ERD показана физическая модель предметной области, созданная при помощи MySQL Workbench.

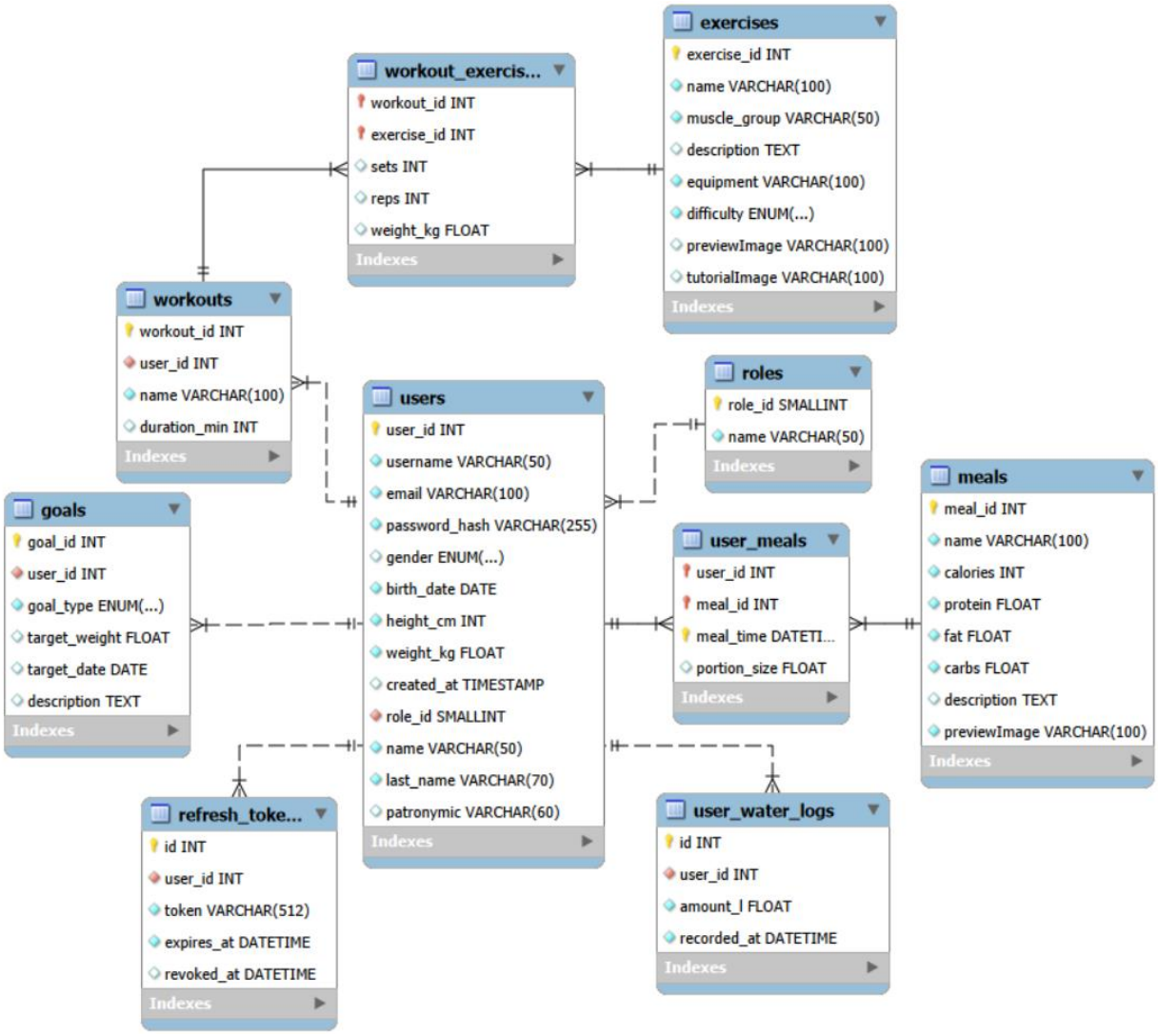

Рисунок 4 – Физическая модель данных

3 Разработка и интеграция модулей ПО

3.1 Разработка программных модулей

Серверная часть реализована в виде Web API на языке C# с использованием ASP.NET Core [2] [4]. В качестве ORM применяется Entity Framework Core, обеспечивающий взаимодействие с БД и автоматическое сопоставление сущностей с таблицами БД [5].

Взаимодействие мобильного клиента с сервером происходит через HTTP-запросы к RESTful API. Ответы от сервера приходят в формате JSON. Для выполнения сетевых запросов на клиенте используется библиотека Retrofit в связке с OkHttp и Gson для сериализации и десериализации данных.

На стороне Android-приложения реализован репозиторий WorkoutRepository, содержащий методы для получения и отправки данных на сервер. Код метода для отправки POST-запроса с данными о новой тренировке представлен листингом 1.

Листинг 1 – Код метода создания новой тренировки на стороне клиента

```
fun createNewWorkout(
    userId: Int, // ID пользователя
    name: String, // Название тренировки
    exercises: List<SelectedExercise>, // Список выбранных
упражнений
    duration: Int?, // Длительность тренировки
    onSuccess: (Int) -> Unit // отдаёт ID при успехе
) {
    // Запускаем корутину внутри ViewModel
    viewModelScope.launch {
                try {
                // Формируем объект тренировки для отправки на API
                val workout = Workout(
                     userId = userId,
                     name = name,
                     durationMin = duration
                 )
                // Отправляем запрос на создание тренировки
                val res = api.createWorkout(workout)
```

```
                // Получаем ID созданной тренировки
                val createdId = res.body()!!.workoutId
                // Сохраняем каждое упражнение, прикрепляя его к
тренировке
                exercises.forEach { sel ->
                    api.addExerciseToWorkout(
                        WorkoutExercise(
                          workoutId = createdId, // ID созданной
тренировки
                          exerciseId = sel.exercise.exerciseId,
// ID упражнения
                          sets = sel.sets, // Кол-во подходов
                          reps = sel.reps, // Кол-во повторов
                          weightKg = sel.weight // Вес
                          )
                        )
                    }
                    // Перезагружаем список тренировок для UI
                    loadWorkouts(userId)
                    // Уведомляем UI об успехе и отдаём ID
                    onSuccess(createdId)
                } catch (e: Exception) {
                // Ловим любые ошибки и обновляем состояние UI
                uiState.value = UiState.Error("Ошибка сохранения
тренировки")
                }
            }
        }
```

На сервере контроллер WorkoutsController принимает HTTP-запросы и сохраняет тренировку в БД, код контроллера представлен листингом 2.

Листинг 2 – Код метода создания тренировки на стороне сервера

```
[HttpPost]
public async Task<IActionResult> CreateWorkoutAsync([FromBody]
Workout workout){
    if (workout == null)
        return BadRequest("Workout is null");
    // Обнуляем ID на случай, если клиент его прислал
    workout.WorkoutId = 0;
    // Добавляем тренировку
    _context.Workouts.Add(workout);
    await _context.SaveChangesAsync();
    return Ok(workout); // Возвращаем объект с уже созданным ID
}
```

3.2 Реализация интерфейса пользователя

Интерфейс пользователя мобильного приложения разработан с использованием Jetpack Compose, который обеспечивает декларативный подход к созданию графических интерфейсов на языке Kotlin.

В приложении используется постраничная навигация между разделами «Главная», «Упражнения», «Блюда» и «Ещё». Переходы выполняются через Navigation Compose, а маршруты объединены в NavHost, что упрощает структуру приложения и передачу данных между экранами.

Каждый экран реализован как отдельная Composable-функция, содержащая элементы управления и визуальные компоненты, оформленные в едином стиле. Для управления состоянием используется архитектурный шаблон MVVM, где данные поступают из слоя ViewModel через StateFlow, обеспечивая реактивное обновление интерфейса при изменении данных.

Для отображения списка упражнений используется компонент ExerciseCard, показывающий название, сложность и группу мышц. Компонент используется на экране списка упражнений и отображается с помощью функции LazyVerticalGrid, обеспечивающей эффективную отрисовку большого количества элементов. Код реализации компонента ExerciseCard представлен листингом 3.

Листинг 3 – Код компонента для отображения карточки упражнений

```
@Composable
fun ExerciseCard(exercise: Exercise, onClick: () -> Unit) {
// Контейнер карточки
Box(
    modifier = Modifier
       // Скругляем углы
       .clip(RoundedCornerShape(20.dp))
       .clickable(
           indication = null,
            interactionSource = remember {
          MutableInteractionSource() }
    ) { onClick() }
```

```
        // Задаём фон
        .background(MatherialTheme.ColorScheme.Background)
        // Пропорции карточки (ширина/высота)
        .aspectRatio(0.9f)
        // Тень под карточкой
        .shadow(8.dp, RoundedCornerShape(20.dp))
    ) {
        // Картинка упражнения (фон)
        AsyncImage(
            model = exercise.previewImage,
            contentDescription = exercise.name,
            contentScale = ContentScale.Crop, // обрезаем, чтобы
заполнить всю карточку
            modifier = Modifier.fillMaxSize()
        )
        // Прозрачный градиент, чтобы текст был читаем
        Box(
            modifier = Modifier
                .fillMaxSize()
                .background(
                    Brush.verticalGradient(
                        listOf(
                   Color.Transparent,       // сверху прозрачный
                   Color.Black.copy(alpha = 0.75f)) // снизу
      затемнение
                    )
                )
        )
        // Блок текста и уровня сложности
        Column(
            modifier = Modifier
                .align(Alignment.BottomStart) // размещаем внизу
слева
                .padding(12.dp) // внутренний отступ
        ) {
            // Название упражнения
            Text(
                text = exercise.name,
                color = Color.White,
                fontWeight = FontWeight.Bold,
                fontSize = 18.sp,
                lineHeight = 20.sp
            )
            Spacer(Modifier.height(6.dp))
            // Тег сложности + группа мышц
            Row(verticalAlignment = Alignment.CenterVertically){
                // Компонент уровня сложности
                LevelTag(level = exercise.difficulty)
                Spacer(Modifier.width(8.dp))
                // Группа мышц
                Text(
                    text = exercise.muscleGroup,
                    color = Color.White.copy(alpha = 0.8f),
```

```
                    fontSize = 13.sp
                )
            }
        }
    }
}
```

3.3 Разграничение прав доступа пользователей

В приложении реализовано разграничение прав доступа с использованием JWT. После успешной авторизации сервер формирует токен, содержащий данные пользователя и его роль. JWT сохраняется в TokenManager и автоматически добавляется к каждому защищённому запросу в заголовке Authorization: Bearer <token>. Код метода авторизации, выполняющего получение и сохранение токена, представлен листингом 4. На сервере используется JWT Authentication middleware, которое проверяет подлинность токена и роль пользователя при каждом запросе.

Листинг 4 – Код метода авторизации

```
fun login(email: String, password: String) {
    viewModelScope.launch {
        try {
            // Отправляем запрос на сервер с email и паролем
            val response = api.login(LoginRequest(email,
password))
            if (response.isSuccessful) {
                // Сервер ответил успешно
                val body = response.body()
                if (body != null) {
                    // Сохраняем полученные токены для
дальнейшей работы приложения
                    tokenManager.saveTokens(body.accessToken,
body.refreshToken)
                    // Сообщаем, что вход выполнен успешно
                    _loginSuccess.postValue(true)
                } else {
                    // Сервер не прислал нужные данные
                    _error.postValue("Пустой ответ от сервера")
                }
            } else {
                // Сервер вернул ошибку авторизации
```

```
                _error.postValue(parseErrorMessage(response,
"Ошибка авторизации"))
            }
        } catch (e: NoInternetException) {
            // Нет соединения с интернетом
            uiState.value = UiState.Error("Нет соединения с
интернетом")
        } catch (e: Exception) {
            // Любая другая ошибка — например, сбой сервера
            uiState.value = UiState.Error("Ошибка сервера")
        }
    }
}
```

На экране упражнений размещена кнопка перехода к созданию тренировки. Она отображается только для пользователей, имеющих права тренера. Код реализации этой логики представлен листингом 5.

Листинг 5 – Код отображения кнопки создания тренировки

```
// Если пользователь — тренер/администратор, показываем кнопку
"+"
if (isTrainer) {
    Text(
        "+",
        fontSize = 32.sp,
        fontWeight = FontWeight.Bold,
        color = MaterialTheme.colorScheme.onBackground,
        modifier = Modifier
            // Обработка клика без стандартной анимации
            .clickable(
                indication = null,
                interactionSource = remember {
MutableInteractionSource() }
            ) {
                // Переход на экран создания упражнения
                navController.navigate("exerciseCreate")
            }
    )
}
```

При создании тренировки после заполнения всех полей и нажатия «Создать», в контроллере ExerciseController вызывается метод PostExercise для добавления нового упражнения (представлен листингом 6).

Листинг 6 – Код метода добавления упражнения

```
[HttpPost]
// Метод для добавления нового упражнения
public async Task<ActionResult<Exercise>>
PostExercise([FromBody] Exercise exercise)
{
    // Сохраняем упражнение в базе данных
    _context.Exercises.Add(exercise);
    await _context.SaveChangesAsync();
    // Возвращаем ответ с ссылкой на созданный ресурс
    return CreatedAtAction(nameof(GetExercise), new { id =
exercise.ExerciseId }, exercise);
}
```

3.4 Экспорт и импорт данных

В приложении реализован экспорт информации о пользователе в формате PDF. Код для формирования PDF файла представлен в листинге 6.

Листинг 6 – Код метода формирования PDF файла

```
fun generatePdfReport(
    user: UserProfileDto,
    workoutsVm: WorkoutViewModel,
    exerciseVm: ExerciseViewModel,
    month: YearMonth
) {
    // Создаём PDF-документ
    val pdf = PdfDocument()
    // Параметры страницы (A4)
    val pageInfo = PdfDocument.PageInfo.Builder(595, 842,
1).create()
    // Начало страницы
    val page = pdf.startPage(pageInfo)
    val canvas: Canvas = page.canvas
    // Заголовочный текст
    val titlePaint = Paint().apply {
        color = Color.BLACK
        textSize = 22f
        isFakeBoldText = true
    }
    // Обычный текст
    val textPaint = Paint().apply {
        color = Color.BLACK
```

```
            textSize = 14f
        }
        // Синяя линия-разделитель
        val blue = Paint().apply {
            color = Color.parseColor("#4DA3FF")
            strokeWidth = 3f
        }
        // Вертикальная позиция
        var y = 40f
        // Заголовок отчёта
        canvas.drawText("ФИТНЕС-ТРЕНЕР — ОТЧЁТ", 40f, y, titlePaint)
        y += 30f
        canvas.drawLine(40f, y, 555f, y, blue); y += 30f
        // Блок пользователя
        canvas.drawText("Пользователь", 40f, y, titlePaint)
        y += 24f
        canvas.drawText("Имя: ${user.username}", 40f, y, textPaint);
y += 18f
        canvas.drawText("Пол: ${if(user.gender=="male") "мужчина"
else "женщина"}", 40f, y, textPaint); y += 18f
        canvas.drawText("Дата рождения: ${user.birthDate}", 40f, y,
textPaint); y += 18f
        canvas.drawText("Рост: ${user.heightCm} см", 40f, y,
textPaint); y += 18f
        canvas.drawText("Вес: ${user.weightKg} кг", 40f, y,
textPaint); y += 30f
        canvas.drawLine(40f, y, 555f, y, blue); y += 30f
        // Период отчёта
        canvas.drawText("Период: ${month.formatRussian()}", 40f, y,
titlePaint); y += 30f
        // Достаём тренировки и выполнения
        val workouts = workoutsVm.workouts.value
        val completions = workoutsVm.completions.value
        // Список тренировок
        canvas.drawText("Тренировки", 40f, y, titlePaint); y += 24f
        workouts.forEach { w ->
            // Название тренировки
            canvas.drawText("• ${w.name}", 40f, y, textPaint)
            y += 18f
            // Длительность, если есть
            w.durationMin?.let {
                canvas.drawText("   Длительность: ${it} мин", 40f,
y, textPaint); y += 18f
            }
            // Даты выполнения
            val done = completions.filter { it.workoutId ==
w.workoutId }
            if (done.isNotEmpty()) {
                val dates = done.joinToString { it.completedAt }
                canvas.drawText("   Выполнена: $dates", 40f, y,
textPaint); y += 18f
                                ...
            } y += 14f
```

```
    }
    // Завершаем страницу
    pdf.finishPage(page)
    // Файл PDF в папке "Загрузки"
    val file = File(
Environment.getExternalStoragePublicDirectory(Environment.DIRECT
ORY_DOWNLOADS),
        "fitness_report_${month.year}_${month.monthValue}.pdf"
    )
    // Записываем PDF в файл
    pdf.writeTo(FileOutputStream(file))
    // Закрываем документ
    pdf.close()
}
```

4 Тестирование и отладка ПО

4.1 Структурное тестирование

В ходе курсового проектирования проведено структурное тестирование [1] метода addWater с помощью стандартной библиотеки JUnit. Код для тестирования метода представлен листингом 7.

Результат тестирования представлен рисунком 5.

Листинг 7 – Код тестирования класса добавления воды

```
@OptIn(ExperimentalCoroutinesApi::class)
class WaterViewModelTest {
    // Правило, подменяющее Dispatchers.Main тестовым
диспетчером.
    @get:Rule
    val mainDispatcherRule = MainDispatcherRule()
    // Заглушка API
    private lateinit var fakeApi: FakeWaterApi
    // Тестируемый ViewModel
    private lateinit var viewModel: WaterViewModelForTest
    @Before
    fun setup() {
        // По умолчанию API настроен на успешный ответ и
сегоднящнего значения выпитой воды на 500
        fakeApi = FakeWaterApi(shouldSucceed = true, todayValue
= 500)
        // Создаём ViewModel с фейковыми зависимостями и
тестовым диспетчером
        viewModel = WaterViewModelForTest(fakeApi,
mainDispatcherRule.dispatcher)
    }
    // ТЕСТ: успешная добавление воды должна сохранить токены
    @Test
    fun add_water_success_updates_dailyWater() = runTest {
        // Вызываем addWater внутри корутинного теста
        viewModel.addWater(1, 250)
        // Ждём выполнения всех корутин
        advanceUntilIdle()
        // Проверяем, что значение воды совпадает
        Assert.assertEquals(750, viewModel.dailyWater.value)
    }
}
```

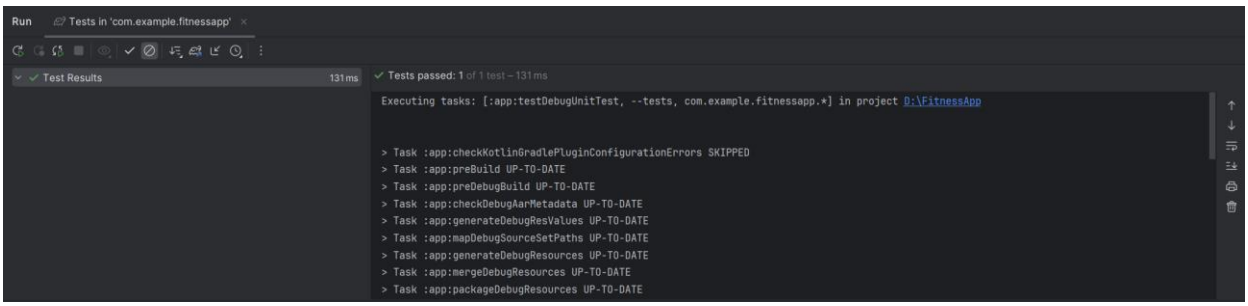

Рисунок 5 – Android Studio. Вид вкладки с результатами тестирования

4.2 Функциональное тестирование

Во время курсового проектирования проведено функциональное тестирование главного окна с помощью метода «чёрного ящика», результаты тестирования представлены в таблице 1.

Таблица 1 – Набор тестов

Действие	Ожидаемый результат	Фактический результат
Нажать на вчерашнюю дату, в которой не пройдена тренировка, в календаре прогресса	Появляется сообщение «Невозможно планировать в прошлом»	Совпадает с ожидаемым
Нажать на элемент вчерашней даты, в которой пройдена тренировка, в календаре прогресса	Появляется элемент, который показывает тренировку, пройденную в эту дату	Совпадает с ожидаемым
Нажать на текущую дату в календаре прогресса	Появляется элемент, который предлагает запланировать тренировку на эту дату	Совпадает с ожидаемым
Нажать на будущую дату в календаре прогресса	Появляется элемент, который предлагает запланировать тренировку на эту дату	Совпадает с ожидаемым
Нажать на кнопку навигации «Начать тренировку»	Переход на экран выбора тренировок	Совпадает с ожидаемым
Нажать на кнопку «+» на элементе кольца прогресса воды	Появляется элемент, который предлагает добавить определённое количество выпитой воды	Совпадает с ожидаемым
Нажать на кнопку навигации «Упражнения»	Переход на экран упражнений	Совпадает с ожидаемым

Продолжение таблицы 1

Действие	Ожидаемый результат	Фактический результат
Нажать на кнопку навигации «Блюда»	Переход на экран блюд	Совпадает с ожидаемым
Нажать на кнопку навигации «Ещё»	Переход на экран настроек	Совпадает с ожидаемым

По результатам тестирования можно сделать вывод, что созданное приложение работает корректно и согласно ожиданиям.

5 Инструкция по эксплуатации ПО

5.1 Установка программного обеспечения

Для функционирования системы на стороне сервера достаточны следующие программные и технические средства:

- ОС Windows x86 64-бит или Linux x86 64-бит;
- доступная оперативная память – 3 ГБ;
- процессор – 1,2 ГГц;
- объем дискового пространства – 10 ГБ.
- дополнительные компоненты: MySQL Server (8.0 и выше).

Для функционирования мобильного приложения достаточны следующие программные и технические средства:

- операционная система – Android 11.0 (API 30);
- процессор – 1,8 ГГц;
- оперативная память – 2 ГБ;
- доступное место в памяти устройства – 500 МБ;
- дополнительно: стабильное подключение к сети Интернет.

В качестве учётных данных используются следующие данные:

- электронная почта: profile@mail.com;
- пароль: immensePassWord12).

5.2 Инструкция по работе

После запуска мобильного приложения на экране появляется окно авторизации, в котором пользователю необходимо указать свои учетные данные (представлено рисунком 6). Приложение автоматически определяет роль вошедшего пользователя и отображает соответствующий набор возможностей, при этом внешний вид экранов остаётся единообразным и интуитивно понятным.

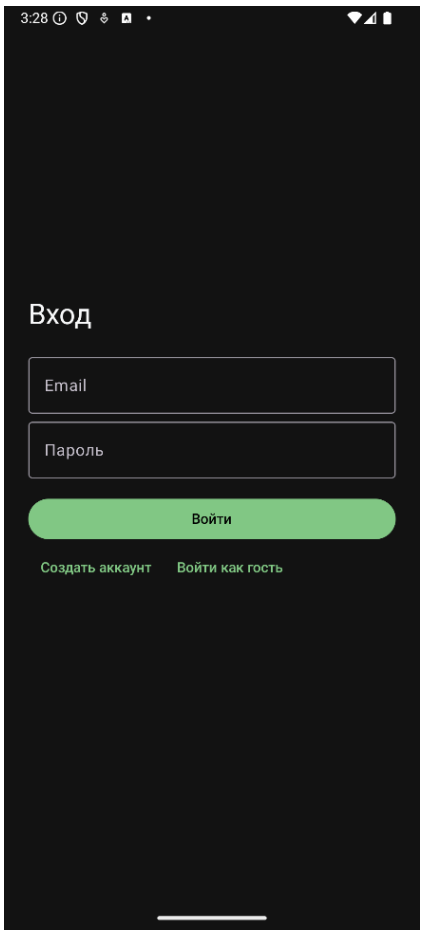

Рисунок 6 – Фитнес-тренер. Вид окна авторизации

В случае успешного входа открывается главный экран, содержащий основные элементы интерфейса и обеспечивающий доступ ко всем функциональным разделам системы, который представлен рисунком 7. Главный экран содержит навигационную панель, позволяющую переходить к разделам главного окна, упражнений, блюдам и настройкам. В центральной части экрана располагаются элементы, отображающие краткую информацию о текущем прогрессе: количество затраченных килокалорий, выпитого количества воды и динамика активности за последние дни.

Раздел упражнений представляет собой каталог, в котором каждая карточка содержит изображение, название, сложность и группу мышц, который показан на рисунке 8. Для ролей «Тренер» и «Администратор» в этом окне дополнительно отображается кнопка создания упражнения.

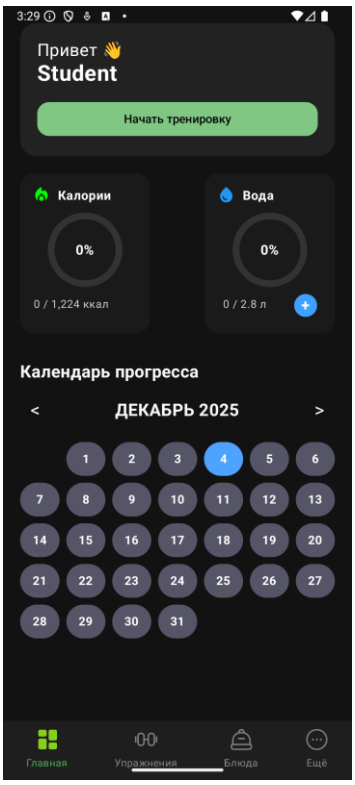

Рисунок 7 – Фитнес-тренер. Вид главного окна

Рисунок 8 – Фитнес-тренер. Вид окна упражнения

При выборе карточки упражнения открывается окно подробного просмотра, где представлены техника выполнения, описание целевой группы мышц и инструкция к выполнению, который представлен рисунком 9. Для ролей «Тренер» и «Администратор» в этом окне дополнительно отображаются кнопки редактирования и удаления упражнения.

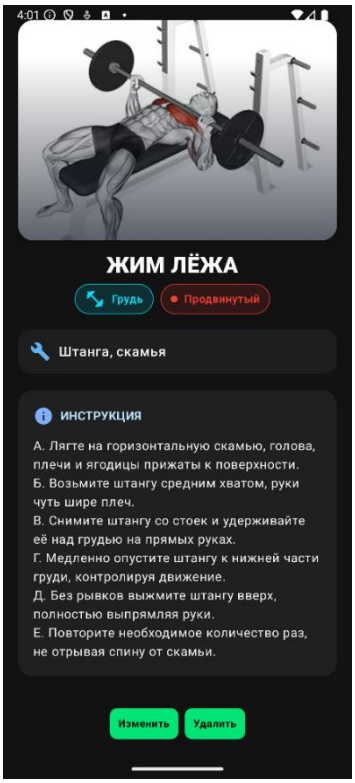

Рисунок 9 – Фитнес-тренер. Вид окна подробностей об упражнении

Работа с тренировками осуществляется через нажатие в главном окне кнопки «Начать тренировку», после чего открывается окно просмотра упражнений, которое представлено рисунком 10. Пользователь может открыть любую из карточек, просмотреть состав тренировки, выполнить, изменить тренировку или создать новую. Процесс создания тренировки включает выбор упражнений из каталога и настройку параметров подходов, времени и количества повторений, который представлен рисунком 11.

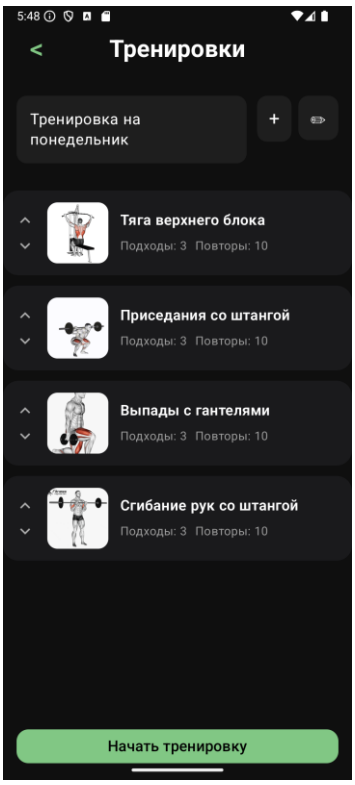

Рисунок 10 – Фитнес-тренер. Вид окна выбора тренировки

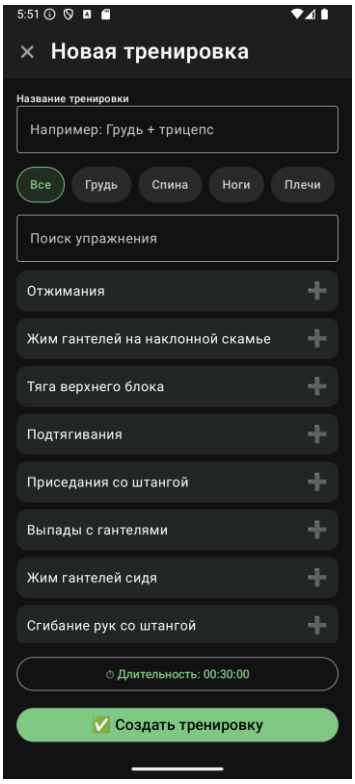

Рисунок 11 – Фитнес-тренер. Вид окна создания тренировки

ЗАКЛЮЧЕНИЕ

Целью курсового проектирования являлась разработка мобильного приложения «Фитнес-тренер», предназначенного для сопровождения персональных тренировок и контроля выполнения упражнений.

Разработанное приложение соответствует актуальным требованиям к современным мобильным сервисам и обеспечивает пользователя удобными средствами для ведения и контроля тренировки. Реализованная функциональность способствует упорядочению тренировочного процесса, повышению его эффективности и наглядности, а также повышает удобство взаимодействия пользователя с приложением, обеспечивая более осознанный подход к выполнению физических упражнений.

Цель курсового проектирования достигнута, в процессе её достижения решены следующие задачи:

- проанализирована предметная область;
- определены требования к разрабатываемому программному средству;
- спроектирована структура БД для хранения информации необходимой для функционирования мобильного приложения;
- разработан интуитивно понятный пользовательский интерфейс;
- обеспечено удобство взаимодействия пользователя с приложением за счёт оптимизации навигации и логики интерфейса;
- реализована серверная часть, обеспечивающую выполнение бизнес-логики приложения и разграничение прав доступа;
- проведено тестирование разработанного программного продукта;
- выполнена адаптация интерфейса под различные разрешения и версии Android;

В ходе работы над проектом создан функциональный и удобный в использовании программный продукт, отвечающий актуальным требованиям к мобильным приложениям спортивной направленности и обеспечивающий пользователю доступ ко всем основным элементам тренировочного процесса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading (дата обращения: 27.11.2025). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

2. Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2025. – 400 с. – URL: https://znanium.ru/catalog/product/2178802 (дата обращения: 10.11.2025). – Режим доступа: по подписке. – Текст : электронный.

3. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ : ИНФРА-М, 2024. – 368 с. – URL: https://znanium.ru/catalog/product/2096940 (дата обращения: 3.11.2025). – Режим доступа: по подписке. – Текст : электронный.

4. Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – URL: https://ibooks.ru/bookshelf/386796/reading (дата обращения: 10.11.2025). – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

5. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. — Москва : КУРС : ИНФРА-М, 2024. — 336 с. – URL: https://znanium.ru/catalog/product/2083407 (дата обращения: 15.11.2025). – Режим доступа: по подписке. – Текст : электронный.