

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное фитнес-приложение для тренировок дома

Студент

Николин Роман Валерьевич

(подпись, дата)

(ФИО полностью)

Руководитель

Балашова И.Ю.

(подпись, дата)

(фамилия, инициалы)

Нормоконтролёр

Такташкин Д.В.

(подпись, дата)

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

Макарычев П.П.

(подпись)

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

Попова Н.А.

(подпись)

(фамилия, инициалы)

Пенза, 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«__»_____2022 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Николин Роман Валерьевич гр. 18ВП2 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное фитнес-приложение для тренировок дома

Тема утверждена приказом ПГУ № 327/О от “18” апреля 2022

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Балашова И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать приложение для мобильных устройств, предоставляющее
возможность для занятий спортивными тренировками дома.

2) Среда выполнения приложения: операционная система Android 6 и выше

Среда выполнения сервера: операционная система Windows 10

Языковые средства разработки: C#, UML

Инструментальные средства разработки: Microsoft Visual Studio 2019,
Microsoft Project, ArgoUML, Unity3D, Firebase

Технологии разработки: RUP, ООП

3) Основные функции мобильного приложения:

а) предоставление возможности входа и регистрации пользователей в системе;

б) демонстрация упражнений для тренировки с их подробным описанием;

в) демонстрация корректного выполнения упражнений в дополненной реальности.

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области постановка задачи	15%	13.04.2022– 21.04.2022	
2	Проектирование	25%	22.04.2022– 05.05.2022	
3	Реализация	40%	06.05.2022– 30.05.2022	
4	Контроль качества программного обеспечения	5%	31.05.2022– 02.06.2022	
5	Оформление пояснительной записки	15%	03.06.2022– 17.06.2022	

Дата выдачи задания "22" декабря 2022

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____

(подпись, дата)

Реферат

Пояснительная записка содержит 76 листов, 23 рисунка, 16 таблиц, 28 использованных источников, 3 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ANDROID, API, C#, RUP, UML

Объектом разработки является мобильное приложение для занятий спортивными тренировками дома.

Цель работы – разработка мобильного приложения для занятий спортивными тренировками дома, предоставляющее готовые комплексы тренировок, а также демонстрирующего правильность выполнения упражнений в дополненной реальности.

Разработка приложения производилась на языке программирования C#. В качестве среды разработки использовалась среда программирования Microsoft Visual Studio, операционная система Windows 10. Приложение предназначено для использования под управлением ОС Android версии 6.0 и выше.

					ПГУ 09.03.04 – 08BP182.15 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Мобильное фитнес-приложение для тренировок дома. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Николин Р.В.						
Пров.		Балашова И.Ю.					3	76
Реценз.						Группа 18ВП2		
Н. контр.		Такташкин Д.В.						
Утверд.		Макарычев П.П.						

Содержание

Введение	6
1 Анализ предметной области и требований приложения.....	8
1.1 Анализ предметной области и постановка задачи	8
1.2 Анализ программных продуктов аналогичного назначения.....	8
1.2.1 Мобильное приложение для тренировок Health.....	9
1.2.2 Мобильное приложение для тренировок Google Fit.....	10
1.2.3 Мобильное приложение для тренировок 7 Minute Workout	12
1.2.4 Сравнение аналогов.....	13
1.3 Анализ целевой аудитории	17
1.4 Анализ функциональных требований	19
1.5 Анализ нефункциональных требований	22
1.6 Планирование разработки и оценка бюджета	23
2 Проектирование мобильного приложения.....	28
2.1 Выбор технологий и инструментальных средств разработки	28
2.2 Проектирование архитектуры программного обеспечения	29
2.3 Проектирование интерфейса пользователя	30
3 Реализация мобильного приложения	37
3.1 Реализация клиентской части приложения.....	37
3.2 Реализация серверной части приложения.....	38
4 Тестирование программного продукта	39
4.1 Тестирование черным ящиком	39
4.1.1 Классы эквивалентности	39
4.1.2 Граничные условия.....	40
4.1.3 Тест-кейсы	40
4.2 Функциональное тестирование белым ящиком.....	41
4.2.1 Алгоритм мобильного приложения.....	42
4.3 Метрики кода.....	43
4.3.1 Метрики размера программы.....	44
4.3.2 Циклом атическая сложность	45
4.3.3 Метрики Холстеда.....	46

	5
Закключение	48
Список использованных источников	49
Приложение А. Глоссарий.....	52
Приложение Б. Исходный код мобильного приложения	54
Приложение В. Результаты тестирования мобильного приложения	70

Введение

Мобильное приложение – это тип приложения, предназначенный для работы на мобильном устройстве, которым может быть смартфон или планшет. Даже если приложения обычно представляют собой небольшие программные единицы с ограниченными функциями, им все же удается предоставлять пользователям качественные услуги и возможности.

В отличие от приложений, разработанных для настольных компьютеров, мобильные приложения отходят от интегрированных программных систем. Вместо этого каждое мобильное приложение предоставляет изолированную и ограниченную функциональность. Например, это может быть игра, калькулятор или мобильный веб-браузер.

Из-за ограниченных аппаратных ресурсов ранних мобильных устройств мобильные приложения избегали многофункциональности. Однако даже если используемые сегодня устройства намного мощнее, мобильные приложения всё ещё остаются узко функциональными. Именно так владельцы мобильных приложений позволяют потребителям выбирать именно те функции, которые должны иметь их устройства.

Мобильные приложения охватывают уже практически все сферы человеческой жизнедеятельности, и фитнес индустрия не стала исключением.

Фитнес стал важнейшей частью жизни современного человека. С каждым днем все больше и больше людей занимаются йогой, кроссфитом, пауэрлифтингом, бегом, начинают следить за своим дневным рационом и сейчас такой образ жизни становится неотъемлемой частью практически каждого человека.

Мобильное фитнес-приложение для тренировок позволяет правильно составить программу упражнений, контролирует изменения веса и форм, показывает, как необходимо правильно выполнять упражнения. Это своеобразный тренер, которого пользователь может использовать как дома, так и в любом другом удобном месте. Разработка подобного приложения – это возможность дать своим клиентам правильный комплекс и альтернативу частному инструктажу.

Таким образом, на основании приведенных выше факторов можно сделать вывод о том, что разработка мобильного фитнес-приложения весьма актуальна и перспективна в условиях современного общества.

1 Анализ предметной области и требований приложения

1.1 Анализ предметной области и постановка задачи

Мобильное фитнес-приложение для тренировок позволяет правильно составить индивидуальную программу спортивных упражнений, подстраиваемую по каждого пользователя. Благодаря руководству по правильному выполнению упражнений пользователь сможет достичь поставленных целей.

Мобильное приложение является аналогом походов в спортзал. Пользователь может использовать данное приложение как дома, так и в любом другом подходящем месте.

Разработка данного мобильного приложения позволит пользователям выполнять комплекс спортивных упражнений, что является хорошим аналогом частного инструктора.

Таким образом, в приложении должны быть представлены следующие основные функции:

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- настраиваемый таймер;
- просмотр статистики выполненных упражнений;
- контроль изменения веса пользователя;
- демонстрация правильного выполнения упражнений в дополненной реальности.

1.2 Анализ программных продуктов аналогичного назначения

На данный момент существует целый ряд приложений, предоставляющих возможности для занятий спортивными тренировками дома [1]. Рассмотрим наиболее популярные приложения такого рода и произведем для них сравнительный анализ.

1.2.1 Мобильное приложение для тренировок Health

Мобильное фитнес-приложение Health от Apple. Компания-разработчик предоставляет бесплатный продукт, который устанавливается на телефоны с операционной системой IOS [2].

Данное приложение хранит информацию о здоровье, включая показатели активности пользователя в течение времени, статистику сна и другие важные показатели.

Информация в приложении хранится в зашифрованном виде для обеспечения конфиденциальности персональных данных пользователя.

Благодаря взаимодействию аппаратных продуктов от компании Apple, присутствует возможность просмотра статистики на всех устройствах, объединенных в одну систему.

Главная страница мобильного приложения Health представлена на рисунке 1.

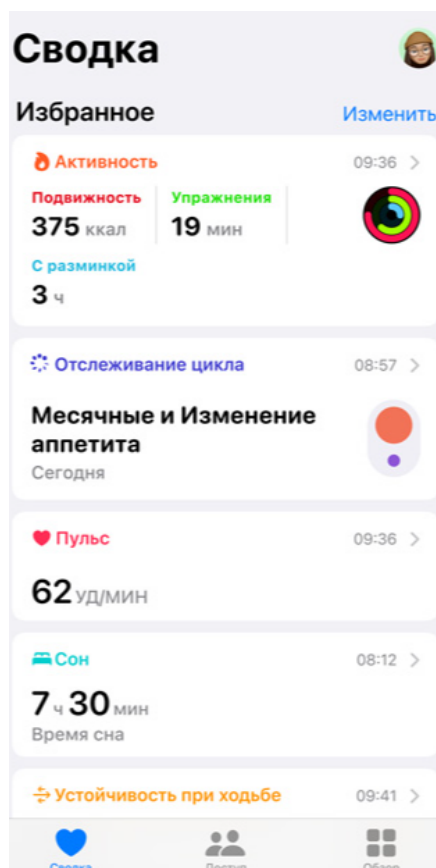


Рисунок 1 – Главная страница мобильного приложения Health

Пользователи данного приложения выделяют следующие преимущества:

- ведение статистики физической активности;
- алгоритмы создания индивидуальной программы тренировок.

Помимо видимых преимуществ, в приложении присутствуют следующие недостатки:

- доступно только на устройствах с операционной системой IOS;
- отсутствие инструкций по правильному выполнению упражнений.

1.2.2 Мобильное приложение для тренировок Google Fit

Приложение Google Fit от компании Google было разработано на основе исследований и рекомендаций Всемирной организации здравоохранения и Американской кардиологической ассоциации.

Данное мобильное приложение предоставляет пользователю возможность настройки индивидуальных тренировок и практических советов, основанных на здоровье человека. В приложении Google Fit используется балльно-рейтинговая система.

В приложении присутствует функция по отслеживанию прогресса и активности пользователя с телефона или со смарт-часов на операционной системе Android, а также присутствует возможность добавления целей активности, подстраиваемых под каждого пользователя индивидуально. Приложение Google Fit предоставляет пользователям советы, основанные на рекомендациях Американской кардиологической ассоциации и Всемирной организации здравоохранения, которые предоставляют показатели для других видов деятельности, помимо ходьбы, для побуждения пользователей к занятиям, которые будут повышать частоту сердечных сокращений, без необходимости посещения тренажерного зала.

Google Fit предоставляет единый набор интерфейсов для приложений и производителей устройств для хранения и доступа к данным об активности из фитнес-приложений и датчиков на Android и других устройствах.

Пользователи могут выбрать, кому будут передаваться их данные о фитнесе, а также удалить эту информацию в любое время.

Главный экран приложения представлен на рисунке 2.



Рисунок 2 – Главный экран мобильного приложения Google Fit

Пользователи данного приложения выделяют следующие преимущества:

- простой и удобный интерфейс пользователя;
- сбор и анализ большого объема данных по тренировкам пользователя;
- демонстрация правильного выполнения упражнений;
- возможность взаимодействия с остальными сервисами Google.

Помимо видимых преимуществ, в приложении присутствуют следующие недостатки:

- для работы требуется подключение к сети интернет;
- быстрый расход заряда батареи мобильного устройства;
- негативные отзывы, связанные с нежеланием пользователей следовать балльно-рейтинговой системе.

1.2.3 Мобильное приложение для тренировок 7 Minute Workout

Главная цель мобильного приложения 7 Minute Workout заключается в демонстрации пользователю инструкций о выполнении 12 базовых упражнений, которые длится в течение семи минут.

Согласно исследованиям, проведенным компанией-разработчиков: семь минут тренировок в день являются минимальной эффективной физической нагрузкой для человека.

Данное приложение предоставляет на выбор 72 упражнения и 22 дополнительных тренировки, которые можно настраивать под себя, благодаря чему возможно создать около 1000 вариантов тренировок.

В приложении присутствует функция Smart Workout, которая оценивает физическую форму и уровень мотивации пользователя и рекомендует различные программы тренировок, наряду с увеличением их интенсивности.

Главная страница мобильного приложения представлена на рисунке 3.

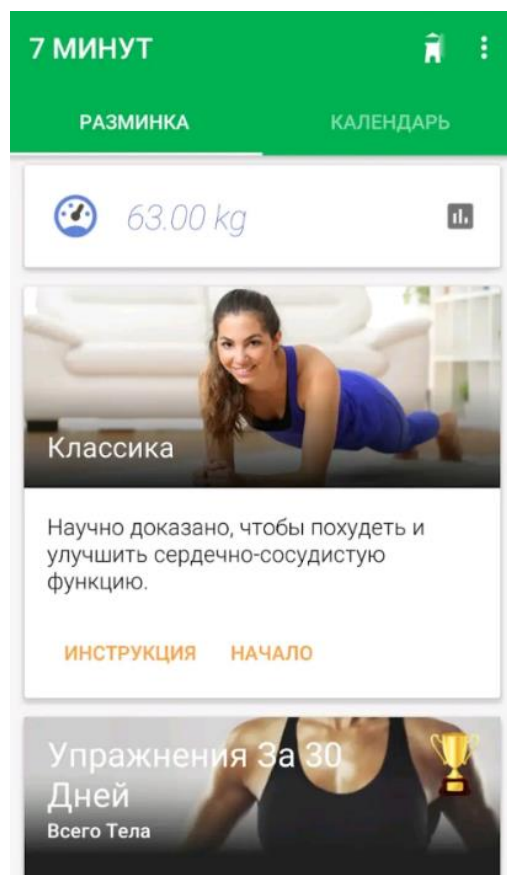


Рисунок 3 – Главная страница мобильного приложения 7 Minute Workout

Пользователи данного приложения выделяют следующие преимущества:

- простой и удобный интерфейс пользователя;
- демонстрация правильного выполнения упражнений;
- быстрые тренировки.

Помимо видимых преимуществ, в приложении присутствуют следующие недостатки:

- ограниченное количество упражнений;
- доступно только на устройствах с операционной системой IOS.

1.2.4 Сравнение аналогов

Рассмотрев аналогий, можно выделить основные функции, которые должны присутствовать у фитнес-приложения для занятия дома:

- простой и удобный интерфейс пользователя;
- демонстрация правильного выполнения упражнений;
- многофункциональный настраиваемый таймер;
- анализ результатов тренировок пользователя;
- ненавязчивые уведомления для напоминания пользователю про выполнение тренировок.

Создание удобного интерфейса пользователя предполагает обеспечение простого, приятного и интуитивно-понятного взаимодействия пользователя с приложением.

Демонстрация правильного выполнения упражнений является основой для получения пользователем качественного, не опасного для здоровья опыта выполнения различного вида тренировок.

Многофункциональный настраиваемый таймер поможет разнообразить тренировки. Благодаря таймеру каждый пользователь сможет настроить программу тренировок индивидуально под себя.

Ненавязчивые уведомления для напоминания пользователю про выполнение тренировок послужат дополнительной мотивацией пользователя к выполнению поставленных упражнений.

Среди критериев, по которым была проведена сравнительная характеристика аналогов, были выделены следующие:

- удобство и простота интерфейса;
- возможность демонстрации выполнения упражнений;
- наличие настраиваемого таймера;
- анализ результатов тренировок.

Каждый из критериев является обязательным для обеспечения положительного опыта использования фитнес-приложения. Рассмотрим подробно каждый из приведенных выше критериев.

Удобство интерфейса показывает, насколько эффективно конечный пользователь будет пользоваться основными функциями, предоставляемыми приложением, и насколько быстро пользователь будет ориентироваться в программе. Оценка критерия от 0 до 5.

Расшифровка оценок критерия представлена в таблице 1.

Таблица 1 – Расшифровка оценок критерия «Удобство и простота интерфейса»

Оценка	Описание
0	Интерфейс не реализован.
1	Интерфейс максимально неудобен для использования, его логические элементы очень трудно воспринимать, и они расположены в порядке, максимально затрудняющем взаимодействие с ними.
2	Удобство интерфейса на низком уровне, имеются небольшие затруднения при взаимодействии с логическими элементами их трудно воспринимать.
3	Удобство интерфейса на среднем уровне, элементы взаимодействия расположены в логичной связке, при этом общее понимание полной функциональности приложения затруднено. Интуитивно понятный пользовательский интерфейс.

Продолжение таблицы 1

Оценка	Описание
4	Интерфейс удобен для использования. Логические связи не нарушены, весь функционал доступен пользователю, при этом незначительные недостатки утомляют при долгом взаимодействии с приложением.
5	Интерфейс удобен для использования без каких-либо недостатков.

Возможность демонстрации выполнения упражнений показывает наличие данной возможности в приложении. Оценка критерия от 0 до 1. Расшифровка оценок критерия приведена в таблице 2.

Таблица 2 – Расшифровка оценок критерия «Возможность демонстрации выполнения упражнений»

Оценка	Описание
0	Функция не реализована
1	Функция реализована

Характеристика по наличию настраиваемого таймера показывает степень реализации данной функции в приложении. Оценка критерия от 0 до 2. Расшифровка оценок критерия приведена в таблице 3.

Таблица 3 – Расшифровка оценок критерия «Наличие настраиваемого таймера»

Оценка	Описание
0	Функция не реализована
1	Функция реализована частично. Таймер присутствует, но не настраивается
2	Функция реализована полностью

Характеристика анализа тренировок показывает степень реализации данной функции в приложении. Оценка критерия от 0 до 2. Расшифровка оценок критерия приведена в таблице 4.

Таблица 4 – Расшифровка оценок критерия «Анализ результатов тренировок»

Оценка	Описание
0	Функция не реализована
1	Функция реализована частично. Анализ результатов тренировок производится и сохраняется локально
2	Функция реализована полностью. Анализ результатов тренировок производится. Данные синхронизируются с другими устройствами.

Результаты сравнения программ-аналогов по представленным выше характеристикам приведены в таблице 5.

Таблица 5 – Сравнительная характеристика продуктов-аналогов

Критерий	Health	Google Fit	7 Minute Workout
Удобство и простота интерфейса	5	3	2
Возможность демонстрации выполнения упражнений	0	1	1
Наличие настраиваемого таймера	0	1	2
Анализ результатов тренировок	1	2	0
Оценка приложения по среднему арифметическому	1.5	1,75	1,25

В результате сравнения программ-аналогов можно сказать, что мобильное приложение Google Fit является наиболее удобным для пользователей, при этом ни одна из программ не содержит в себе полный функционал, который необходим фитнес-приложению. По итоговой оценке, можно сделать выводы, что приложения Health от Apple и Google Fit от Google больше подходят для занятия тренировками дома.

Разрабатываемое приложение должно обеспечить пользователю интуитивно понятный, простой пользовательский интерфейс, доступ к анализу результатов тренировок, демонстрации правильного выполнения упражнений. Благодаря вышеперечисленному набору функций пользователь получит качественную здоровую тренировку дома.

1.3 Анализ целевой аудитории

Целевой аудиторией проекта являются люди, желающие поддерживать здоровый образ жизни и иметь спортивное телосложение, используя для этого лишь мобильное устройство. В частности, подобный вывод можно сделать на основании статистики изменения числа пользователей у 30 самых популярных мобильных приложений данной тематики. График, на котором показана динамика изменения числа пользователей с соотношением их возраста за последний год, представлен на рисунке 4.

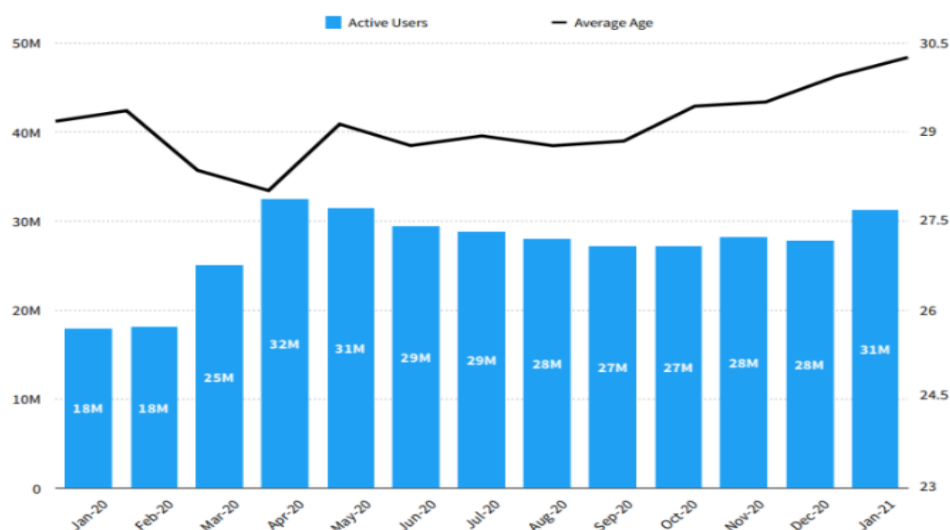


Рисунок 4 – Изменение числа пользователей у самых популярных мобильных фитнес-приложений

Как можно увидеть, рынок мобильных фитнес-приложений постоянно растет. Кроме того, среди пользователей мобильных приложений женщины занимают больший процент, чем мужчины. Среди всех видов фитнес-приложений, наиболее популярным является тот, который предоставляет возможность занятий тренировками и другими физическими активностями. График популярности различных видов фитнес-приложений среди женщин представлен на рисунке 5.

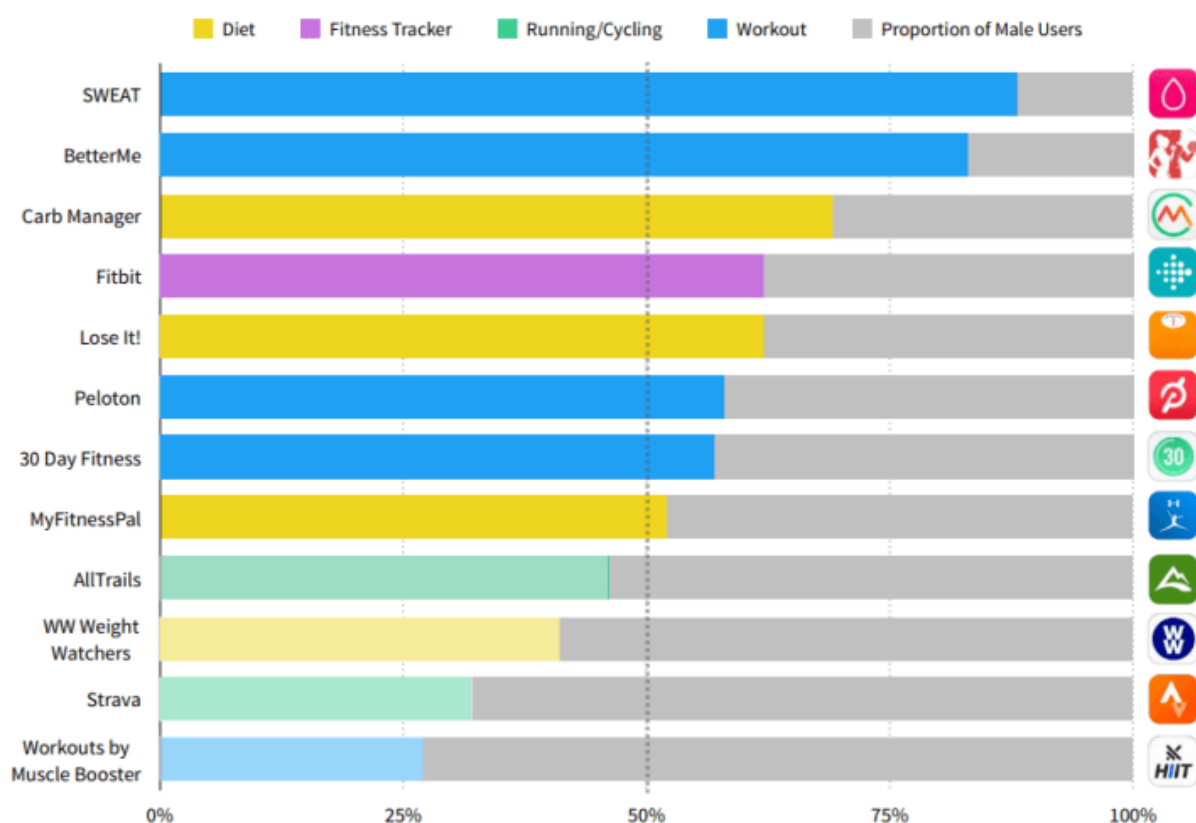


Рисунок 5 – Доля популярности различных видов фитнес-приложений среди женщин

Можно заметить, что в списке наиболее популярных приложений, лидирующие места занимают написанные под телефоны с операционной системой Android [3]. Поэтому приложение будет разрабатываться таким образом, чтобы оно успешно функционировало для Android 6.0 и более новых версий системы.

1.4 Анализ функциональных требований

В мобильном фитнес-приложении должны быть представлены следующие основные функции:

- регистрация новых пользователей в системе;
- вход зарегистрированных пользователей в систему;
- просмотр информации о здоровье;
- тренировка по готовым наборам упражнений;
- просмотр информации о тренировке;
- ввод данных для анализа состояния здоровья;
- демонстрация правильного выполнения упражнений в дополненной реальности;
- предоставление пользователю возможности тренироваться по таймеру.

В результате анализа требований задания была создана диаграмма вариантов использования мобильного фитнес приложения (рисунок 6).



Рисунок 6 – Диаграмма вариантов использования мобильного приложения

Для некоторых из представленных вариантов использования были составлены спецификации, позволяющие более подробно рассмотреть каждую функцию отдельно.

Составим спецификации вариантов использования «Зарегистрироваться в системе» (рисунок 7).

Прецедент: Зарегистрироваться в системе
ID: 1
Краткое описание: Новый пользователь регистрируется в системе
Главные актеры: Пользователь
Предусловие: Пользователь запустил приложение
Основной поток: 1. В главном меню игрового приложения открывается окно авторизации и регистрации 2. Пользователь нажимает кнопку для регистрации 3. Пользователь вводит данные для регистрации
Постусловие: Пользователь зарегистрирован

Рисунок 7 – Спецификация варианта использования «Зарегистрироваться в системе»

Составим спецификации вариантов использования «Авторизоваться в системе» (рисунок 8).

Прецедент: Авторизоваться в системе
ID: 2
Краткое описание: Пользователь авторизируется в системе
Главные актеры: Пользователь
Предусловие: Пользователь запустил приложение
Основной поток: 1. В главном меню игрового приложения открывается окно авторизации и регистрации 2. Пользователь нажимает кнопку для авторизации 3. Пользователь вводит данные для авторизации
Постусловие: Пользователь авторизован

Рисунок 8 – Спецификация варианта использования «Авторизоваться в системе»

Составим спецификации вариантов использования «Ввести данные для анализа состояния здоровья» (рисунок 9).

Прецедент: Ввести данные для анализа состояния здоровья
ID: 3
Краткое описание: Пользователь вводит данные для анализа состояния здоровья
Главные актеры: Пользователь
Предусловие: Пользователь запустил приложение и авторизовался в системе
Основной поток: 1. В окне ввода информации пользователь вводит свои данные 2. Пользователь нажимает на кнопку для продолжения
Постусловие: Данные пользователя сохранены

Рисунок 9 – Спецификация варианта использования «Ввести данные для анализа состояния здоровья»

Более подробное описание функционирования прецедентов можно представить в виде диаграммы деятельности. На рисунке 10 представлена диаграмма деятельности для варианта использования «Зарегистрироваться в системе».

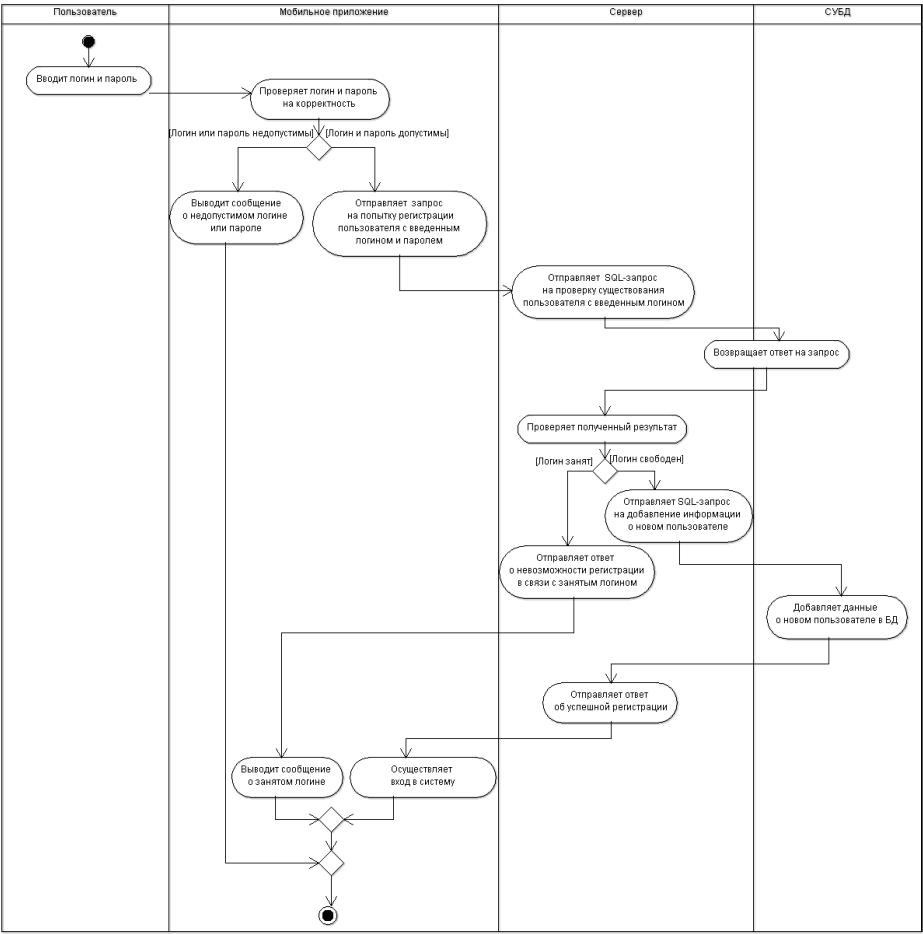


Рисунок 10 – Диаграмма деятельности варианта использования «Зарегистрироваться в системе»

Для варианта использования «Ввести данные для анализа состояния здоровья» была составлена диаграмма последовательности, представленная на рисунке 11.

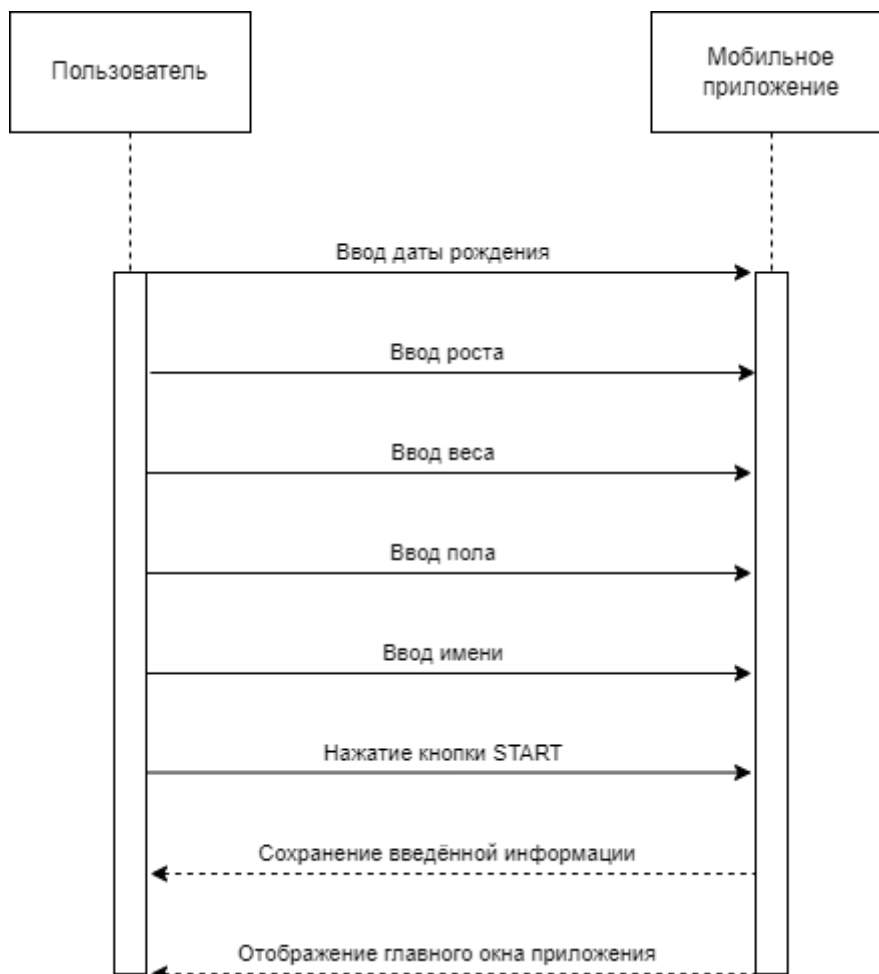


Рисунок 11 – Диаграмма последовательности варианта использования «Ввести данные для анализа состояния здоровья»

В результате анализа функциональных требований были успешно выявлены основные функции разрабатываемого приложения. При этом для разрабатываемого приложения были построены диаграммы вариантов использования, деятельности и последовательности, а также для некоторых выявленных вариантов использования были составлены спецификации.

1.5 Анализ нефункциональных требований

Разработка серверной части приложения должна быть выполнена на языке программирования C#, в среде программирования Microsoft Visual Studio 2019. Данные пользователя сохраняются на телефоне. Данные для

авторизации сохраняются в облачный сервер, предоставляемый средствами Firebase [4]. Визуальная часть создается с помощью многофункционального графического редактора Adobe Photoshop. приложение должно быть отлажено и должно успешно работать под управлением Android 5 и выше. Для проектирования бюджета будет использован Microsoft Project.

1.6 Планирование разработки и оценка бюджета

Основной целью структурного планирования является описание состава и взаимосвязи технологических операций, которые требуются для реализации проекта. Также на этом этапе необходимо определить или предварительно оценить продолжительности работ.

Структурное планирование можно разделить на следующие этапы:

- разбиение проекта на совокупность отдельных работ, выполнение которых необходимо для реализации проекта;
- построение сетевого графика, описывающего последовательность выполнения работ;
- оценка временных характеристик работ и анализ сетевого графика.

Результатом структурного планирования является сетевой график – ориентированный граф, вершинами которого являются работы проекта, а дугами временные взаимосвязи.

Он должен удовлетворять следующим свойствам:

- каждой работе соответствует только одна вершина;
- ни одна работа не может быть начата, пока не закончатся все непосредственно предшествующие ей работы;
- ни одна работа, которая непосредственно следует за некоторой работой, не может начаться до момента ее окончания;
- начало и конец проекта (вехи) обозначены работами с нулевой продолжительностью.

Для оценки стоимости и трудоемкости проекта необходимо провести правильное планирование. Для этого были определены задачи и ресурсы,

которые необходимы для создания игрового приложения. В таблице 6 представлены задачи, которые представляют этапы создания проекта.

Таблица 6 – Перечень задач проекта

Название задачи	Длительность	Начало	Окончание	Предшественники
Начало	0 дней	Вт 01.03.22	Вт 01.03.22	
Анализ требований	10 дней	Вт 01.03.22	Пн 14.03.22	
Анализ аналогичных решений	6 дней	Вт 01.03.22	Вт 08.03.22	1
Анализ правил проведения соревнований	3 дней	Ср 09.03.22	Пт 11.03.22	3
Принятие окончательное решения	1 день	Пн 14.03.22	Пн 14.03.22	4
Анализ требований завершен	0 дней	Пн 14.03.22	Пн 14.03.22	5
Выбор средств реализации проекта	8 дней	Вт 01.03.22	Чт 10.03.22	
Анализ возможных программных средств	6 дней	Вт 01.03.22	Вт 08.03.22	1
Выбор программного средства для реализации	2 дней	Ср 09.03.22	Чт 10.03.22	8
Выбор средств завершен	0 дней	Чт 10.03.22	Чт 10.03.22	9
Реализация проекта	12 дней	Вт 15.03.22	Ср 30.03.22	
Разработка структуры данных	2 дней	Вт 15.03.22	Ср 16.03.22	6; 10
Разработка игрового приложения	10 дней	Чт 17.03.22	Ср 30.03.22	
Программирование логики приложения	3 дней	Чт 17.03.22	Пн 21.03.22	12
Разработка редактора карт	7 дней	Вт 22.03.22	Ср 30.03.22	14
Разработка пользовательского интерфейса	10 дней	Чт 17.03.22	Ср 30.03.22	
Разработка макета интерфейса приложения	4 дней	Чт 17.03.22	Вт 22.03.22	12
Верстка приложения	6 дней	Ср 23.03.22	Ср 30.03.22	17
Реализация проекта завершена	0 дней	Ср 30.03.22	Ср 30.03.22	15; 18
Отладка	13 дней	Чт 31.03.22	Пн 18.04.22	
Тестирование пользовательского интерфейса	3 дней	Чт 31.03.22	Пн 04.04.22	19
Исправление ошибок	10 дней	Вт 05.04.22	Пн 18.04.22	21
Тестирование завершено	0 дней	Пн 18.04.22	Пн 18.04.22	22
Внедрение программного средства в систему проведения соревнований города	15 дней	Вт 19.04.22	Пн 09.05.22	
Внедрение программного средства	10 дней	Вт 19.04.22	Пн 02.05.22	23
Поиск и исправления ошибок при внедрении	5 дней	Вт 03.05.22	Пн 09.05.22	25
Внедрение завершено	0 дней	Пн 09.05.22	Пн 09.05.22	26
Конец проекта	0 дней	Пн 09.05.22	Пн 09.05.22	27

Из таблицы, представленной выше видно, что для реализации проекта необходимо выполнить 28 задач. Список ресурсов представлен на таблице 7.

Таблица 7 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка
Дизайнер-программист	Трудовой	300,00 Р/ч
Программист	Трудовой	200,00 Р/ч
Дизайнер	Трудовой	150,00 Р/ч
Компьютер	Материальный	400,00 Р
Интернет	Затраты	
Геймдизайнер-программист	Трудовой	300,00 Р/ч

Календарное планирование проведем с помощью системы управления проектами Microsoft Office Project 2007 [5 – 9].

На рисунке 12 представлен список задач из таблицы 7, внесенный в Microsoft Project. Началом проекта была принята дата 01.03.2022.

	Название задачи	Длительность	Начало	Окончание	Предшественн	Названия
1	Начало	0 дней	Вт 01.03.22	Вт 01.03.22		
2	Анализ требований	10 дней	Вт 01.03.22	Пн 14.03.22		
3	Анализ аналогичных решений	6 дней	Вт 01.03.22	Вт 08.03.22	1	Геймдизайнер-прс
4	Анализ правил проведения сорее	3 дней	Ср 09.03.22	Пт 11.03.22	3	Геймдизайнер-прс
5	Принятие окончательное решени	1 день	Пн 14.03.22	Пн 14.03.22	4	Геймдизайнер-прс
6	Анализ требований завершен	0 дней	Пн 14.03.22	Пн 14.03.22	5	
7	Выбор средств реализации проекта	8 дней	Вт 01.03.22	Чт 10.03.22		
8	Анализ возможных программных	6 дней	Вт 01.03.22	Вт 08.03.22	1	Геймдизайнер-прс
9	Выбор программного средства дл	2 дней	Ср 09.03.22	Чт 10.03.22	8	Геймдизайнер-прс
10	Выбор средств завершен	0 дней	Чт 10.03.22	Чт 10.03.22	9	
11	Реализация проекта	12 дней	Вт 15.03.22	Ср 30.03.22		
12	Разработка структуры данных	2 дней	Вт 15.03.22	Ср 16.03.22	6;10	Геймдизайнер-прс
13	Разработка игрового приложения	10 дней	Чт 17.03.22	Ср 30.03.22		
14	Программирование логики при	3 дней	Чт 17.03.22	Пн 21.03.22	12	Геймдизайнер-прс
15	Разработка редактора карт	7 дней	Вт 22.03.22	Ср 30.03.22	14	Геймдизайнер-прс
16	Разработка пользовательского ин	10 дней	Чт 17.03.22	Ср 30.03.22		
17	Разработка макета интерфейса	4 дней	Чт 17.03.22	Вт 22.03.22	12	Дизайнер;Компью
18	Верстка приложения	6 дней	Ср 23.03.22	Ср 30.03.22	17	Программист;Комп
19	Реализация проекта завершена	0 дней	Ср 30.03.22	Ср 30.03.22	15;18	
20	Отладка	13 дней	Чт 31.03.22	Пн 18.04.22		
21	Тестирование пользовательского	3 дней	Чт 31.03.22	Пн 04.04.22	19	Геймдизайнер-прс
22	Исправление ошибок	10 дней	Вт 05.04.22	Пн 18.04.22	21	Геймдизайнер-прс
23	Тестирование завершено	0 дней	Пн 18.04.22	Пн 18.04.22	22	
24	Внедрение программного средства	15 дней	Вт 19.04.22	Пн 09.05.22		
25	Внедрение программного средст	10 дней	Вт 19.04.22	Пн 02.05.22	23	Геймдизайнер-прс
26	Поиск и исправления ошибок при	5 дней	Вт 03.05.22	Пн 09.05.22	25	Геймдизайнер-прс
27	Внедрение завершено	0 дней	Пн 09.05.22	Пн 09.05.22	26	
28	Конец проекта	0 дней	Пн 09.05.22	Пн 09.05.22	27	

Рисунок 12 – Задачи проекта в среде Microsoft Project

На рисунке 13 представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

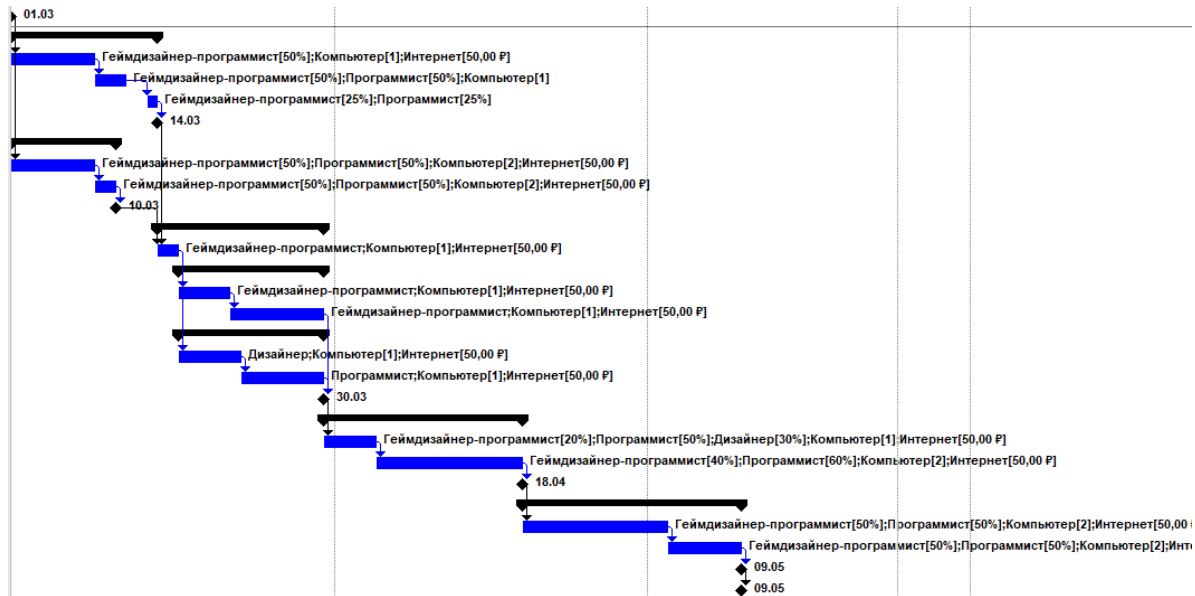


Рисунок 13 – Диаграмма Ганта

Из диаграммы Ганта можно определить интервалы работы сотрудников и сроки их завершения, что позволит оптимизировать процессы разработки мобильного приложения.

На рисунке 14 представлен обзор проекта.

Даты			
Начало:	Вт 01.03.22	Окончание:	Пн 09.05.22
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	50 дней	Оставшаяся:	50 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	50 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	516 ч	Оставшиеся:	516 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	516 ч	Процент завершения:	0%
Затраты			
Текущий план:	135 320,00 Р	Оставшиеся:	135 320,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	135 320,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	28	Трудовые ресурсы:	4
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дос:	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	28	Итого ресурсов:	5

Рисунок 14 – Обзорный отчет проекта

Из рисунка видно, что общая длительность проекта составляет 50 дней, а предполагаемые затраты – 135 320 руб.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

В процессе разработки приложения было решено использовать методологию RUP [10]. Разработка приложения в соответствии с методологией RUP носит итеративный характер. На каждой итерации перед разработчиками ставятся определенные цели, которые должны быть достигнуты к ее концу. В результате в конце каждой итерации должна получиться промежуточная функциональная версия разрабатываемого мобильного приложения. В методологии RUP существуют четыре основных фазы, каждая из которых может включать в себя одну или несколько итераций:

- фаза начала проекта – формируется общее видение проекта и экономическое обоснование;
- уточнение – анализ предметной области и построение архитектуры продукта;
- построение – реализация основного функционала продукта;
- внедрение – получение финальной версии продукта и его передача заказчику, фаза включает в себя бета-тестирование и устранение выявленных в его процессе ошибок, а также обучение пользователей.

Каждая фаза завершается вехой, на которой команда разработчиков оценивает выполнение поставленных целей и принимает критически важные решения о ходе дальнейшей работы над продуктом.

Выбор в пользу данной методологии продиктован рядом ее достоинств. С одной стороны, методология RUP обеспечивает гибкость процесса разработки и позволяет быстро вносить коррективы в проект в случае выявления в нем ошибок или изменения требований к разрабатываемому приложению [11]. Кроме того, благодаря данной методологии удастся выявить основные риски уже на ранних стадиях разработки. При этом RUP, в отличие от некоторых других популярных методологий, например XP, не требует постоянного взаимодействия с заказчиком, которое в рамках данного проекта невозможно.

Для анализа требований, а также моделирования структуры проекта в соответствии с принципами RUP было решено использовать диаграммы на языке моделирования UML. Для построения диаграмм использовалось средство UML моделирования ArgoUML [12].

На этапе планирования проекта было решено использовать приложение Microsoft Project. Оно позволяет обеспечить эффективное управление проектом на всех стадиях его выполнения. Microsoft Project предоставляет возможность выполнить структуризацию проекта, составить его календарный план, а также рассчитать затраты трудовых, материальных и денежных ресурсов в процессе работы над проектом.

Для создания игрового приложения было решено использовать платформу разработки – Unity [13], а также среду программирования Microsoft Visual Studio 2019, с использованием языка программирования C#.

2.2 Проектирование архитектуры программного обеспечения

Диаграмма классов представлена на рисунке 15.

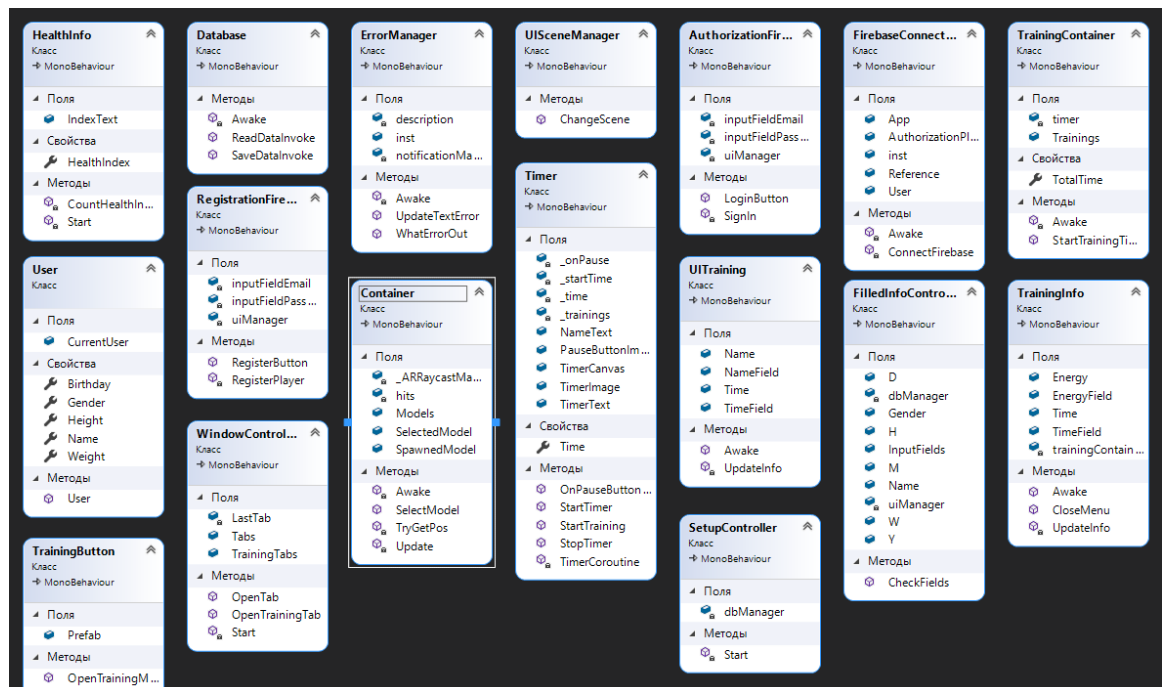


Рисунок 15 – Диаграмма классов

Описание классов игрового приложения представлены ниже:

– healthInfo – это класс подсчета индекса массы тела;

- user – это класс, содержащий всю информацию о пользователе;
- database – это класс, описывающий логику сохранение и загрузку данных пользователя на устройство;
- registrationFirebase – это класс, описывающий логику регистрации новых пользователей в системе;
- windowController – это класс, описывающий логику смены окон в приложении;
- errorManager – это класс, определяющий ошибки и уведомления приложения и, выводящий информацию на экран приложения в виде уведомлений;
- container – это класс, содержащий логику отображения анимированных моделей в дополненной реальности;
- uiSceneManager – это класс, содержащий логику смены сцен в приложении;
- timer – это класс, отвечающий за логику таймера в приложении;
- authorizationFirebase – это класс, описывающий логику авторизации зарегистрированных пользователей в системе;
- firebaseConnection – это класс, описывающий логику предварительного подключения к системе Firebase;
- trainingInfo – это класс, содержащий информацию о тренировки.

2.3 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был разработан интерфейс мобильного приложения. Он включает в себя страницы входа, регистрации, заполнения данных о здоровье пользователя и главного окна приложения.

Страница входа представлена на рисунке 16.

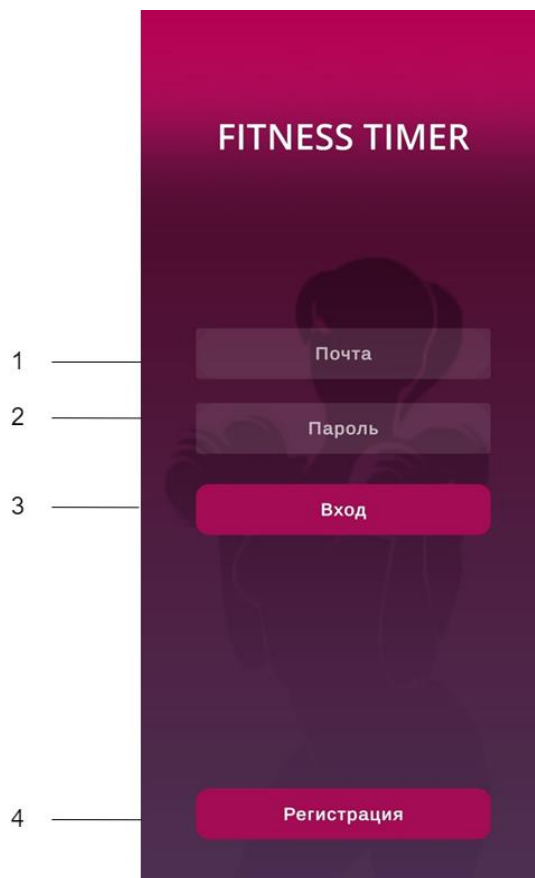


Рисунок 16 – Страница входа

Рассмотрим основные элементы интерфейса страницы авторизации, представленные на рисунке 16:

- 1 – поле ввода логина пользователя;
- 2 – поле ввода пароля пользователя;
- 3 – кнопка «Вход»;
- 4 – кнопка перехода на страницу регистрации.

При нажатии на кнопку «Вход» система проверяет, существует ли на самом деле пользователь с таким логином и паролем. Если такого пользователя нет, будет выведено сообщение о неверном логине или пароле. В противном случае будет проверено, не заблокирован ли данный пользователь. Если он заблокирован, выводится сообщение о блокировке. Иначе авторизация будет считаться выполненной успешно.

Страница регистрации представлена на рисунке 17.



Рисунок 17 – Страница регистрации

Рассмотрим основные элементы интерфейса страницы регистрации, представленные на рисунке 17:

- 1 – поле ввода логина пользователя;
- 2 – поле ввода пароля пользователя;
- 3 – кнопка «Регистрация»;
- 4 – кнопка перехода на страницу входа.

При нажатии кнопки «Регистрация» система проверяет, являются ли введенные логин и пароль допустимыми. Если это условие не выполняется, будет выведено сообщение о недопустимом логине или пароле. В противном случае будет проверено, не занят ли данный логин. Если он занят, выводится соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

Страница ввода данных о здоровье пользователя представлена на рисунке 18.

Рисунок 18 – Страница ввода данных

Рассмотрим основные элементы интерфейса страницы ввода данных, представленные на рисунке 18:

- 1 – поле ввода дня рождения;
- 2 – поле ввода месяца рождения;
- 3 – поле ввода года рождения пользователя;
- 4 – поле ввода роста;
- 5 – поле ввода веса;
- 6 – поле ввода пола пользователя;
- 7 – кнопка «Продолжить».

При нажатии на кнопку «Продолжить» произойдет переход на главную страницу мобильного приложения, представленную на рисунке 19.

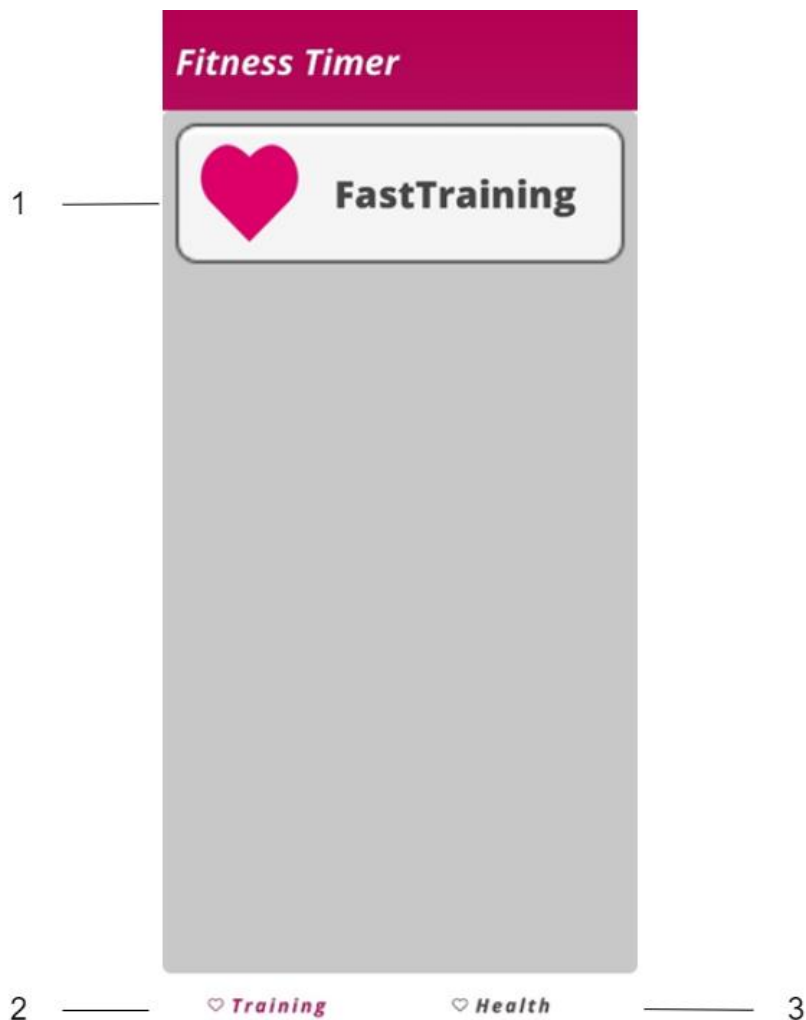


Рисунок 19 – Главная страница приложения

Рассмотрим основные элементы интерфейса главной страницы приложения, представленные на рисунке 19:

1 – кнопка «Быстрая тренировка»;

2 – кнопка «Тренировка»;

3 – кнопка «Здоровье».

При нажатии на кнопки «Быстрая тренировка» открывается окно с соответствующей спортивной тренировкой. Пример окна тренировки представлен на рисунке 20. При нажатии на кнопку «Тренировка» переходит возврат к окну с тренировками, представленному на рисунке 20.

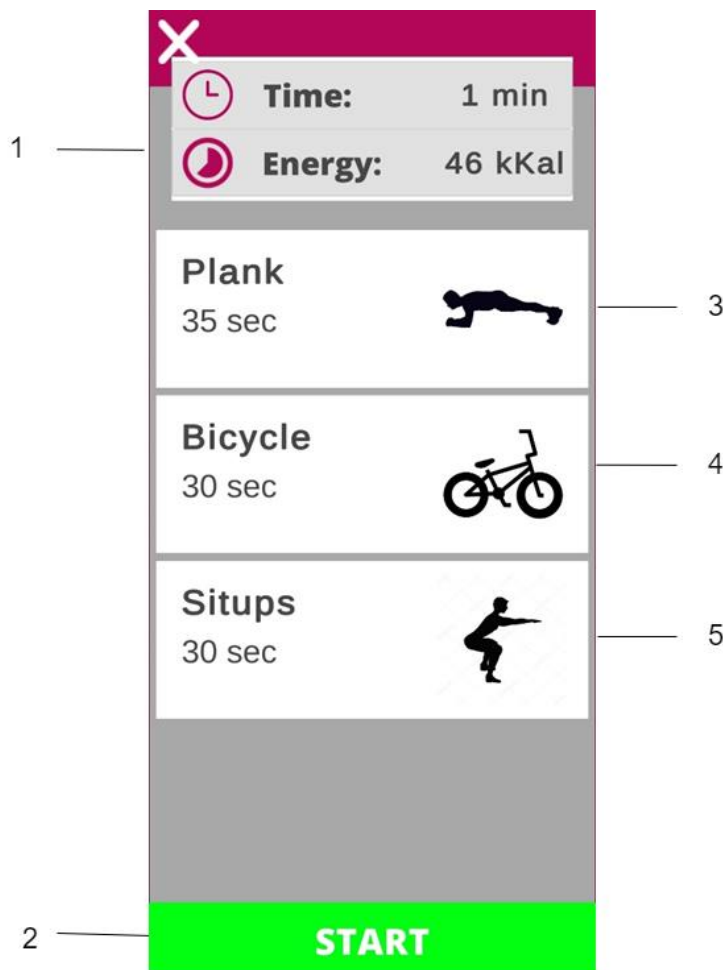


Рисунок 20 – Страница с информацией о тренировке

Рассмотрим основные элементы интерфейса страницы приложения тренировки, представленной на рисунке 20:

- 1 – текстовый блок;
- 2 – кнопка «Старт»;
- 3 – кнопка «Упражнение планка»;
- 4 – кнопка «Упражнение велосипед»;
- 5 – кнопка «Упражнение приседания».

В текстовом блоке отображается информация о выбранной тренировке. При нажатии на кнопку упражнения открывается демонстрация соответствующей анимированной модели в дополненной реальности. При нажатии на кнопку «Старт» происходит начало выбранной тренировки. При нажатии на кнопку «Здоровье» на главной странице приложения открывается окно с информацией о здоровье пользователя, рассчитанной на основании его данных. Данное окно представлено на рисунке 21.

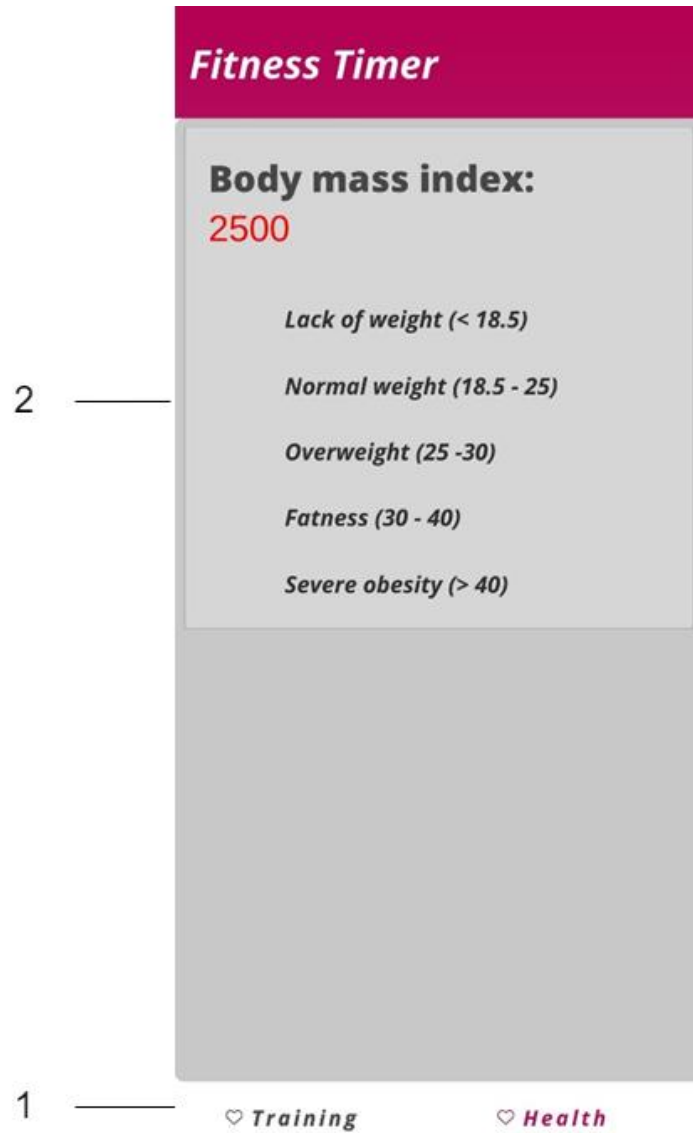


Рисунок 21 – Страница с информацией о здоровье пользователя

Рассмотрим основные элементы интерфейса страницы информации о здоровье, представленной на рисунке 21:

1 – кнопка «Тренировка»;

2 – текстовый блок.

При нажатии на кнопку «Тренировка» открывается главное окно приложения. В текстовом блоке располагается информация о здоровье пользователя.

3 Реализация мобильного приложения

3.1 Реализация клиентской части приложения

Разработка мобильного фитнес-приложения производилась на языке C# с использованием среды программирования Microsoft Visual Studio и мультиплатформенной инструментальной среды Unity [14]. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 22.

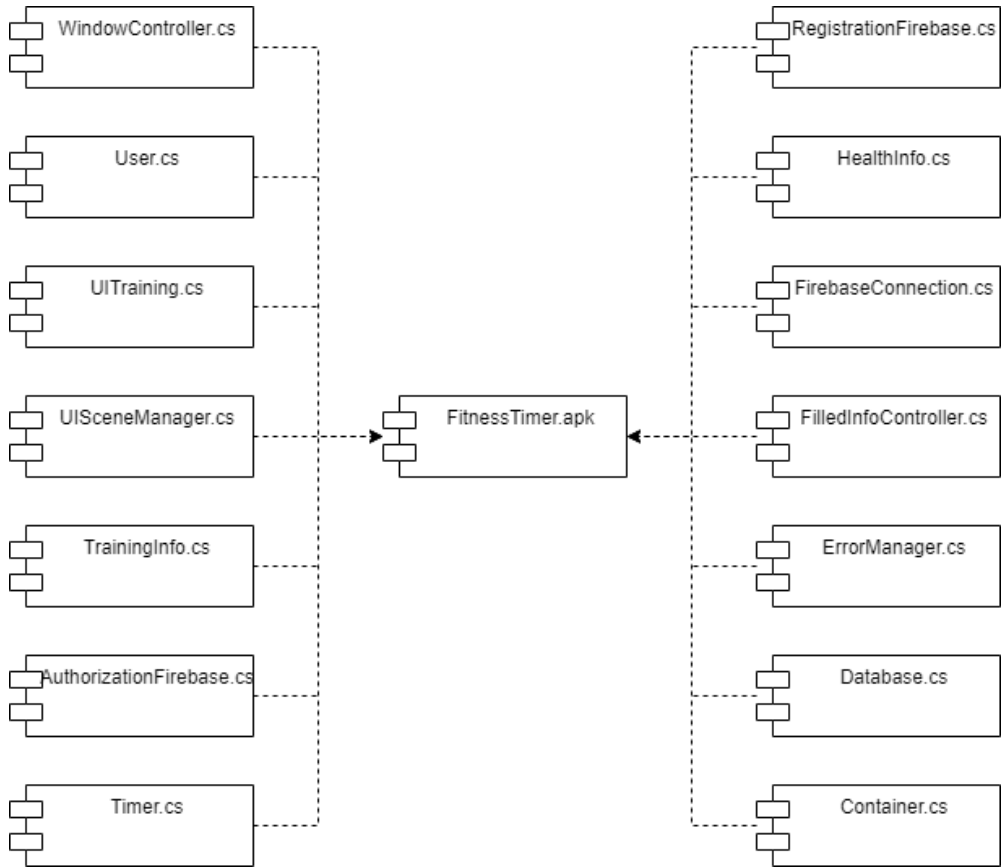


Рисунок 22 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 8.

Таблица 8 – Компоненты клиентского приложения

Компонент	Назначение
FitnessTimer.apk	Исполняемый файл приложения.
WindowController.cs	Файл класса, реализующий переход между окнами в главном меню приложения.
User.cs	Файл класса, хранящий информацию об авторизованном пользователе.

Продолжение таблицы 8

Компонент	Назначение
UITraining.cs	Файл класса, реализующий логику обновления информации о тренировке.
UISceneManager.cs	Файл класса, реализующий переход между сценами в приложении.
TrainingInfo.cs	Файл класса, хранящий информацию о тренировке.
AuthorizationFirebase.cs	Файл класса, реализующий логику авторизации зарегистрированного пользователя в системе.
Timer.cs	Файл класса, реализующий логику таймера.
RegistrationFirebase.cs	Файл класса, реализующий логику регистрации новых пользователей в системе.
HealthInfo.cs	Файл класса, реализующий логику расчета индекса массы тела пользователя.
FirebaseConnection.cs	Файл класса, реализующий логику настройки подключения приложения к облачным сервисам Firebase.
FilledInfoController.cs	Файл класса, реализующий логику валидации введенной пользователем информации о своем здоровье.
ErrorManager.cs	Файл класса, реализующий логику отображения уведомлений об ошибках в приложении.
DataBase.cs	Файл класса, реализующий логику сохранения и загрузки данных пользователя на устройсто.
Container.cs	Файл класса, хранящий анимированные модели для демонстрации в дополненной реальности.

Данные пользователя о его здоровье хранятся в реестре его устройства.

Листинг клиентской части программы представлен в приложении Б.

3.2 Реализация серверной части приложения

Авторизация и регистрация пользователя были реализованы посредством облачной платформы Firebase, предназначенной для разработки приложений с использованием облачных технологий. Основной сервис – облачная СУБД класса NoSQL, позволяющая разработчикам приложений хранить и синхронизировать данные между несколькими клиентами.

Поддержаны особенности интеграции с приложениями под операционные системы Android и iOS. Данное API для приложений было разработано на JavaScript, Java, Objective-C и Node.js.

4 Тестирование программного продукта

4.1 Тестирование черным ящиком

Тестирование черного ящика или поведенческое тестирование – стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта. Иначе говоря, тестированием черного ящика занимаются тестировщики, не имеющие доступ к исходному коду приложения [15]. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из технических требований и их спецификаций.

4.1.1 Классы эквивалентности

Классы эквивалентности – это техника тестирования ПО, которая делит вводимые данные на классы эквивалентных друг другу значений, на базе которых создаются тест-кейсы [16]. Было принято решение сформировать классы эквивалентности для пароля пользователя. Классы эквивалентности приведены на таблице 9.

Таблица 9 – Классы эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина пароля	4 - 32 (1)	0 - 3 (2) 33 - ∞ (3)
Структура	Начинается с специального символа (4)	Начинается не со специального символа (5)
Содержание	Буквы Латинские (6) Цифры (7) Символы: !@#\$ (8)	Остальное (9)

Таким образом были сформированы классы эквивалентности для функции валидации пароля пользователя.

4.1.2 Граничные условия

Граничное тестирование – это процесс тестирования между крайними концами или границами между разделами входных значений [17]. Таким образом, эти крайние значения, такие как начальное-конечное, нижнее-верхнее, максимальное-минимальное, значение Just Inside-Just Outside, называются граничными значениями, а тестирование называется «граничным тестированием». Граничные условия для заданного алгоритма приведены на таблице 10.

Таблица 10 – Граничные условия

Границы	Описание
Физические	Строка не может быть пустой
Технологические	Программа перестает работать при полной загрузке оперативной памяти
Произвольные	Пароль не больше 50 символов

Таким образом были сформированы граничные условия для мобильного приложения.

4.1.3 Тест-кейсы

По ранее заданным классам эквивалентности и граничным условиям были созданы тест-кейсы для их проверки. Тест-кейсы представлены в таблице 11.

Таблица 11 – Тест-кейсы

№	КЭ	Входные данные	Выходные данные
1	Все правильные(1, 4, 5, 8, 9, 10)	Rim-im@mail.ru	Успешно
2	(2, 4, 5, 8, 10)	V@m.ru	Недопустимая почта
3	(3, 4, 5, 8, 9, 10)	Zewtrxytcvigbohioi,uymdtnsrxtfc vghbnijplukfjdhngblhi,ufymjhclbi vhvhvmgnfhgchcmhllvykck@m. r	Больше 50 символов
4	(1, 5, 6, 8, 9, 10)	\$@ras@ib.ru	Ошибка начинается со специального символа

Таким образом были сформированы тест-кейсы для приложения.

4.2 Функциональное тестирование белым ящиком

Для проверки корректности работы клиентской части приложения было проведено функциональное тестирование. Его результаты представлены в таблице 12.

Таблица 12 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Ввести валидные данные для регистрации. Email: roma@mail.ru. Пароль: 123321.	Тест пройден. Пользователь зарегистрирован в системе (рисунок В.1).
	Ввести данные с некорректным паролем. Email: dasd@mail.ru. Пароль: 123.	Тест пройден. Выведено уведомление о том, что пароль является слабым (рисунок В.2).
	Ввести данные с некорректной почтой. Email: dasd@mai. Пароль: 123.	Тест пройден. Выведено уведомление о том, что Email является некорректным (рисунок В.3).
	Ввести неполные данные для регистрации. Email: dasd@mail.ru. Пароль: -	Тест пройден. Выведено уведомление о том, что пароль не введен (рисунок В.4).
	Ввести неполные данные для регистрации. Email: - Пароль: 123321.	Тест пройден. Выведено уведомление о том, что Email не введен (рисунок В.5).
Авторизоваться	Ввести валидные данные для авторизации. Email: roma@mail.ru. Пароль: 123321.	Тест пройден. Пользователь авторизован в системе (рисунок В.6).
	Ввести данные с некорректным паролем. Email: roma@mail.ru. Пароль: 123.	Тест пройден. Выведено уведомление о том, что пароль является неправильным (рисунок В.7).
	Ввести данные с некорректной почтой. Email: dasd@mai. Пароль: 123.	Тест пройден. Выведено уведомление о том, что Email не был найден (рисунок В.8).

Продолжение таблицы 12

Вариант использования	Тест и параметры	Результат
	Ввести неполные данные для авторизации. Email: dasd@mail.ru. Пароль: -	Тест пройден. Выведено уведомление о том, что пароль не введен (рисунок В.9).
	Ввести неполные данные для авторизации. Email: - Пароль: 123321.	Тест пройден. Выведено уведомление о том, что Email не введен (рисунок В.10).

Функциональное тестирование показало корректность работы клиентской части мобильного приложения.

4.2.1 Алгоритм мобильного приложения

В качестве тестируемого алгоритма было принято решение взять функцию проверки уникальности имени пользователя при регистрации в системе. На примере данной функции, наиболее удобно рассмотреть

Покрытие решений, условий и маршрутов выбранной функции приведено в таблице 13.

Таблица 13 – Покрытие параметров функции

Тип	Условия	Входные данные	Выходные данные
Покрытие решений	abdhi	Username = ""	Некорректное имя пользователя
	abfhi	Username = "Roma"	Пользователь успешно зарегистрирован в системе
	abghi	Username = "Roma"	Данное имя пользователя уже используется
Покрытие условий	Username.Length <= 0 – True.	Username = ""	Некорректное имя пользователя
	Username.Length <= 0 – False. SameUserName == 0 – False	Username = "Roma"	Данное имя пользователя уже используется

Продолжение таблицы 13

Тип	Условия	Входные данные	Выходные данные
	Username.Length <= 0 – False. SameUserName == 0 – True	Username = "Roma"	Пользователь успешно зарегистрирован в системе
Покрытие маршрутов	abcdhi	Username = ""	Некорректное имя пользователя
	abcefh	Username = "Roma"	Пользователь успешно зарегистрирован в системе
	abceghi	Username = "Roma"	Данное имя пользователя уже используется

Таким образом были сформулированы покрытие решений, покрытие условий, покрытие маршрутов для выбранной функции [18 – 20].

4.3 Метрики кода

Метриками кода называются некоторые количественные величины, позволяющие подсчитать, насколько код программы обладает тем или иным свойством [21]. Существует огромное количество самых разнообразных метрик, имеющих различное назначение. Наиболее известными являются следующие группы метрик:

- метрики размера программы (SLOC-метрики);
- метрики сложности потока управления программ (метрики Холстеда и цикломатическая сложность).

Рассмотрим метод Login класса UserController, отвечающий за обработку запроса на вход в систему существующего пользователя нового пользователя на сервере. Его код представлен в листинге 1.

Листинг 1 – Вход в систему

```
private Message CheckUserName(string username, List<User>
registeredUsers)
{
    //Сообщение, которое будет выведено в результате
    Message res;
```

```

        if (username.Length <= 0)
            res = Message.BadUsername;
        else
        {
            //Поиск данного имени, среди зарегистрированных
пользователей
            var sameUserName = registeredUsers.Find(x => x.Name ==
username);
            if (sameUserName == 0)
            {
                res = Message.Success;
            }
            else
            {
                //Имя пользователя уже есть в системе
                res = Message.BusyUsername;
            }
        }
        return res;
    }
}

```

Данный метод находится в серверной части приложения и запускается при попытке произвести вход пользователя в систему. Он проверяет существование пользователя с заданными логином и паролем. В случае нахождения такого пользователя он также выясняет, не заблокирован ли он, и, если блокировка отсутствует, подтверждает вход пользователя в систему. При неверном логине или пароле или наличии у пользователя активной блокировки отправляется соответствующее сообщение об ошибке.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Данные метрики представляют собой подсчет количества строк программного кода, а также числа строк с комментариями. Вычислим метрики

размера программы для приведенного выше программного кода. Результаты представлены в таблице 14.

Таблица 14 – Покрытие маршрутов

Метрика	Количество
Количество строк кода	25
Строки-комментарии	5

Также вычислим процент комментариев. Он равен $(5 / 25) * 100 = 20$. Это свидетельствует о достаточном количестве комментариев в коде для его хорошего понимания при отсутствии избыточных комментариев, переполняющих программу.

4.3.2 Цикломатическая сложность

Граф потока управления функции представлен на рисунке 23.

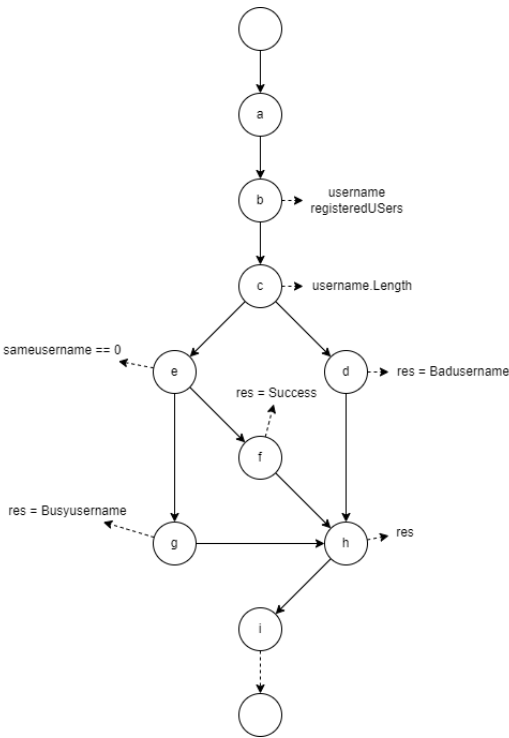


Рисунок 23 – Граф потока управления

На основе данного графа была вычислена цикломатическая сложность программного кода [22].

Вычисление данных:

- количество дуг

$$E = 11;$$

- количество вершин

$$N = 7;$$

- количество компонент связности

$$P = 2;$$

- цикломатическая сложность

$$CC = E - N + 2P = 11 - 7 + 2 = 5.$$

Таким образом цикломатическая сложность алгоритма $CC = 5$.

4.3.3 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить следующие характеристики программы:

- число уникальных операторов;
- число уникальных операндов;
- общее число операторов.

С целью вычисления этих характеристик были составлен словарь операторов (таблица 15) и словарь операндов (таблица 16).

Таблица 15 – Словарь операторов

Оператор	Число операторов
if	2
==	2
;	6
()	4
=	4
{}	4
return	1
&&	1
<=	1

Таким образом, на основании выбранной функции, был построен словарь операторов, с помощью которого можно рассчитать основные характеристики кода.

Таблица 16 – Словарь операндов

Операнд	Число операндов
res	5
sameUserName	2
registeredUsers	2
username	3
Message	4
Name	1

Таким образом, число уникальных операторов программы $n1 = 9$, число уникальных операндов программы $n2 = 6$, общее число операторов в программе $N1 = 25$, общее число операндов в программе $N2 = 17$ [23].

На основе метрик Холстеда можно вычислить целый ряд характеристик программы [24]. Рассмотрим некоторые из них:

- словарь программы (n):

$$n = n1 + n2;$$

- длина программы (N):

$$N = N1 + N2;$$

- объем программы (V):

$$V = N * \log_2 n;$$

- сложность программы (D):

$$D = (n1 / 2) * (N2 / n2);$$

- оценка интеллектуальных усилий (E):

$$E = D * V.$$

Рассчитаем словарь программы: $n = 9 + 6 = 15$.

Рассчитаем длину программы $N = 25 + 17 = 42$.

Рассчитаем объем программы: $V = 42 * \log_2 15 = 163,8$.

Рассчитаем сложность программы: $D = (9 / 2) * (17 / 6) = 12,75$.

Рассчитаем оценку усилий: $E = 12,75 * 163,8 = 2088,45$.

Полученные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки [25].

Заключение

В ходе выполнения выпускной квалификационной работы разработано мобильное фитнес-приложение для тренировок дома.

На этапе анализа требований и предметной области выполнен анализ программных продуктов аналогичного назначения, в результате которого выявлены их основные достоинства и недостатки, определены функциональные и нефункциональные требования к создаваемому приложению и произведен анализ целевой аудитории.

Осуществлено планирование разработки приложения и оценка бюджета. Результаты планирования показали, что общая длительность проекта составляет 50 дней, а предполагаемые затраты на реализацию – 135 320 рублей, что свидетельствует об экономической целесообразности данного программного проекта.

На этапе проектирования разработана архитектура мобильного приложения. Приложение разработано с помощью платформы разработки Unity 2020.3.8f на языке C# с использованием среды программирования Microsoft Visual Studio 2019.

Выполнено тестирование мобильного приложения. Оно показало корректность его работы. Также рассчитаны цикломатическая сложность и метрики программного кода для оценки его качества.

В дальнейшем планируется добавить в приложение новые тренировки и упражнения, а также опубликовать данное приложение Google Play и App Store.

Таким образом, в ходе выполнения выпускной квалификационной работы успешно выполнены все поставленные задачи.

Список использованных источников

1. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 21.04.2022).
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 21.04.2022).
3. Данные об оборудовании и ПО пользователей // Steam – платформа для игроков и разработчиков [Электронный ресурс]. URL: <https://store.steampowered.com/hwsurvey> (дата обращения: 21.04.2022).
4. Explore the world of Windows 11 // Microsoft [Электронный ресурс]. URL: <https://www.microsoft.com/en-us/windows/windows-11?r=1> (дата обращения: 21.04.2022).
5. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.
6. Методология разработки программного обеспечения RUP. Рабочие потоки, действия и артефакты стадий [Электронный ресурс]. URL: <https://skarlupka.ru/articles.php?id=38> (дата обращения: 05.05.2022).
7. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
8. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
9. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. - СПб.: Питер, 2019. – 208 с.
10. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
11. Унифицированный процесс Rational (RUP) // Интернет-технологии.ру [Электронный ресурс]. – URL: <https://www.internet->

technologies.ru/articles/newbie/unificirovannyu-process-rational-rup.html (дата обращения: 05.05.2022).

12. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.

13. Описание продукта // Официальный сайт Unity [Электронный ресурс]. URL: <https://unity.com/> (дата обращения: 21.04.2022).

14. Бонд Д. Г. Unity и C#. Геймдев от идеи до реализации, 2-е издание / Бонд Д. Г. // Питер, 2022. – 928 с.

15. Тестирование методом черного ящика // habr.com [Электронный ресурс]. URL: <https://habr.com/ru/post/462837/> (дата обращения: 27.05.2022).

16. Классы эквивалентности // НОУ «ИНТУИТ» [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/1040/209/lecture/5388> (дата обращения: 27.05.2022).

17. Техника анализа граничных значений // QA Evolution [Электронный ресурс]. URL: <https://qaevolution.ru/testovaya-dokumentaciya/test-dizajn/tehnika-analiza-granichnyx-znachenij/> (дата обращения: 27.05.2022).

18. Метод покрытия операторов // Оценка качества программного обеспечения вычислительных систем tstu.ru [Электронный ресурс]. URL: https://tstu.ru/book/elib3/mm/2015/loskutov/pages/4_3_1_metod_pokrytija_operat_urov.html (дата обращения: 27.05.2022).

19. Метод покрытия условий // Оценка качества программного обеспечения вычислительных систем tstu.ru [Электронный ресурс]. URL: https://tstu.ru/book/elib3/mm/2015/loskutov/pages/4_3_3_metod_pokrytija_uslovij.html (дата обращения: 27.05.2022).

20. Тестирование маршрутов. Структурное тестирование. // <https://studopedia.ru/> [Электронный ресурс]. URL: https://studopedia.ru/9_40672_strukturnoe-testirovanie.html (дата обращения: 27.05.2022).

21. Метрики кода и практическая реализация по их сбору и анализу. Часть 1 – метрики // <https://club.cnews.ru/> [Электронный ресурс]. URL: https://club.cnews.ru/blogs/entry/metriki_koda_i_prakticheskaya_realizatsiya_po_ih_sboru_i_analizu_chast_1_ (дата обращения: 27.05.2022).

22. Цикломатическое тестирование // уроки, учебники и другие материалы по программированию <https://coderlessons.com/> [Электронный ресурс]. URL: <https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/tsiklomaticheskoe-testirovanie> (дата обращения: 27.05.2022).

23. Программный код и его метрики // habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 27.05.2022).

24. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.

25. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

Приложение А
ГЛОССАРИЙ
(справочное)

Мобильное приложение – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для платформ iOS, Android, Windows Phone.

Android – мобильная операционная система для смартфонов, планшетов и других мобильных устройств, разрабатываемая и выпускаемая американской компанией Google и используемая на подавляющем большинстве современных мобильных устройств.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит итеративная модель разработки.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Приложение Б
ИСХОДНЫЙ КОД МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Б.1 – Файл AuthorizationFirebase.cs

```

using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using Firebase;
using TMPro;

public class AuthorizationFirebase : MonoBehaviour
{
    [SerializeField] private TMP_InputField inputFieldEmail,
inputFieldPassword;
    [SerializeField] private UIManager uiManager;

    public void LoginButton()
    {
        StartCoroutine(SignIn(inputFieldEmail.text,
inputFieldPassword.text));
    }

    private IEnumerator SignIn(string email, string password)
    {
        var loginTask =
FirebaseConnectionSetup.AuthorizationPlayer.SignInWithEmailAndPasswordAsync(e
mail, password);
        yield return new WaitUntil(predicate: () => loginTask.IsCompleted);

        if (loginTask.Exception != null)
        {
            ErrorManager.inst.WhatErrorOut(loginTask.Exception.GetBaseException() as
FirebaseException);
        }
        else
        {
            FirebaseConnectionSetup.User = loginTask.Result;;
            uiManager.ChangeScene(2);
            ErrorManager.inst.UpdateTextError("Успешная авторизация!");
        }
    }
}

```

Листинг Б.2 – Файл Container.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.XR.ARFoundation;
public class Container : MonoBehaviour
{
    public List<GameObject> Models;

    public GameObject SelectedModel;
    public GameObject SpawnedModel;
    private ARRaycastManager _ARRaycastManager;

    static List<ARRaycastHit> hits = new List<ARRaycastHit>();

    private void Awake()
    {
        _ARRaycastManager = FindObjectOfType<ARRaycastManager>();
    }
}

```

```

public void SelectModel(string name)
{
    if (null != SelectedModel)
        Destroy(SelectedModel);

    if (SpawnedModel != null)
        Destroy(SpawnedModel);

    SelectedModel = Models.Find((x) => x.name == name);
}

private bool TryGetPos(out Vector2 touchPosition)
{
    if (Input.touchCount > 0)
    {
        touchPosition = Input.GetTouch(0).position;
        return true;
    }

    touchPosition = default;
    return false;
}

private void Update()
{
    if (SelectedModel == null)
        return;

    if (!TryGetPos(out Vector2 touchPosition))
    {
        return;
    }

    if (_ARRaycastManager.Raycast(touchPosition, hits,
UnityEngine.XR.ARSubsystems.TrackableType.PlaneWithinPolygon))
    {
        var hitPos = hits[0].pose;
        if (null == SpawnedModel)
            SpawnedModel = Instantiate(SelectedModel, hitPos.position,
hitPos.rotation);
        else
            SpawnedModel.transform.position = hitPos.position;
    }
}
}

```

Листинг В.3 – Файл Database.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Firebase.Database;
using Firebase;

public class Database : MonoBehaviour
{
    public void SaveDataInvoke()
    {
        string info =
$"{{User.CurrentUser.Birthday}},{{User.CurrentUser.Height}},{{User.CurrentUser.Weigh
ght}},{{User.CurrentUser.Gender}},{{User.CurrentUser.Name}}";
        PlayerPrefs.SetString("UserInfo", info);
    }

    public void ReadDataInvoke()

```

```

    {
        string[] info = PlayerPrefs.GetString("UserInfo").Split(',');

        User.CurrentUser = new User(info[0], info[1], info[2], info[3],
info[4]);
    }

    private void Awake()
    {
        Application.targetFrameRate = 60;
    }
}

```

Листинг В.4 – Файл ErrorManager.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Firebase.Auth;
using Firebase;
using TMPro;
using Michsky.UI.ModernUIPack;

public class ErrorManager : MonoBehaviour
{
    [SerializeField] private NotificationManager notificationManager;
    [SerializeField] private TextMeshProUGUI description;

    public static ErrorManager inst;

    private void Awake()
    {
        if (inst == null)
        { // Экземпляр менеджера был найден
            inst = this; // Задаем ссылку на экземпляр объекта
        }
        else if (inst == this)
        { // Экземпляр объекта уже существует на сцене
            Destroy(gameObject); // Удаляем объект
        }

        Application.targetFrameRate = 60;
        DontDestroyOnLoad(gameObject);
    }

    public void UpdateTextError(string newDescription)
    {
        description.SetText(newDescription);
        notificationManager.OpenNotification();
    }

    public void WhatErrorOut(FirebaseException firebaseException)
    {
        AuthError errorCode = (AuthError)firebaseException.ErrorCode;
        string message = string.Empty;
        switch (errorCode)
        {
            case AuthError.None:
                message = "None";
                break;
            case AuthError.Unimplemented:
                message = "Нереализованная функция!";
                break;
            case AuthError.Failure:
                message = "Неудача!";
                break;
        }
    }
}

```

```

case AuthError.InvalidCustomToken:
    message = "InvalidCustomToken";
    break;
case AuthError.CustomTokenMismatch:
    message = "CustomTokenMismatch";
    break;
case AuthError.InvalidCredential:
    message = "InvalidCredential";
    break;
case AuthError.UserDisabled:
    message = "UserDisabled";
    break;
case AuthError.AccountExistsWithDifferentCredentials:
    message = "AccountExistsWithDifferentCredentials";
    break;
case AuthError.OperationNotAllowed:
    message = "OperationNotAllowed";
    break;
case AuthError.EmailAlreadyInUse:
    message = "EmailAlreadyInUse";
    break;
case AuthError.RequiresRecentLogin:
    message = "RequiresRecentLogin";
    break;
case AuthError.CredentialAlreadyInUse:
    message = "CredentialAlreadyInUse";
    break;
case AuthError.InvalidEmail:
    message = "Некорректный Email";
    break;
case AuthError.WrongPassword:
    message = "Неправильный пароль";
    break;
case AuthError.TooManyRequests:
    message = "TooManyRequests";
    break;
case AuthError.UserNotFound:
    message = "UserNotFound";
    break;
case AuthError.ProviderAlreadyLinked:
    message = "ProviderAlreadyLinked";
    break;
case AuthError.NoSuchProvider:
    message = "NoSuchProvider";
    break;
case AuthError.InvalidUserToken:
    message = "InvalidUserToken";
    break;
case AuthError.UserTokenExpired:
    message = "UserTokenExpired";
    break;
case AuthError.NetworkRequestFailed:
    message = "NetworkRequestFailed";
    break;
case AuthError.InvalidApiKey:
    message = "InvalidApiKey";
    break;
case AuthError.AppNotAuthorized:
    message = "AppNotAuthorized";
    break;
case AuthError.UserMismatch:
    message = "UserMismatch";
    break;
case AuthError.WeakPassword:
    message = "Слабый пароль!";
    break;

```

```

case AuthError.NoSignedInUser:
    message = "NoSignedInUser";
    break;
case AuthError.ApiNotAvailable:
    message = "ApiNotAvailable";
    break;
case AuthError.ExpiredActionCode:
    message = "ExpiredActionCode";
    break;
case AuthError.InvalidActionCode:
    message = "InvalidActionCode";
    break;
case AuthError.InvalidMessagePayload:
    message = "InvalidMessagePayload";
    break;
case AuthError.InvalidPhoneNumber:
    message = "InvalidPhoneNumber";
    break;
case AuthError.MissingPhoneNumber:
    message = "MissingPhoneNumber";
    break;
case AuthError.InvalidRecipientEmail:
    message = "InvalidRecipientEmail";
    break;
case AuthError.InvalidSender:
    message = "InvalidSender";
    break;
case AuthError.InvalidVerificationCode:
    message = "InvalidVerificationCode";
    break;
case AuthError.InvalidVerificationId:
    message = "InvalidVerificationId";
    break;
case AuthError.MissingVerificationCode:
    message = "MissingVerificationCode";
    break;
case AuthError.MissingVerificationId:
    message = "MissingVerificationId";
    break;
case AuthError.MissingEmail:
    message = "Email не введен!";
    break;
case AuthError.MissingPassword:
    message = "Пароль не введен!";
    break;
case AuthError.QuotaExceeded:
    message = "QuotaExceeded";
    break;
case AuthError.RetryPhoneAuth:
    message = "RetryPhoneAuth";
    break;
case AuthError.SessionExpired:
    message = "SessionExpired";
    break;
case AuthError.AppNotVerified:
    message = "AppNotVerified";
    break;
case AuthError.AppVerificationFailed:
    message = "AppVerificationFailed";
    break;
case AuthError.CaptchaCheckFailed:
    message = "CaptchaCheckFailed";
    break;
case AuthError.InvalidAppCredential:
    message = "InvalidAppCredential";
    break;

```

```

case AuthError.MissingAppCredential:
    message = "MissingAppCredential";
    break;
case AuthError.InvalidClientId:
    message = "InvalidClientId";
    break;
case AuthError.InvalidContinueUri:
    message = "InvalidContinueUri";
    break;
case AuthError.MissingContinueUri:
    message = "MissingContinueUri";
    break;
case AuthError.KeychainError:
    message = "KeychainError";
    break;
case AuthError.MissingAppToken:
    message = "MissingAppToken";
    break;
case AuthError.MissingIosBundleId:
    message = "MissingIosBundleId";
    break;
case AuthError.NotificationNotForwarded:
    message = "NotificationNotForwarded";
    break;
case AuthError.UnauthorizedDomain:
    message = "UnauthorizedDomain";
    break;
case AuthError.WebContextAlreadyPresented:
    message = "WebContextAlreadyPresented";
    break;
case AuthError.WebContextCancelled:
    message = "WebContextCancelled";
    break;
case AuthError.DynamicLinkNotActivated:
    message = "DynamicLinkNotActivated";
    break;
case AuthError.Cancelled:
    message = "Cancelled";
    break;
case AuthError.InvalidProviderId:
    message = "InvalidProviderId";
    break;
case AuthError.WebInternalError:
    message = "WebInternalError";
    break;
case AuthError.WebStorateUnsupported:
    message = "WebStorateUnsupported";
    break;
case AuthError.TenantIdMismatch:
    message = "TenantIdMismatch";
    break;
case AuthError.UnsupportedTenantOperation:
    message = "UnsupportedTenantOperation";
    break;
case AuthError.InvalidLinkDomain:
    message = "InvalidLinkDomain";
    break;
case AuthError.RejectedCredential:
    message = "RejectedCredential";
    break;
case AuthError.PhoneNumberNotFound:
    message = "PhoneNumberNotFound";
    break;
case AuthError.InvalidTenantId:
    message = "InvalidTenantId";
    break;

```

```

        case AuthError.MissingClientIdentifier:
            message = "MissingClientIdentifier";
            break;
        case AuthError.MissingMultiFactorSession:
            message = "MissingMultiFactorSession";
            break;
        case AuthError.MissingMultiFactorInfo:
            message = "MissingMultiFactorInfo";
            break;
        case AuthError.InvalidMultiFactorSession:
            message = "InvalidMultiFactorSession";
            break;
        case AuthError.MultiFactorInfoNotFound:
            message = "MultiFactorInfoNotFound";
            break;
        case AuthError.AdminRestrictedOperation:
            message = "AdminRestrictedOperation";
            break;
        case AuthError.UnverifiedEmail:
            message = "UnverifiedEmail";
            break;
        case AuthError.SecondFactorAlreadyEnrolled:
            message = "SecondFactorAlreadyEnrolled";
            break;
        case AuthError.MaximumSecondFactorCountExceeded:
            message = "MaximumSecondFactorCountExceeded";
            break;
        case AuthError.UnsupportedFirstFactor:
            message = "UnsupportedFirstFactor";
            break;
        case AuthError.EmailChangeNeedsVerification:
            message = "EmailChangeNeedsVerification";
            break;
        default:
            break;
    }
    UpdateTextError(message);
}
}

```

Листинг В.5 – Файл FilledInfoController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using TMPPro;
using UnityEngine;
using UnityEngine.UI;

public class FilledInfoController : MonoBehaviour
{
    [SerializeField] private UISceneManager uiManager;
    [SerializeField] private Database dbManager;

    public List<TMP_InputField> InputFields;

    public TMP_InputField D;
    public TMP_InputField M;
    public TMP_InputField Y;

    public TMP_InputField W;
    public TMP_InputField H;

    public TMP_InputField Name;
}

```

```

public Toggle Gender;
public void CheckFields()
{
    foreach (var field in InputFields)
    {
        if (field.text == "")
        {
            ErrorManager.inst.UpdateTextError("Не все поля заполнены!");
            return;
        }
    }

    string birthday = new DateTime(int.Parse(Y.text), int.Parse(M.text),
int.Parse(D.text)).ToString();
    string gender = Gender ? "male" : "female";
    User.CurrentUser = new User(birthday, H.text, W.text, gender,
Name.text);
    dbManager.SaveDataInvoke();
    uiManager.ChangeScene(2);
}
}

```

Листинг В.6 – Файл FirebaseConnectionSetup.cs

```

using Firebase;
using Firebase.Auth;
using Firebase.Database;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FirebaseConnectionSetup : MonoBehaviour
{
    public static FirebaseAuth AuthorizationPlayer;
    public static DatabaseReference Reference;
    public static FirebaseUser User;
    public static FirebaseApp App;
    public static FirebaseConnectionSetup inst;

    private void Awake()
    {
        if (inst == null)
        { // Экземпляр менеджера был найден
            inst = this; // Задаем ссылку на экземпляр объекта
        }
        else if (inst == this)
        { // Экземпляр объекта уже существует на сцене
            Destroy(gameObject); // Удаляем объект
        }

        Application.targetFrameRate = 60;
        DontDestroyOnLoad(gameObject);

        StartCoroutine(ConnectFirebase());
    }

    private IEnumerator ConnectFirebase()
    {
        var task = FirebaseApp.CheckAndFixDependenciesAsync();
        yield return new WaitUntil(predicate: () => task.IsCompleted);

        var dependencyStatus = task.Result;
        if (dependencyStatus == Firebase.DependencyStatus.Available)
        {
            // Create and hold a reference to your FirebaseApp,

```

```

        // where app is a Firebase.FirebaseApp property of your
application class.
        App = FirebaseApp.DefaultInstance;
        //App = FirebaseApp.DefaultInstance;
        Reference = FirebaseDatabase.DefaultInstance.RootReference;
        AuthorizationPlayer = FirebaseAuth.DefaultInstance;

        // Set a flag here to indicate whether Firebase is ready to use
by your app.
    }
    else
    {
        UnityEngine.Debug.LogError(System.String.Format(
            "Could not resolve all Firebase dependencies: {0}",
dependencyStatus));
        // Firebase Unity SDK is not safe to use here.
    }
}
}

```

Листинг В.7 – Файл HealthInfo.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using System;

public class HealthInfo : MonoBehaviour
{
    public TextMeshProUGUI IndexText;

    public float HealthIndex
    {
        get
        {
            return float.Parse(IndexText.text);
        }
        set
        {
            IndexText.SetText(value.ToString());
        }
    }

    private void Start()
    {
        HealthIndex = CountHealthIndex();
    }

    private float CountHealthIndex()
    {
        return float.Parse(User.CurrentUser.Weight) /
(float.Parse(User.CurrentUser.Height) / 100 *
float.Parse(User.CurrentUser.Height) / 100);
    }
}

```

Листинг В.8 – Файл RegistrationFirebase.cs

```

using Firebase.Auth;
using Firebase;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

```

```

using UnityEngine.UI;
using TMPro;
public class RegistrationFirebase : MonoBehaviour
{
    [SerializeField] private TMP_InputField inputFieldEmail,
inputFieldPassword;
    [SerializeField] private UIManager uiManager;

    public void RegisterButton()
    {
        StartCoroutine(RegisterPlayer(inputFieldEmail.text,
inputFieldPassword.text));
    }

    private IEnumerator RegisterPlayer(string email, string password)
    {
        var registerTask =
FirebaseConnectionSetup.AuthorizationPlayer.CreateUserWithEmailAndPasswordAsy
nc(email, password);
        yield return new WaitUntil(predicate: () =>
registerTask.IsCompleted);

        if (registerTask.Exception != null)
        {
            ErrorManager.inst.WhatErrorOut(registerTask.Exception.GetBaseException() as
FirebaseException);
        }
        else
        {
            FirebaseConnectionSetup.User = registerTask.Result;

            if (FirebaseConnectionSetup.User != null)
            {
                UserProfile profile = new UserProfile { DisplayName = email
};

                var profileTask =
FirebaseConnectionSetup.User.UpdateUserProfileAsync(profile);
                yield return new WaitUntil(predicate: () =>
profileTask.IsCompleted);

                if (profileTask.Exception != null)
                {
                    ErrorManager.inst.UpdateTextError(profileTask.Exception.Message);
                }
                else
                {
                    uiManager.ChangeScene(1);
                    ErrorManager.inst.UpdateTextError("Успешная
регистрация!");
                }
            }
        }
    }
}

```

Листинг В.9 – Файл SetupController.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SetupController : MonoBehaviour

```

```

{
    [SerializeField] private Database dbManager;

    private void Start()
    {
        dbManager.ReadDataInvoke();
    }
}

```

Листинг В.10 – Файл Timer.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;

public class Timer : MonoBehaviour
{
    public GameObject TimerCanvas;
    public Image TimerImage;
    public Image PauseButtonImage;
    public TextMeshProUGUI TimerText;
    public TextMeshProUGUI NameText;

    private List<UITraining> _trainings;

    private int _time;
    private int _startTime;
    private bool _onPause;
    public int Time
    {
        get
        {
            return _time;
        }
        set
        {
            _time = value;
            TimerText.SetText(_time.ToString());
            TimerImage.fillAmount = (float)_time / _startTime;
        }
    }

    public void OnPauseButtonClicked()
    {
        _onPause = !_onPause;
        PauseButtonImage.color = _onPause ? Color.red : Color.green;
        if (_onPause)
            StopTimer();
        else
            StartCoroutine(TimerCoroutine());
    }

    public void StartTimer()
    {
        StopTimer();
        _onPause = false;
        StartCoroutine(TimerCoroutine());
    }

    public void StartTraining(List<UITraining> trainings)
    {
        _trainings = trainings;
        TimerCanvas.SetActive(true);
    }
}

```

```

        StartTimer();
    }

    private IEnumerator TimerCoroutine()
    {
        foreach (var train in _trainings)
        {
            _startTime = train.Time;
            Time = train.Time;
            NameText.SetText(train.Name);

            while (Time != 0)
            {
                Time--;
                yield return new WaitForSeconds(1);
            }
        }

        yield return null;
    }

    public void StopTimer()
    {
        StopAllCoroutines();
    }
}

```

Листинг В.11 – Файл TrainingButton.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TrainingButton : MonoBehaviour
{
    public GameObject Prefab;
    public void OpenTrainingMenu()
    {
        Instantiate(Prefab);
    }
}

```

Листинг В.12 – Файл TrainingContainer.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TrainingContainer : MonoBehaviour
{
    public int TotalTime
    {
        get
        {
            var time = 0;
            foreach (var item in Trainings)
            {
                time += item.Time;
            }
            return time;
        }
    }
}

```

```

public List<UITraining> Trainings;

private Timer timer;

private void Awake()
{
    timer = FindObjectOfType<Timer>();
}
public void StartTrainingTimer()
{
    timer.StartTraining(Trainings);
}
}

```

Листинг В.13 – Файл TrainingInfo.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;

public class TrainingInfo : MonoBehaviour
{
    public TextMeshProUGUI TimeField;
    public TextMeshProUGUI EnergyField;

    private TrainingContainer trainingContainer;
    public int Time;
    public int Energy;

    public void Awake()
    {
        trainingContainer = GetComponent<TrainingContainer>();
        UpdateInfo();
    }

    private void UpdateInfo()
    {
        TimeField.SetText($"{trainingContainer.TotalTime / 60} min");
        EnergyField.SetText($"{Energy} kKal");
    }

    public void CloseMenu()
    {
        Destroy(gameObject);
    }
}

```

Листинг В.14 – Файл UISceneManager.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class UISceneManager : MonoBehaviour
{
    public void ChangeScene(int window)
    {
        SceneManager.LoadScene(window);
    }
}

```

Листинг В.15 – Файл UITraining.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

[ExecuteInEditMode]
public class UITraining : MonoBehaviour
{
    public TextMeshProUGUI NameField;
    public TextMeshProUGUI TimeField;

    public string Name;
    public int Time;

    public void Awake()
    {
        UpdateInfo();
    }

    private void UpdateInfo()
    {
        NameField.SetText($"{Name}");
        TimeField.SetText($"{Time} sec");
    }
}
```

Листинг В.16 – Файл User.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[Serializable]
public class User
{
    public static User CurrentUser;

    public string Birthday { get; set; }
    public string Height { get; set; }
    public string Weight { get; set; }
    public string Gender { get; set; }
    public string Name { get; set; }

    public User(string birthday, string height, string weight, string gender,
string name)
    {
        Birthday = birthday;
        Height = height;
        Weight = weight;
        Gender = gender;
        Name = name;
    }
}
```

Листинг В.17 – Файл WindowController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class WindowController : MonoBehaviour
{
    public List<GameObject> Tabs;
    public List<GameObject> TrainingTabs;

    private GameObject LastTab;

    private void Start()
    {
        LastTab = Tabs[0];
    }

    public void OpenTab(int index)
    {
        LastTab.SetActive(false);
        Tabs[index].SetActive(true);
        LastTab = Tabs[index];
    }

    public void OpenTrainingTab(int index)
    {
        TrainingTabs[index].SetActive(true);
    }
}
```

Приложение В
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

УВЕДОМЛЕНИЕ
Успешная регистрация!

FITNESS TIMER

Birthday
D M YEAR

Height
CM Centimeters

Weight
KG Kilograms

Gender
☒ Male ☐ Female

Info Next

Рисунок В.1 – Успешная регистрация в системе

< УВЕДОМЛЕНИЕ
Слабый пароль!

dasd@mail.ru

Регистрация

Рисунок В.2 – Слабый пароль

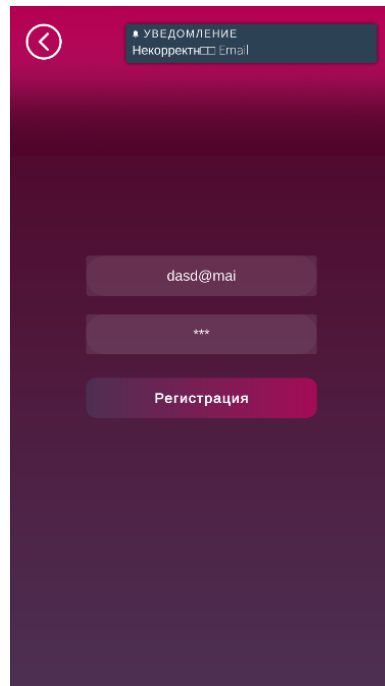


Рисунок В.3 – Некорректная почта

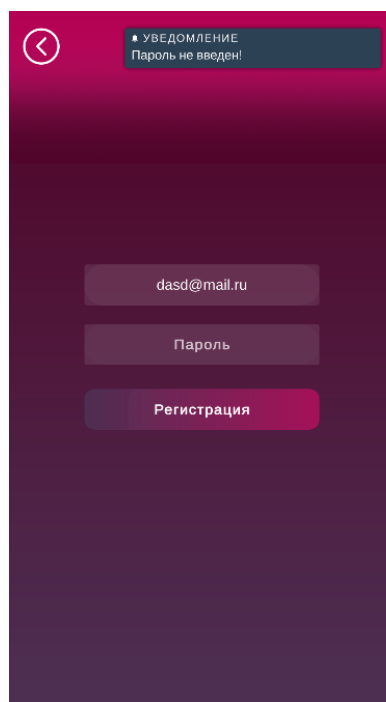


Рисунок В.4 – Пароль не введен

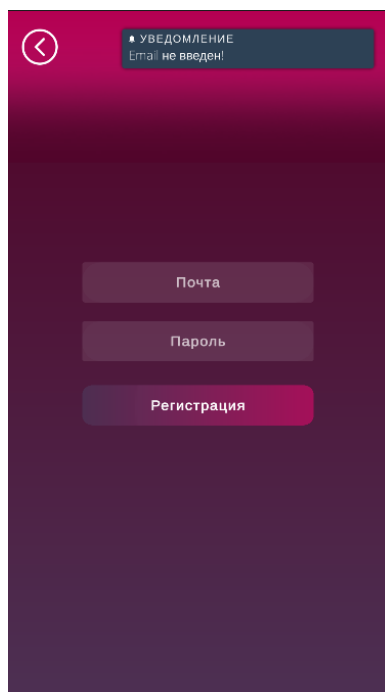


Рисунок В.5 – Почта не введена

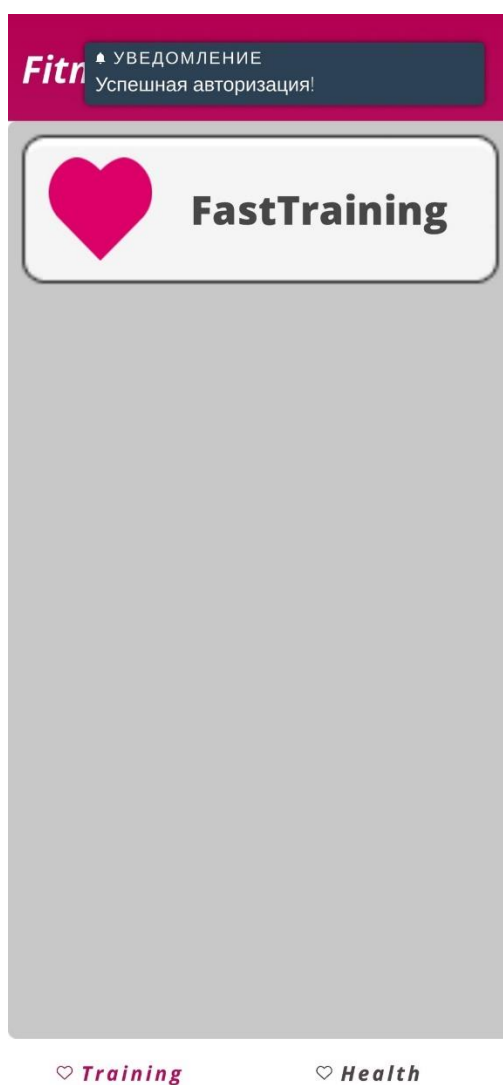


Рисунок В.6 – Успешная авторизация

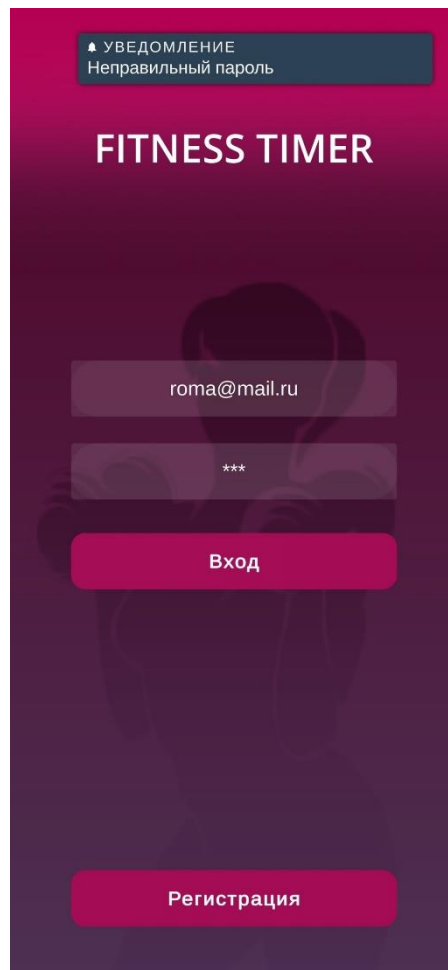


Рисунок В.7 – Неправильный пароль

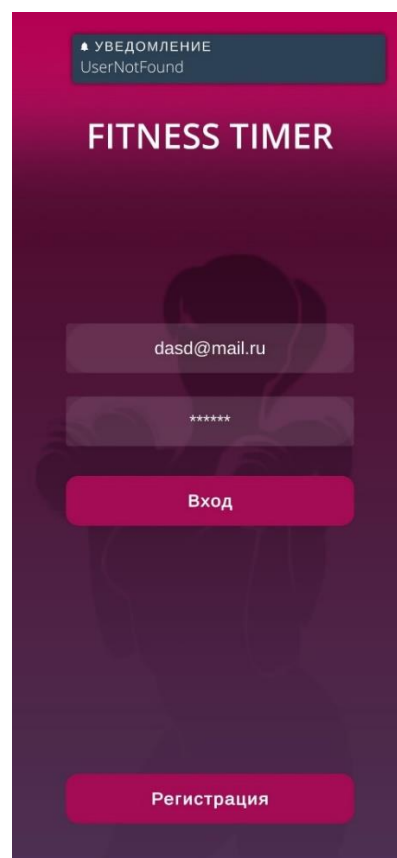
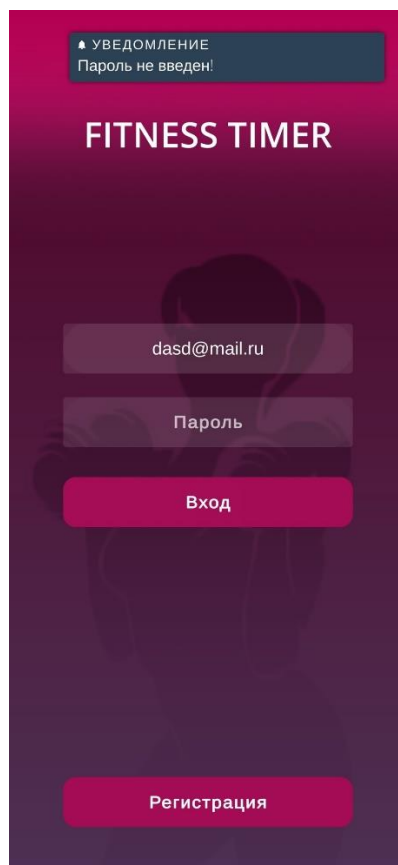


Рисунок В.8 – Почта не найдена



▲ УВЕДОМЛЕНИЕ
Пароль не введен!

FITNESS TIMER

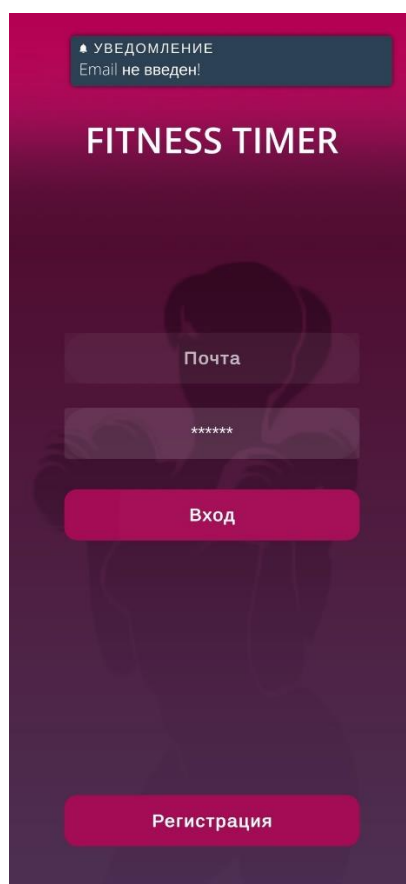
dasd@mail.ru

Пароль

Вход

Регистрация

Рисунок В.9 – Пароль не введен



▲ УВЕДОМЛЕНИЕ
Email не введен!

FITNESS TIMER

Почта

Вход

Регистрация

Рисунок В.10 – Почта не введена