

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

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Разработка программных средств сбора и анализа
кардиологических данных**

Студент

 14.06.21
(подпись, дата)

Федоров Дмитрий Алексеевич
(ФИО полностью)

Руководитель

 14.06.21
(подпись, дата)

Попова Н.А.
(фамилия, инициалы)

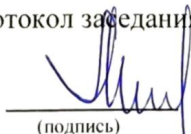
Нормоконтролер

 22.06.21
(подпись, дата)

Такташкин Д.В.
(фамилия, инициалы)

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

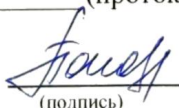
Заведующий кафедрой


(подпись)

Макарычев П.П.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от 15.06.21 № 10)

Секретарь ГЭК


(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ИПЕЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«21» января 2021 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Федоров Дмитрий Алексеевич гр. 17ВП1 факультета ВТ
(Фамилия, имя, отчество полностью)
2. Тема работы Разработка программных средств сбора и анализа
кардиологических данных

Тема утверждена приказом ПГУ № 0027 а/о от « 21 » января 2021 г.

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Попова Н.А.

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

1) *Назначение: программные средства предназначены для расчета риска развития ССЗ и сбора кардиологических данных для последующего анализа и построения новой математической модели расчета риска.*

- 2) *Среда выполнения: операционная система Android, начиная с версии Android 4.4.
Средства разработки: языки программирования Kotlin, Node.js.
Инструментальные средства: Android Studio, Microsoft Visual Studio Code
Технологии разработки: ООП, UML.*

- 3) *Основные функции программных средств сбора и анализа кардиологических данных:*

- а) сбор анамнеза, лабораторных и инструментальных данных;*
б) вывод оценки состояния здоровья согласно введенным лабораторным и инструментальным данным;
в) вывод рекомендаций для профилактических действий согласно полученной оценке;
г) оформление отчета по собранным системой данным.

5. Перечень подлежащих разработке вопросов

Анализ предметной области и постановка задачи

Проектирование программных средств

Реализация программных средств

Тестирование и анализ качества программного кода

Рефакторинг программных средств

План разработки и оценка бюджета

Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	14.04.2021 – 20.05.2021	Григорьев
2	План разработки и оценка бюджета	5%	21.04.2021 – 24.04.2021	Григорьев
3	Проектирование программных средств	25%	25.04.2021 – 15.05.2021	Григорьев
4	Реализация программных средств	30%	16.05.2021 – 30.05.2021	Григорьев
5	Тестирование и анализ качества программного кода	20%	01.06.2021 – 07.06.2021	Григорьев
6	Оформление пояснительной записки	10%	08.06.2021 – 10.06.2021	Григорьев

Дата выдачи задания « 28 » декабря 2020 г.

Руководитель бакалаврской работы

Григорьев Н.А.
(подпись, дата)

Задание к исполнению принял студент

Алексей Федоров Н.А.
(подпись, дата)

Работу к защите допустить

Декан факультета

(подпись, дата)

Реферат

Пояснительная записка содержит 124 страницы, 29 рисунков, 18 таблиц, 13 использованных источников и 4 приложения.

ПРОГРАММНЫЕ СРЕДСТВА, IDEF1X, MVVM, СБОР, АНАЛИЗ, КАРДИОЛОГИЯ, KOTLIN, UML.

Объект исследования – расчет риска заболевания сердечно-сосудистыми заболеваниями.

Цель выпускной квалификационной работы – разработка программных средств сбора кардиологических данных для последующего анализа и построения математической модели расчета риска заболевания сердечно-сосудистыми заболеваниями.

Технология разработки – ООП, UML.

Разработка серверной части проводилась на языке Node.js в среде Microsoft Visual Studio Code. Для разработки клиентской части применялись язык Kotlin и среда разработки Android Studio.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы программные средства для сбора кардиологических данных, состоящие из клиентской и серверной частей. Все требования к разработке программных средств соблюдены.

					ПГУ 09.03.04 – 08BP171.18 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Разработка программных средств сбора и анализа кардиологических данных. Пояснительная записка	Лист	Лист	Листов	
Разраб.		Федоров Д.А.		14.06.2				3	124
Проект		Попова Н.А.		14.06.2					
И. Контр.		Гаклашкин Д.В.		24.06.2					
Утверд		Макарычев П.П.							
					Группа 17ВП1				

Содержание

Введение	6
1 Анализ предметной области требований	8
1.1 Анализ предметной области.....	8
1.2 Постановка задач	10
1.3 Методы реализации.....	10
1.4 Анализ требований.....	12
1.4.1 Анализ функциональных требований	12
1.4.2 Анализ нефункциональных требований	14
1.5 Выбор инструментальных средства	15
2 Проектирование программных средств сбора и анализа кардиологических данных	16
2.1 Базовая архитектура	16
2.2 Построение концептуальной модели базы данных.....	17
2.3 Логическое проектирование базы данных	19
2.4 Проектирование мобильного приложения.....	22
3 Реализация программных средств	28
3.1 Физическая модель базы данных	28
3.2 Реализация серверной части.....	29
3.3 Реализация клиентской части.....	31
3.4 представление развертывания программных средств	32
4 Тестирование программных средств	33
4.1 Выбор режима тестирования программных средств	33
4.2 Структурное тестирование	33
4.2.1 Метод покрытия операторов	35
4.2.2 Метод покрытия условий.....	36
4.2.3 Метод покрытия маршрутов	37
4.2.4 Классы эквивалентности.....	38
4.2.5 Функциональная диаграмма	40
4.3 Тестирование серверной части приложения.....	44

	5
4.4 Метрики программного кода проекта	45
4.4.1 Цикломатическая сложность.....	46
4.4.2 Метрики сложности потока управления данными.....	47
5 Планирование и оценка бюджета проекта	48
Заключение.....	56
Список использованных источников.....	57
Приложение А. Глоссарий.....	59
Приложение Б. Результаты тестирования	61
Приложение В. Листинг серверной части.....	64
Приложение Г. Листинг клиентской части.....	97

Введение

Несмотря на высокий уровень развития медицины довольно сложно найти границу между состоянием человека. Именно поэтому контроль за состоянием здоровья человека должен вестись не только во время болезни, а постоянно. Помимо своевременного обнаружения заболевания, это позволит предсказать возможные заболевания на основании данных об индивидуальной предрасположенности человека и текущих показателях здоровья [1].

На сегодняшний день разрабатывается множество решений в области ИТ для удовлетворения следующих задач в области здравоохранения [2]:

- мониторинг и контроль физиологических параметров пациентов;
- поддержка мероприятий по первичной и вторичной профилактике заболеваний и их ранней диагностике;
- обеспечение пожилого населения качественным амбулаторным наблюдением;
- реабилитация пациентов, нуждающихся в психиатрической, психофизиологической и/или психологической помощи.

По итогам ряда клинических исследований, собранными в статье Наливаевой А.В., были выявлены положительные тенденции внедрения ИТ в практическом здравоохранении, что увеличивает необходимость глобального внедрения ИТ в медицине. Так можно отметить снижение смертности среди больных с сердечно-сосудистыми заболеваниями на 20 – 25% по сравнению с рутинной технологией организации медицинской помощи, т.е. без использования ИТ систем [3].

Использование подобных систем особенно актуально для сельских медицинских учреждений, где в последние годы наблюдается острая нехватка квалифицированных медицинских работников, в особенности врачей. В связи с сокращением больниц и фельдшерско-акушерских пунктов (ФАП) на селе нагрузка на медицинский персонал резко возрастает, что не позволяет фельдшерам и медсестрам уделять достаточно внимания каждому пациенту.

Действенную помощь профессионалам оказывают волонтеры: в рамках программы «#ДоброВСело» участники Всероссийского общественного движения «Волонтеры-медики» в 2019 году, например, осуществили 44 выезда на село, посетив 12 населенных пунктов четырех районов Пензенской области и осмотрев около полутора тысяч пациентов.

Однако волонтерского участия для оказания действительно эффективной помощи сельским жителям оказывается недостаточно, в первую очередь, в силу отсутствия у волонтеров необходимой квалификации. Как правило, это студенты-медики, которые могут измерить пациенту артериальное давление, провести антропометрию и т.п. Однако осуществить полноценное консультирование больного, назначить ему лечение и дать ему рекомендации они не могут в силу отсутствия необходимых знаний и опыта.

Актуальность решаемой задачи обуславливается широким распространением в нашей стране сердечно-сосудистых заболеваний. Практическая ценность разрабатываемой информационной системы (ИС) заключается в помощи врачу выбирать и ранжировать по важности рекомендации пациенту и методы профилактики ССЗ, опираясь на научно-обоснованные результаты.

1 Анализ предметной области требований

1.1 Анализ предметной области

На сегодняшний день сердечно-сосудистая патология занимает лидирующие позиции в заболеваемости и смертности населения мира. Так 32% смертей связаны с заболеваниями сердца, что составляет 18 млн. в абсолютном значении смертей в год. В России этот показатель достигает 46% (850 тыс. смертей). Наиболее уязвимым местом современного отечественного здравоохранения является проблема оказания помощи пациентам сельской области. Наряду с модернизацией и последующей оптимизацией городской медицины произошло радикальное отдаление от медицины сельской, сокращение количества фельдшерско-акушерских пунктов (ФАП) и обусловленный этим процессом рост охвата курируемого сельского населения средним и старшим медицинским персоналом. С 2005 по 2019 годы в регионах закрылось более 8 тыс. ФАПов и сельских амбулаторий, что составляет примерно 15% от общего их числа. Нехватка медицинских пунктов с одной стороны, и повышение нагрузки на существующий медицинский персонал с другой, влекут за собой снижение качества и доступности медицинской помощи в регионах. Зачастую пациент не может даже обратиться к врачу, не говоря о прохождении комплекса медицинского обследования и рационального лечения. Высокая нагрузка на фельдшеров и медсестер ФАП диктует им необходимость максимально сократить длительность приема и количество проводимых манипуляций.

Вопросам профилактики, диагностики и лечения кардиопатологии отводится ключевая роль в отечественном здравоохранении. Всероссийское общественное движение «Волонтеры-медики» принимает активное участие в борьбе с болезнями системы кровообращения, ежегодно проводя тренинги, квесты, дебаты, открытые диалоги с населением и иные спортивно-образовательные программы в рамках федеральной программы «Оберегая сердца». Одной из наиболее актуальных программ является акция «#ДоброВСело», в рамках которой волонтеры проводят обследование и

консультации по вопросам здоровья пациентов в труднодоступных сельских местностях. По данным социологического опроса среди населения районов Пензенской области, проведенного в 2019 году, лишь 14% пациентов удовлетворены оказываемой им медицинской помощью, в то время как почти 89% пациентов хотели бы знать больше о своем здоровье. В 2019 году волонтеры-медики совершили 44 выезда в районы Пензенской области для помощи сельским медикам. Однако, работая без врачей, волонтерам часто приходится сталкиваться с трудными случаями и самостоятельно определять объем и сроки медицинской помощи. К тому же, имеющийся багаж знаний волонтеров, которые в подавляющей своей массе состоят из студентов медицинских ВУЗов, оказывается слишком мал для полноценной врачебной консультации и дачи рекомендаций. Поэтому, в рамках настоящего проекта предлагается разработать программные средства, которые на основании имеющихся у пациентов факторов риска, лабораторных и инструментальных данных, рассчитывают риск развития трех сердечно-сосудистых заболеваний: ишемической болезни сердца, нарушений ритма сердца, гипертонической болезни. Полученные процентные величины риска градируются на низкий, средний и высокий, что позволяет провести качественную стратификацию объема и сроков оказания квалифицированной врачебной помощи каждому пациенту. Помимо этого, в рамках проекта планируется апробировать полученные программные средства путем телемедицинских консультаций с врачом-специалистом для верификации получаемых показателей.

Таким образом, предлагаемый проект позволит повлиять на проблему оказания помощи пациентам в сельской местности, снизив тем самым заболеваемость и увеличив доступность медицинской помощи. Проект внесет большой вклад в развитие Всероссийского проекта по улучшению условий функционирования фельдшерско-акушерских пунктов «#ДоброВСело», а также в Федеральную программу по профилактике сердечно-сосудистых заболеваний и их осложнений «Оберегая сердца», в реализации которых участвует Всероссийское общественное движение «Волонтеры-медики».

1.2 Постановка задач

В ходе изучения предметной области была поставлена следующая цель данной квалификационной работы: обеспечение удобного инструмента для скрининга сердечно-сосудистых заболеваний для использования в сельской местности в рамках акции «#ДоброВСело» волонтерами-медиками.

Для достижения поставленной цели разрабатываемые программные средства должны обеспечивать сбор анамнестических, лабораторных и инструментальных данных пациента для составления тестового доврачебного отчета о здоровье пациента. Собранные данные необходимо хранить и предоставлять в удобной форме для последующего анализа и выявления предикторов возникновения сердечно-сосудистых заболеваний, а также для анализа эпидемиологического состояния региона.

На основе анализа предметной области и поставленных перед данной квалификационной работой цели была поставлена задача разработка программных средств для:

- сбора анамнестических, лабораторных и инструментальных данных в условиях низкой медицинской обеспеченности;
- предоставления отчета, содержащий качественные (наличие/отсутствие) и количественные (выраженные в условных единицах) показатели отобранных коррелятов ССЗ;
- социологического опроса влияния проекта на развитие медицинского волонтерства в Пензенской области путем анкетирования удовлетворенности населения медицинской помощью.

Разрабатываемые программные средства должны выполнять все поставленные перед ними задачи, быть удобным в использовании и поддерживать как можно большее количество устройств.

1.3 Методы реализации

Для решения поставленных задач были использованы следующие методы реализации:

- сбор медицинских данных из выписок историй болезни пациентов кардиологических отделений стационаров г. Пенза;
- проведение статистического анализа собранных данных для выявления предикторов возникновения сердечно-сосудистых заболеваний;
- построение шкалы оценки риска возникновения сердечно-сосудистых заболеваний на основе имеющихся факторов риска и их отношения шансов.

На первом этапе будет проведен ретроспективный анализ историй болезни пациентов кардиологических отделений следующих стационаров г. Пенза: ГБУЗ «Пензенская областная клиническая больница им. Н.Н. Бурденко», ГБУЗ «Клиническая больница №6 им. Г.А. Захарьина». В анализируемые показатели входят: пол, возраст, диагноз, факторы риска, лабораторные и инструментальные данные. Предполагаемый объем данных составляет 1000 историй болезни, что соответствует 6 – 9 месяцам работы указанных подразделений и является достаточной статистической выборкой для получения достоверных результатов.

На втором этапе будет проведен корреляционный и регрессионный анализ имеющихся данных. По результатам корреляционного анализа будут отобраны факторы, имеющие наибольшую положительную связь ($r > 0.5$) в развитии следующих сердечно-сосудистых заболеваний (ССЗ): ишемическая болезнь сердца (ИБС), нарушения ритма сердца (НРС), гипертоническая болезнь (ГБ). По результатам анализа планируется отобрать не более 15 коррелятов для каждого заболевания. Регрессионный анализ позволит оценить вклад конкретного признака в развитии заболевания и составить уравнение регрессии вида:

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot x_3 + e,$$

которое будет использовано в разрабатываемых программных средствах.

На последнем этапе для расчета вклада каждого изучаемого признака в когортных группах пациентов с ССЗ будет использован метод отношения шансов (ОШ), который позволяет оценить связь между определенным

исходом и фактором риска в качественном и количественном плане, что необходимо для построения шкалы прогнозирования риска развития заболевания в разрабатываемых программных средствах.

1.4 Анализ требований

1.4.1 Анализ функциональных требований

На основе проведенного анализа были выделены следующие основные функциональные требования программных средств сбора и анализа кардиологических данных:

- сбор анамнестических, инструментальных и лабораторных данных;
- возможность сделать или выбрать снимок, который прилагается к инструментальным или лабораторным данным;
- аутентификация с заранее созданной базой пользователей;
- вывод отчета, содержащий качественные (наличие/отсутствие) и количественные (выраженные в условных единицах) показатели отобранных коррелятов ССЗ;
- вывод вместе с отчетом рекомендательных действий по профилактике здоровья пациента на основе качественных и количественных показателей;
- опрос удовлетворенности работой волонтеров-медиков и результатом работы программных средств;
- вывод отчета по собранным анамнестическим, инструментальным и лабораторным данным в электронной таблице формата MS Excel для дальнейшей статистической обработки.

Для более полного понимания функционирования системы была спроектирована диаграмма вариантов использования на основе функциональных требований. Диаграмма представлена на рисунке 1.

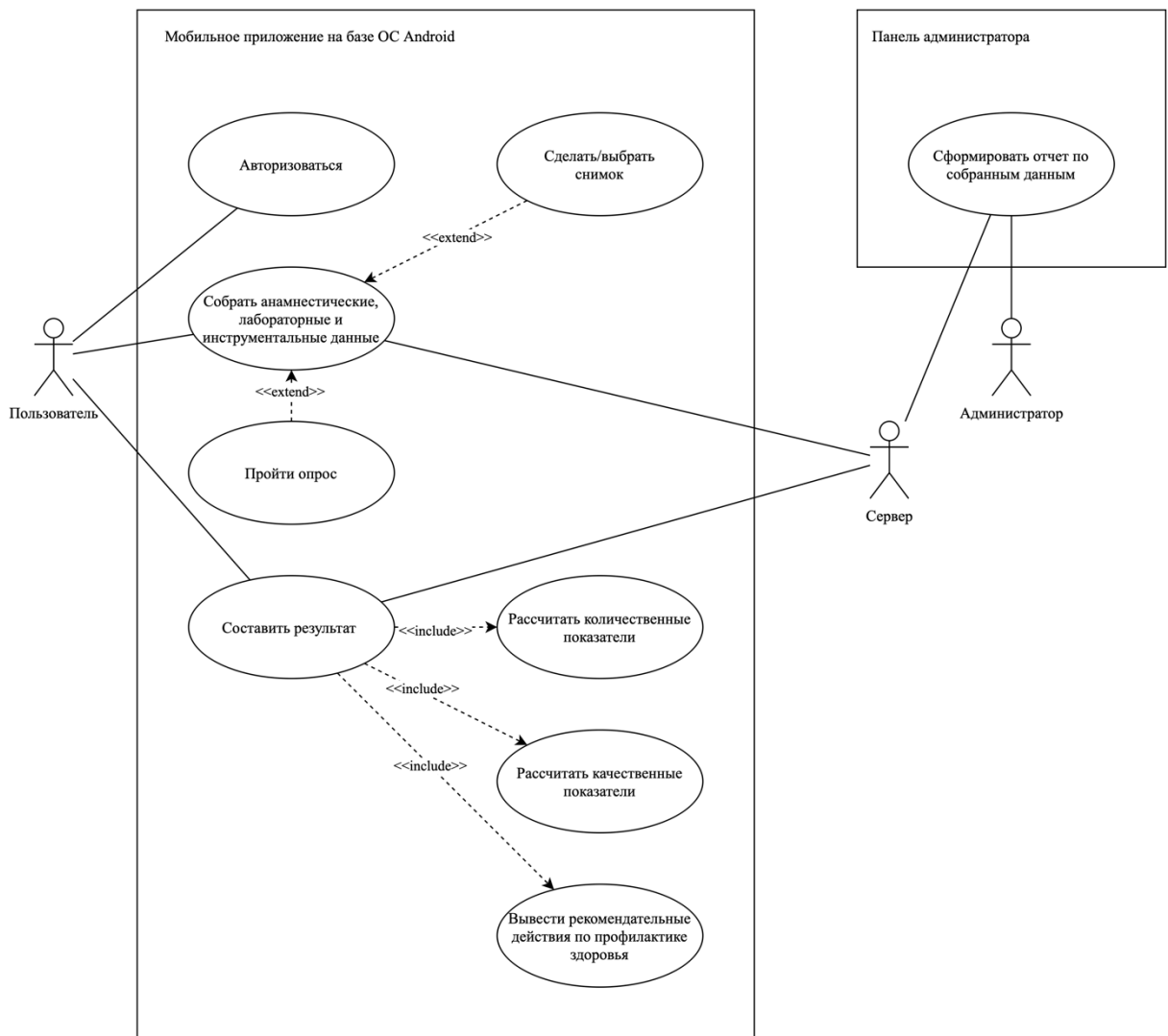


Рисунок 1 – Диаграмма вариантов использования

Диаграмма отражает отношения между вариантами использования и актерами и описывает систему на концептуальном уровне. На данной диаграмме большой прямоугольник обозначает границу ответственности мобильного приложения. Именно мобильное приложение на базе ОС Android является основным инструментом для сбора анамнестических, лабораторных и инструментальных данных. На этапе составления технического задания выбор данной версии позволял охватить 98.1% мобильных устройств с данной операционной системой.

Для доступа к собранным данным было выбрано использовать статических сайт, функционала которого достаточно для работы на всех современных браузерах.

1.4.2 Анализ нефункциональных требований

На основе анализа предметной области были выделены следующие нефункциональные требования к программным средствам:

- интерфейс пользователя;
- требования к производительности;
- требования к аппаратно-программному окружению.

Разрабатываемые программные средства для сбора и анализа кардиологических данных должны поддерживать стабильную работу при наличии установленного с сервером соединения, особенно важно обеспечивать бесперебойную работу при одновременном доступе к серверу нескольких пользователей системы.

Мобильное приложение, входящее в состав разрабатываемых программных средств, должно обладать простым и удобным для пользователя графическим интерфейсом, позволяющий использовать все ключевые функции системы. Выбор одной ОС для мобильного приложения возник из-за большой доли Android на рынке мобильных операционных систем, что составляет 74,43%. Помимо этого, разработка приложений под эту систему не привязана к определенной экосистеме, как, например, разработка под iOS (тестирование приложений доступно только под ОС macOS). Минимальные системные требования для корректной работы клиентского мобильного приложения:

- ОС: Android версии 4.4 и выше;
- наличие встроенной видеокамеры;
- оперативная память (ОП): не менее 512 Мб.

Технические средства для сервера не имеют определенных требований, т.к. в качестве серверной части было выбрано использование облачных сервисов Google Firebase. Единственное требование – минимизация затрат при использовании сервиса по подписке Blaze, которая предлагает минимальные ограничения для бесплатного использования.

1.5 Выбор инструментальных средства

Разработка программных средств сбора и анализа кардиологических данных будет проводиться и использованием проектирования разнообразных моделей, построенных на унифицированном языке моделирования UML, который смог объединить в себе наилучшие приемы разработки и моделирования ПО. UML позволяет посмотреть на систему и ее поведение практически со всех аспектов [4]. Для проектирования диаграмм и использованием языка UML было выбрано CASE-средство draw.io.

Для разработки мобильного приложения был выбран язык объектно-ориентированного программирования (ООП) Kotlin, который поддерживается большим многообразием инструментов, надежен благодаря обработке исключений и безопасности типов. Данный язык прекрасно подходит для быстрой и надежной разработки клиент-серверного приложения. В качестве средства для разработки была выбрана IDE Android Studio.

В качестве серверной части было решено воспользоваться возможностями облачных технологий Google Firebase, а именно сервисом Firebase Functions, реализующий свою работу на программной платформе Node.js с использованием языка программирования JavaScript. Интегрированная среда разработки – Microsoft Visual Studio Code.

Для управления проектами был выбран Microsoft Project 2018, созданный для планирования и оптимизации проекта. Главными функциями данного продукта являются анализ объема работ, распределения ресурсов оп задачам, а также наглядная демонстрация процесса разработки на сетевом графе и диаграмме Ганта [5].

2 Проектирование программных средств сбора и анализа кардиологических данных

2.1 Базовая архитектура

Разрабатываемые программные средства сбора и анализа кардиологических имеют клиент-серверную архитектуру [6], которую схематично можно представить на рисунке 2.

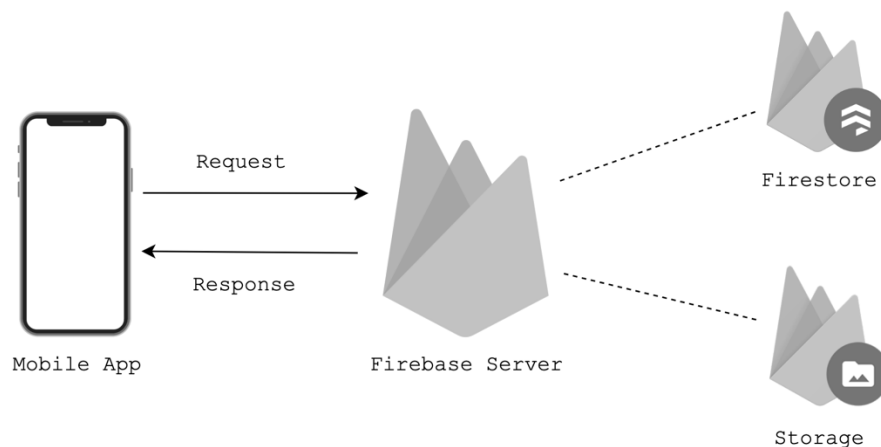


Рисунок 2 – Клиент-серверная архитектура мобильного приложения

Основными компонентами разрабатываемой системы являются:

- мобильное приложение на базе ОС Android;
- сервер, работающий на Firebase Functions;
- документно-ориентированная база данных Firebase Firestore;
- хранилище файлов Firebase Storage.

Мобильное приложение в данной архитектуре выполняет функции сбора анамнестических, инструментальных и лабораторных данных. Отчет с качественными и количественными показателями формируется на стороне сервера. Технически это можно представить как процесс обмена запросами и ответами между сервером на Firebase Functions и мобильным приложением.

Выбор сервиса Firebase Functions для обработки поступающих запросов от пользователя был сделан по ряду причин. Вот некоторые из них [7]:

- скорость разработки;
- тарифный план по низким ценам;

- сквозная платформа для разработки приложений;
- не требует развертывания сервера;
- мониторинг ошибок;
- безопасность.

Данный сервис использует готовое API для работы с базой данных и имеет ограниченную кастомизацию сервера. Также он предлагает готовый к настройке и использованию процесс аутентификации [8].

База данных построена на документно-ориентированном подходе, схожим с JSON-форматом представления данных, использующем относительно новую парадигму проектирования NoSQL. Она направлена на решение проблем масштабируемости и доступности за счет согласованности и атомарности данных.

Хранилище Firebase Storage необходимо для хранения фотографий, используемых для лабораторных данных. Оно имеет древовидную структуру, подобную типичным файловым системам.

2.2 Построение концептуальной модели базы данных

Основная задача концептуальной модели – передать фундаментальные принципы и основные функциональные возможности системы, которую она представляет. Кроме того, концептуальная модель должна быть разработана таким образом, чтобы обеспечить легко понятную системную интерпретацию для пользователей модели.

Прежде чем построить концептуальную модель первым делом необходимо выделить сущности предметной области, их атрибуты и отношения.

Построение концептуальной модели включает в себя описание типов сущностей и описание типов связей между типами сущностей. Данные описания представлены в таблице 1 и таблице 2 соответственно.

Таблица 1 – Описание типов сущностей

№	Наименование	Описание
1	Волонтер	Пользователь системы, заполняющий лабораторные и инструментальные данные пациента
2	Запись	Лабораторные и инструментальные данные пациента, полученные в ходе беседы последнего с волонтером
3	Атрибут	Составленные заранее факторы риска, заполненные волонтером (анамнез, лабораторные и инструментальные данные)
4	Фотография	Объекты, описывающие местоположение фотографии на диске

Таблица 2 – Описание типов связей между типами сущностей

№	Тип сущности 1	Тип сущности 2	Описание	Модность	Обязательность
1	Волонтер	Запись	Волонтер составил запись	1:1	Обязательно
2	Запись	Атрибут	Запись содержит некоторые атрибуты	1:M	Обязательно
3	Атрибут	Фотография	Заполненный атрибут содержит некоторые массив фотографий	0:M	Необязательно

В ходе анализа научной литературы, имеющих теоретических положений, структурирования объекта, четкой интерпретации основных понятий, была спроектировано графическое представление концептуальной модели базы данных в виде диаграммы «сущность-связь» для проектируемых программных средств. Данная модель представлена на рисунке 3.

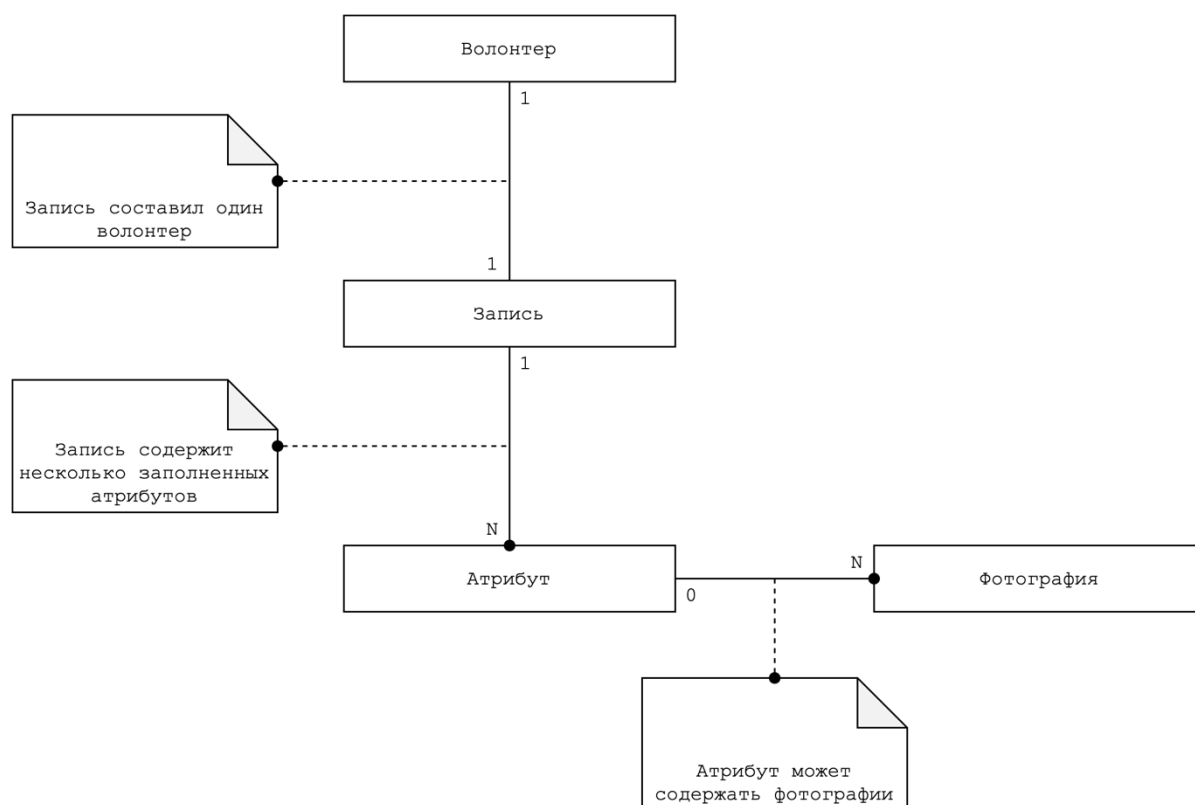


Рисунок 3 – Концептуальная модель базы данных

В результате построения концептуальной модели были получены описаны атрибуты у следующих сущностей: «Волонтер», «Запись», «Атрибут», «Фотография».

2.3 Логическое проектирование базы данных

Основная цель, которую стремятся получить при проектировании логической модели, состоит в получении графического представления логической структуры базы данных. Логическая модель предметной области отражает сущности и взаимоотношения между ними. При проектировании логической модели стоит абстрагироваться от требований платформы и языка реализации или способа дальнейшего использования данных.

На основе концептуальной модели данных была разработана логическая модель. Она представлена на рисунке 4.

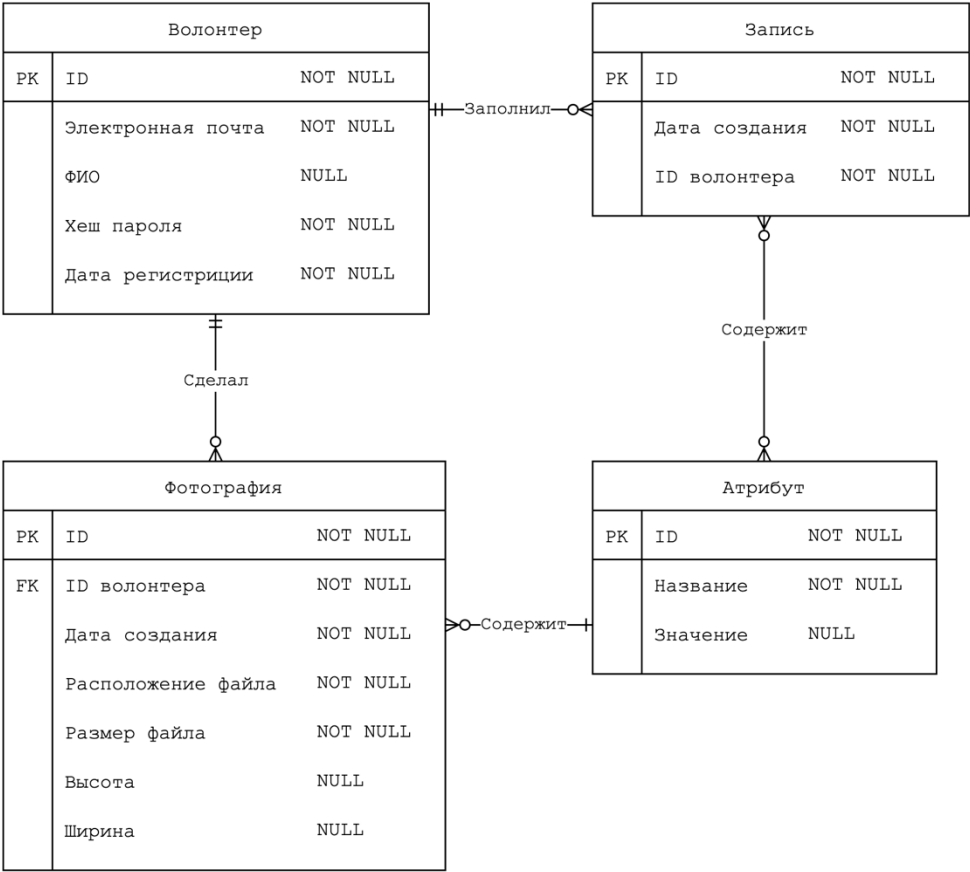


Рисунок 4 – Логическая модель

Описание атрибутов сущностей логической модели представлено в таблицах 3 – 6.

Таблица 3 – Описание атрибутов сущности «Волонтер»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
ID	Идентификатор сущности	PK	—
Электронная почта	Электронная почта волонтера (необходима для авторизации)	—	—
ФИО	ФИО волонтера	—	—
Хеш пароля	Результат хеш-функции, входным параметром которой является пароль	—	—
Дата регистрации	Дата создания учетной записи волонтера в системе	—	—

Таблица 4 – Описание атрибутов сущности «Фотография»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
ID	Идентификатор сущности	РК	—
ID волонтера	Идентификатор волонтера, создавшего фотографию	—	FK
Дата создания	Дата загрузки фотографии на сервер	—	—
Расположение файла	Абсолютный путь файла с фотографией	—	—
Размер файла	Размер файла в байтах	—	—
Высота	Высота изображения в пикселях	—	—
Ширина	Ширина изображения в пикселях	—	—

Таблица 5 – Описание атрибутов сущности «Запись»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
ID	Идентификатор сущности	РК	—
Дата создания	Дата выгрузки результата обследования на сервер	—	—
ID волонтера	Идентификатор волонтера, проведшего обследование	—	—

Таблица 6 – Описание атрибутов сущности «Атрибут»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
ID	Идентификатор сущности	РК	—
Название	Название атрибута	—	—
Значение	Введенное волонтером значение атрибута	—	—

Таким образом, была разработана логическая модель базы данных для разрабатываемых программных средств сбора кардиологических данных.

2.4 Проектирование мобильного приложения

На основе функциональных требований спроектирован пользовательский интерфейс мобильного приложения. Он включает в себя следующие экраны:

- экран для авторизации пользователя системы;
- экран с атрибутами;
- экран с результатом в виде ряда оценок здоровья пациента, а также рекомендации для профилактики здоровья на основе этих оценок;
- экран для оценки результата работы приложения.

На рисунке 5 изображен экран для авторизации пользователя системы. Данный экран открывается при запуске приложения, если пользователь не был ранее авторизован.

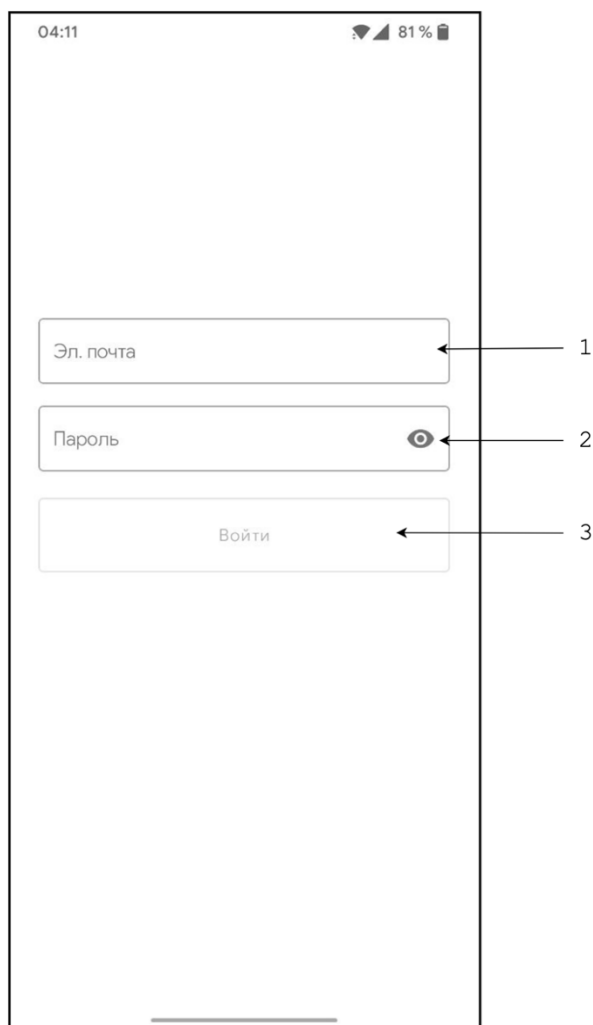


Рисунок 5 – Экран авторизации

Рассмотрим основные элементы интерфейса данного экрана авторизации, их несколько:

- 1 – текстовое поле для ввода адреса электронной почты;
- 2 – текстовое поле для ввода пароля;
- 3 – кнопка для авторизации.

На рисунке 6 представлен главный экран – экран с атрибутами, где пользователю необходимо вводить необходимые лабораторные и инструментальные данные. Из-за большого разнообразия атрибутов было принято решение воспользоваться одним из элементов популярного стиля графического дизайна интерфейсов мобильных приложения под Android под названием «Backdrop», позволяющего разделить логически заполнение атрибутов на несколько шагов, тем самым упростив навигацию для конечного пользователя [9].

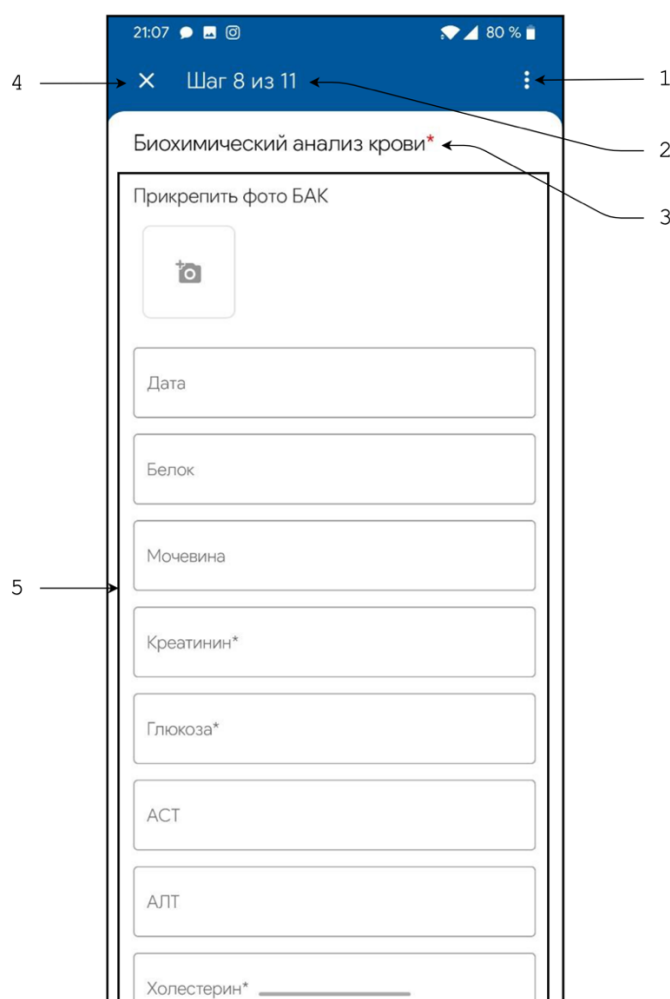


Рисунок 6 – Главный экран с атрибутами

Основные элементы главного экрана:

- 1 – кнопка для вызова контекстного меню (рисунок 7)
- 2 – номер текущего шага;
- 3 – название текущего шага;
- 4 – кнопка для открытия навигационной панели (рисунок 8);
- 5 – основное содержание шага, где находятся атрибуты для заполнения.

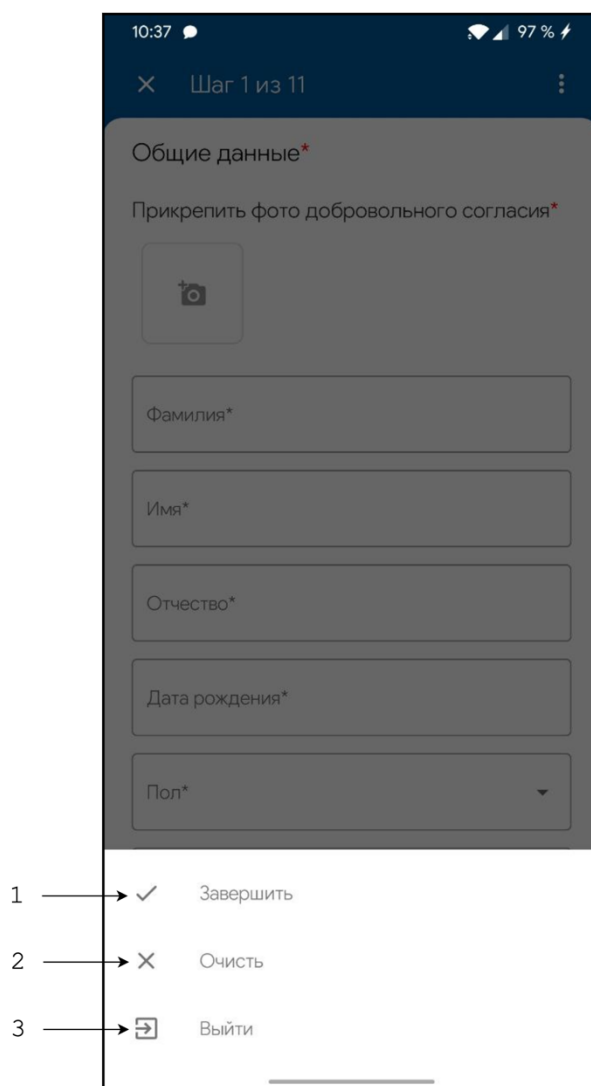


Рисунок 7 – Контекстное меню главного экрана

Основные элементы контекстного меню:

- 1 – кнопка для завершения отчета и получения результата;
- 2 – кнопка для очистки главного экрана от введенных значений;
- 3 – кнопка для выхода пользователя (перенаправляет на экран авторизации).

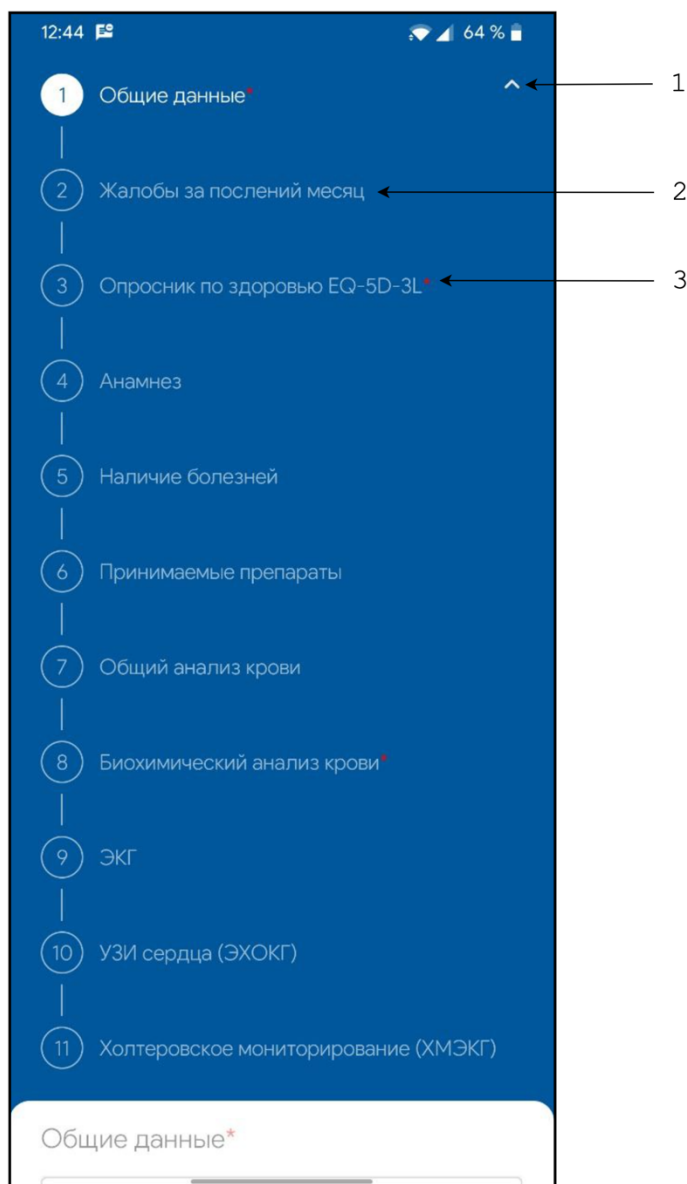


Рисунок 8 – Навигационная панель главного экрана

Основные элементы представленной выше навигационной панели главного экрана:

- 1 – кнопка для закрытия навигационной панели;
- 2 – название одного из шагов;
- 3 – красная звездочка, указывающая на наличие обязательных для заполнения атрибутов.

На рисунке 9 представлен экран с результатами отчета. Он содержит процент риска смерти от ССЗ по шкале SCORE, скорость клубочковой

фильтрации по шкале CKD-EPI и индекс массы тела, а также описание значений с рекомендациями согласно данному описанию.

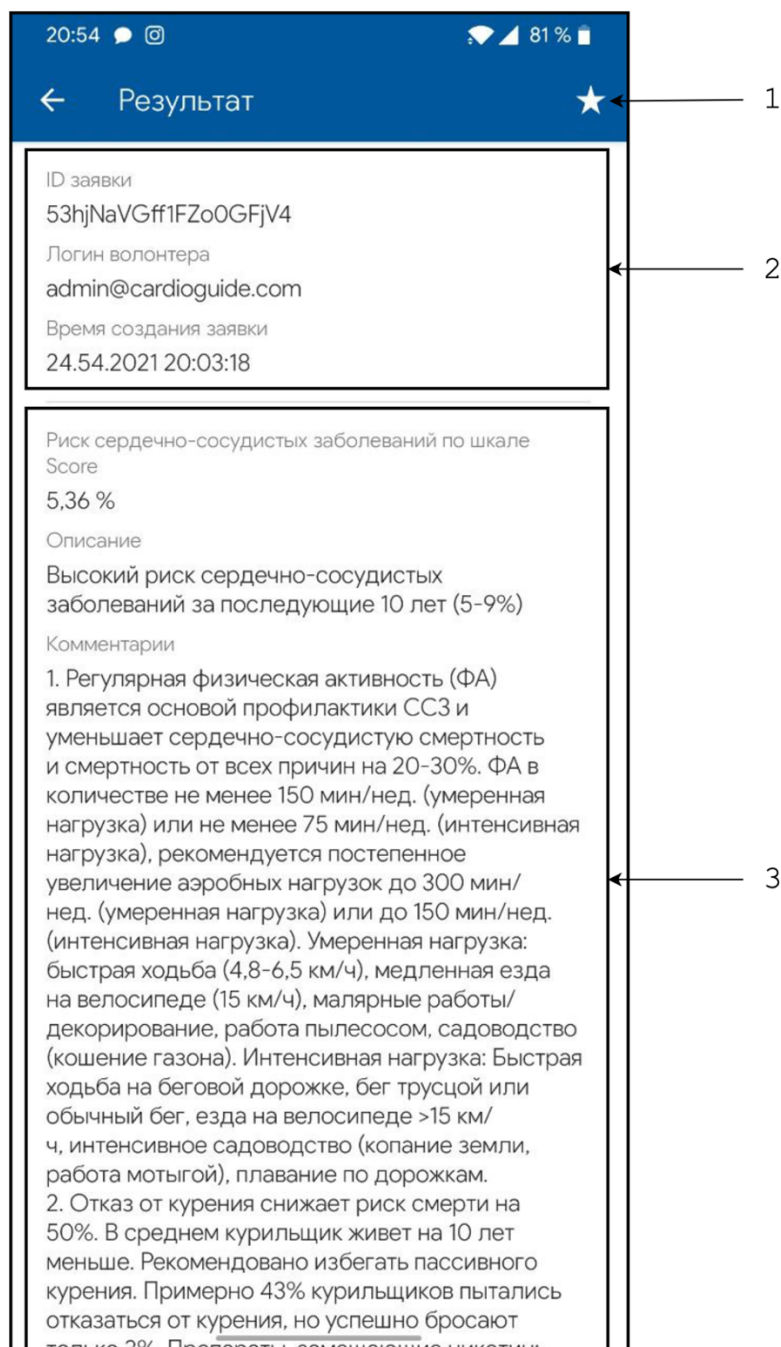


Рисунок 9 – Экран с результатом

Основными элементами экран с результатом являются:

- 1 – кнопка для оценки отчета;
- 2 – служебная информация данного отчета;
- 3 – основные результаты отчета о здоровье пациента.

На рисунке 10 представлен экран для оценки результатов отчета.

10:12 100 %

← Оценить

Знали ли вы о рекомендациях, которые вам дало наше приложение?

☐ Да

☒ Нет

Были ли для вас полезны данные рекомендации?

☒ Да

☐ Нет

Пожалуйста, оцените приложение от 1 до 5

☐ 1

☐ 2

☐ 3

☐ 4

☒ 5

Отправить

1

2

Рисунок 10 – Экран оценки результатов отчета

Основными элементами экрана оценки результатов отчета являются:

- 1 – основные вопросы оценки;
- 2 – кнопка для отправки оценки.

Таким образом, был спроектирован пользовательский интерфейс мобильного приложения. В процессе создания пользовательских элементов были учтены требования конечного пользователя, а также соблюдены основные правила построения графического интерфейса [10].

3 Реализация программных средств

3.1 Физическая модель базы данных

Документно-ориентированный подход, гибкий, иерархический характер документов и документных баз данных позволяет им развиваться в соответствии с потребностями приложений. Основная структура подобных баз данных представлена на рисунке 11.

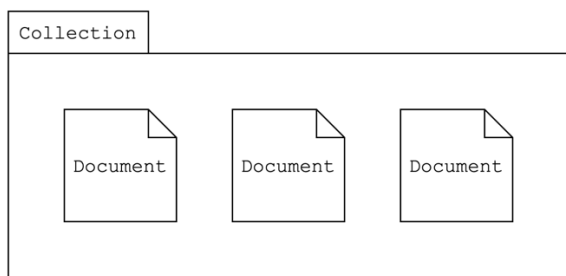


Рисунок 11 – Документная база данных

Для реализации базы данных было принято решение воспользоваться сервисом Firebase Firestore, реализующий документную базу данных. На рисунке 12 представлены заполненная тестовыми данными документно-ориентированная база данных на сервисе Firebase Firestore.

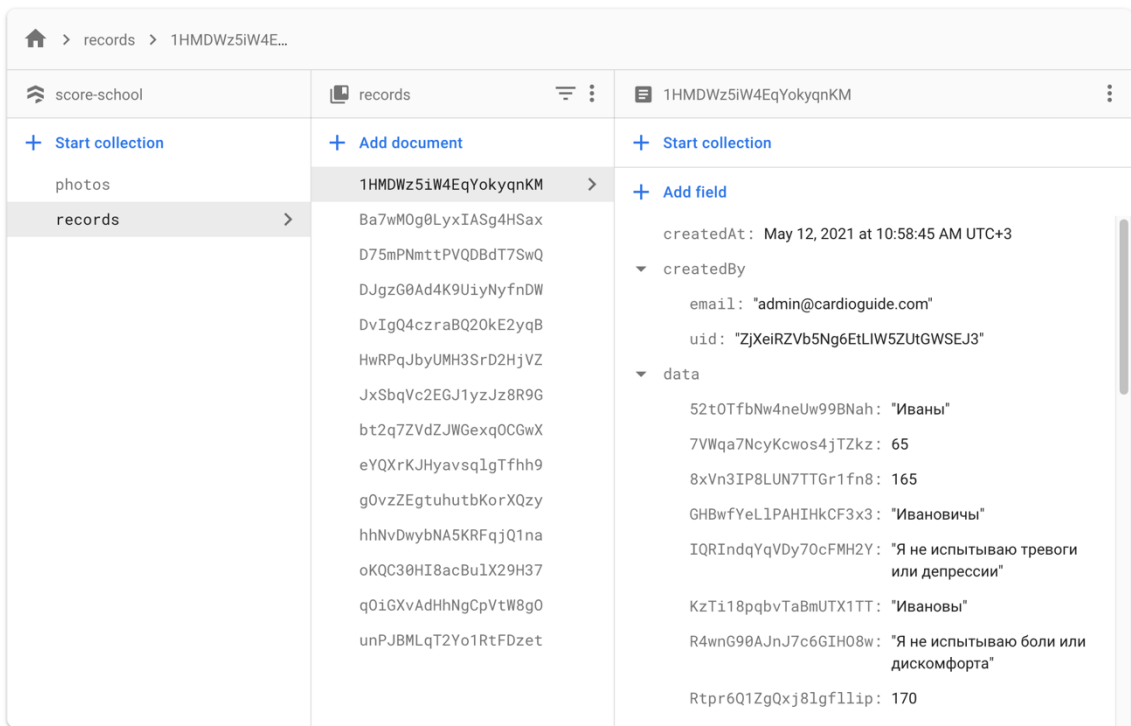
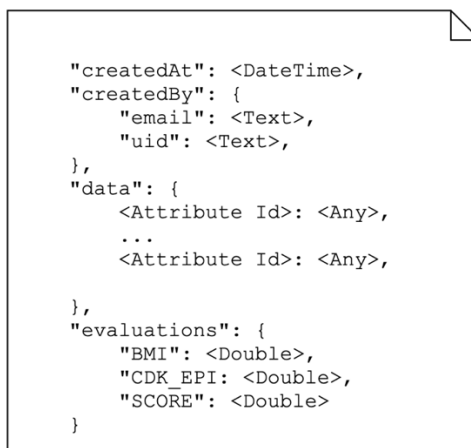


Рисунок 12 – Тестовые данные базы данных

Для представления иерархических структур было решено воспользоваться форматом JSON, как основоположником документно-ориентированного подхода. Структура документа, представляющая отчет, представлена на рисунке 13.



```
"createdAt": <DateTime>,
"createdBy": {
  "email": <Text>,
  "uid": <Text>,
},
"data": {
  <Attribute Id>: <Any>,
  ...
  <Attribute Id>: <Any>,
},
"evaluations": {
  "BMI": <Double>,
  "CDK_EPI": <Double>,
  "SCORE": <Double>
}
```

Рисунок 13 – Структура документа коллекции records

Как видно из рисунка, представленного выше, главные данные, собираемые пользователями программных средств в результате личного общения с пациентом, содержатся в поле «data», в котором собраны данные на введенные атрибуты. Именно там собраны все инструментальные, лабораторные и анамнестические данные одного пациента.

3.2 Реализация серверной части

Так как для разработки серверной части было принято решение использовать сервисы Firebase, основным языком стал Node.js, для которого имеются все нужные зависимости для работы с сервисом Firebase Functions.

Firebase Functions позволяет разворачивать код в виде функций в отказоустойчивом, безопасном и автоматически масштабируемом окружении без создания и обслуживания аппаратных или виртуальных машин. Основным преимуществом выбора данного сервиса является:

- бессерверные вычисления, благодаря которым возможен запуск приложения без создания виртуальных машин;

- гибкое масштабирование, для которого характерно, например, автоматическое увеличение дополнительных экземпляров функции при увеличении количества вызовов;
- наличие триггеров, с помощью которых возможна настройка интеграции между функциями Firebase Functions и другими сервисами Firebase без написания интеграционного кода;
- мониторинг ошибок, позволяющий быстро отладить и протестировать программный продукт.

На рисунке 14 представлена диаграмма для основных компонентов серверной части разрабатываемых программных средств. Она отражает общую структуру исходного кода приложения и показывает взаимосвязи между компонентами, которые представляют из себя физические части системы, обеспечивающие реализацию классов и отношений.

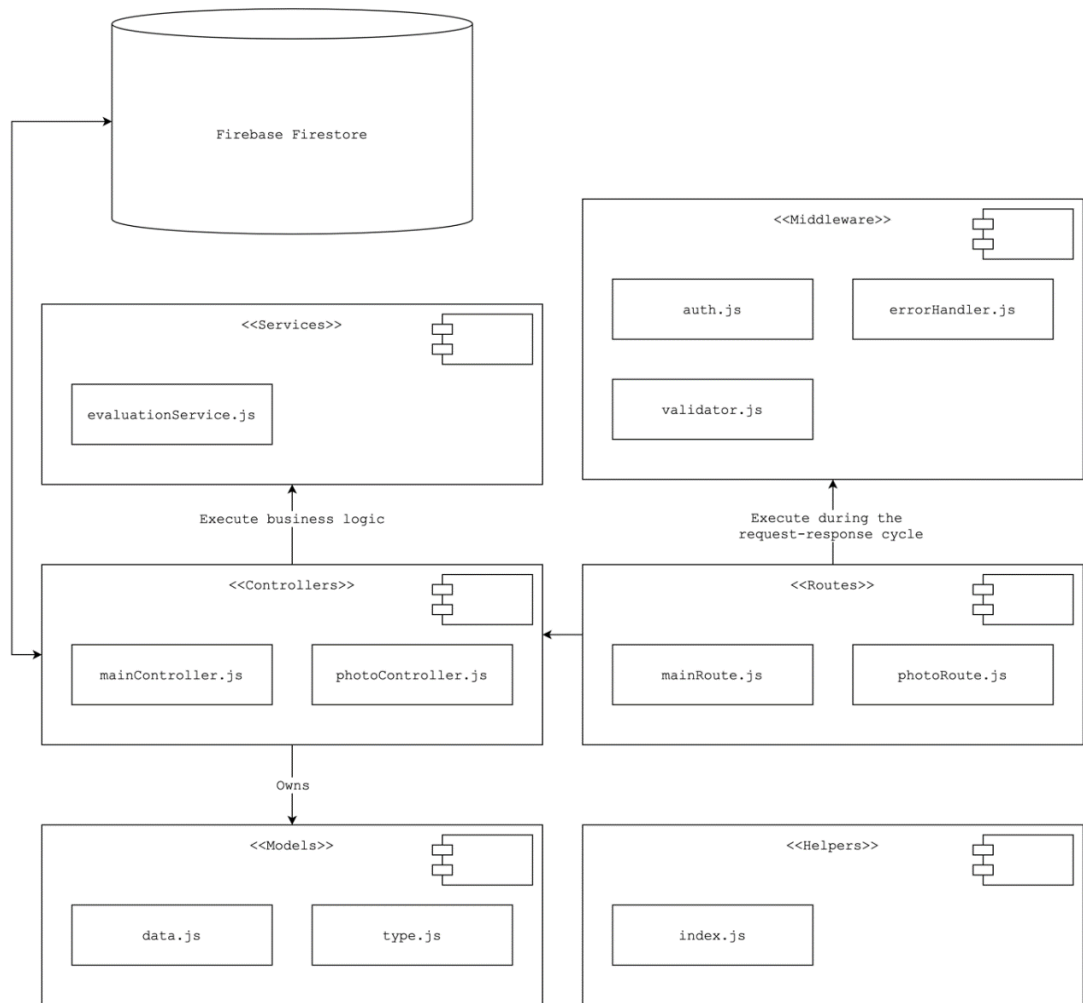


Рисунок 14 – Диаграмма компонентов серверной части

Диаграмма отражает распределение компонентов по архитектурным слоям, их взаимодействие друг с другом. Здесь компонент «Controllers» взаимодействует с базой данных посредством API, предоставляемый Firebase.

Для обеспечения безопасности данных, Firebase предоставляет гибкую настройку уровней доступа посредством правил. Для разрабатываемых программных средств было целесообразным полностью закрыть доступ кроме сервиса Firebase Functions.

Листинг серверной части представлен в приложении Б.

3.3 Реализация клиентской части

Разработка клиентской части производилась для ОС Android на языке программирования Kotlin в среде разработки Android Studio.

Диаграмма компонентов для разрабатываемого мобильного приложения представлена на рисунке 15. На ней отображены лишь основные компоненты и основные файлы классов этих компонентов, которые определяют архитектуру приложения.

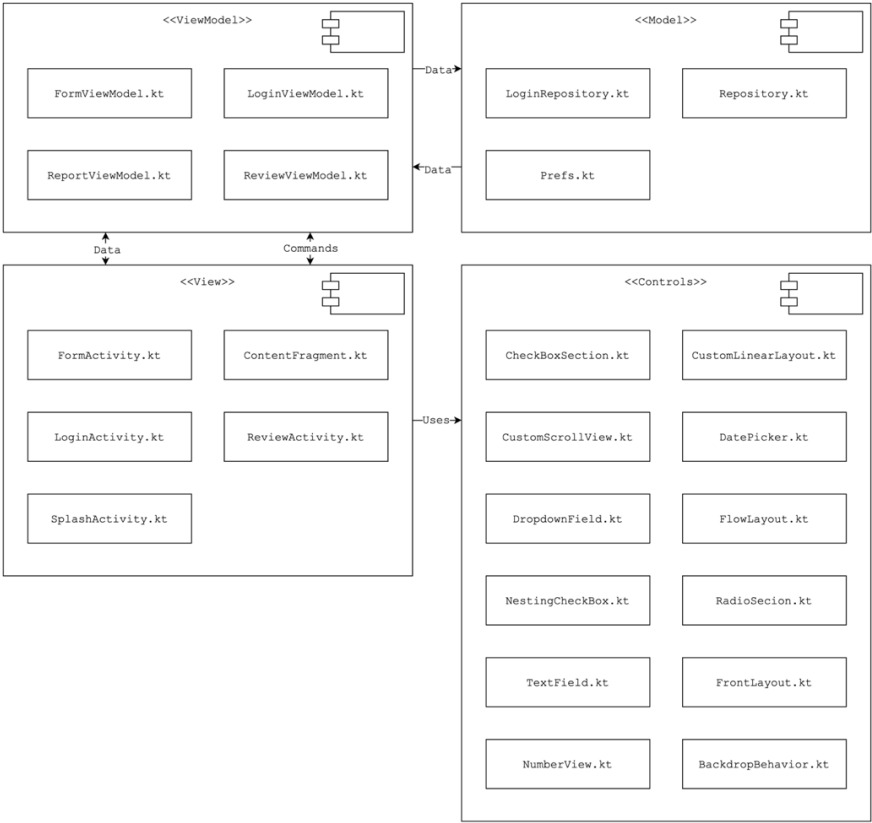


Рисунок 15 – Диаграмма компонентов мобильного приложения

Как видно из данной диаграммы, базовая архитектура построена на MVVM (Model-View-Presenter). Также стоит заметить большое разнообразие собственных представлений, указанных в компоненте «Controls». Данный шаг был предпринят в целях разделения функционала для более удобного понимания работы программного кода.

Листинг программного кода мобильного приложения представлен в приложении В.

3.4 представление развертывания программных средств

Для отражения программных средств как единой системы со всеми устройствами, процессами и связями между ними существует диаграмма развертывания, которая всецело отражает особенности ее реализации [11].

Диаграмма развертывания программных средств сбора и анализа кардиологических данных представлена на рисунке 16.

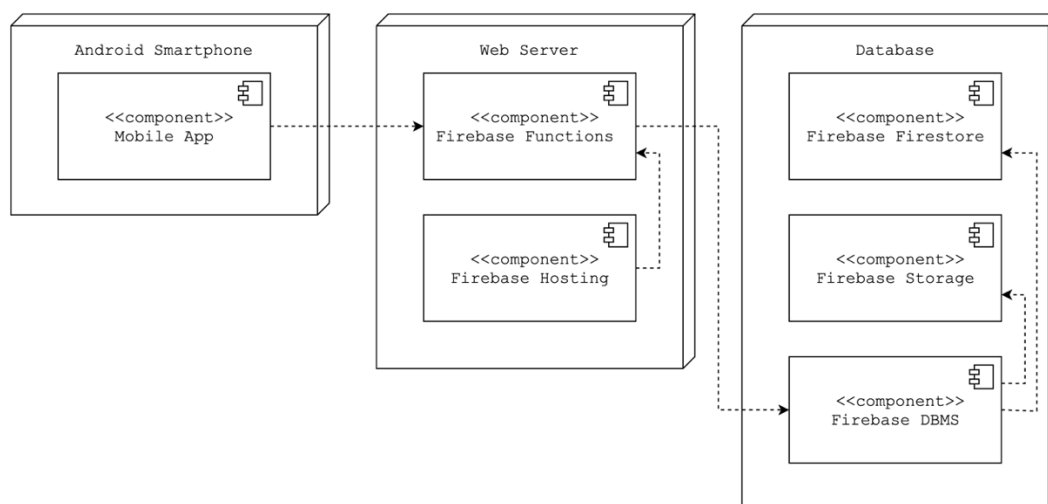


Рисунок 16 – Диаграмма развертывания

Как видно из указанной диаграммы, система состоит из трех основных частей: мобильного телефона на ОС Android, сервера на Firebase и базы данных на Firebase Firestore и Firebase Storage. Сервис Firebase Hosting используется для панели администрирования, где только пользователь с правами администратора может получить отчет о всех собранных данных в формате excel.

4 Тестирование программных средств

4.1 Выбор режима тестирования программных средств

Существует достаточно много видов тестирования программного продукта. Были рассмотрены следующие виды тестирования [12]:

- функциональное тестирование;
- нефункциональное тестирование;
- модульное тестирование.

Функциональное тестирование создано для проверки реализуемости функциональных требований, или способности разработанного продукта в определенных условиях решать задачи, нужные пользователям. Функциональное тестирование на сегодняшний день является одним из наиболее часто применяемых видов тестирования, так как позволяет наиболее полно проверить программу на соответствие требованиям.

Нефункциональное тестирование позволяет проверить соответствие свойств ПО с поставленными нефункциональными требованиями.

Модульное тестирование – тестирование отдельных частей сложной программной системы, или модулей для того, чтобы удостовериться в корректной работе всей системы, которую определяют данные модули [13]. Если во время тестирования системы возникли проблемы, данный метод позволяет проще выявить те модули, которые вызвали эти проблемы, и устранить соответствующие ошибки в них.

На основе рассмотренных инструментов тестирования было принято решение осуществить проверку работы расчета показателей здоровья при формировании отчета с помощью модульного тестирования. Такой выбор обусловлен сложностью данной подсистемы.

4.2 Структурное тестирование

Для структурного тестирования характерно детальное изучение логики программы, а также разработка тестов, которые позволяют проверить максимальное число логических ветвлений и операторов условий. Для

данного типа тестирования характерно максимально полный разбор всех маршрутов программы.

В качестве тестируемого алгоритма возьмем функцию переключения фрагмента с представлениями. Листинг данной функции представлен в листинге 1.

Листинг 1 – Функция тестируемого алгоритма

```
private fun showPage(index: Int) {
    val transaction = supportFragmentManager.beginTransaction()
    transaction.setCustomAnimations(R.animator.fade_in,
    R.animator.fade_out)

    val currentFragment =
supportFragmentManager.primaryNavigationFragment
    var fragment =
supportFragmentManager.findFragmentByTag(FRAGMENT_TAG_PREFIX + index)
    if (currentFragment != null && (fragment == null ||
currentFragment != fragment)) {
        transaction.detach(currentFragment)
    }
    if (fragment == null) {
        fragment =
ContentFragment.newInstance(viewModel.selectedStep.value ?: 0)
        transaction.add(FRAGMENT_CONTAINER, fragment,
FRAGMENT_TAG_PREFIX + index)
    } else {
        transaction.attach(fragment)
    }

    transaction.setPrimaryNavigationFragment(fragment)
    transaction.commitNow()
}
```

Блок-схема алгоритма представлена на рисунке 17. Основная задача данного алгоритма состоит в переключении содержимого главного экрана. Так как при разработке программных средств появилась необходимость в заполнении большого количества входных данных, было решено разделить экран для ввода данных на несколько логических шагов, между которыми необходимо переключаться. Каждый шаг имеет индекс, который передается данной функции, которая в свою очередь переключает содержимое на фрагмент, передавая ему индекс.

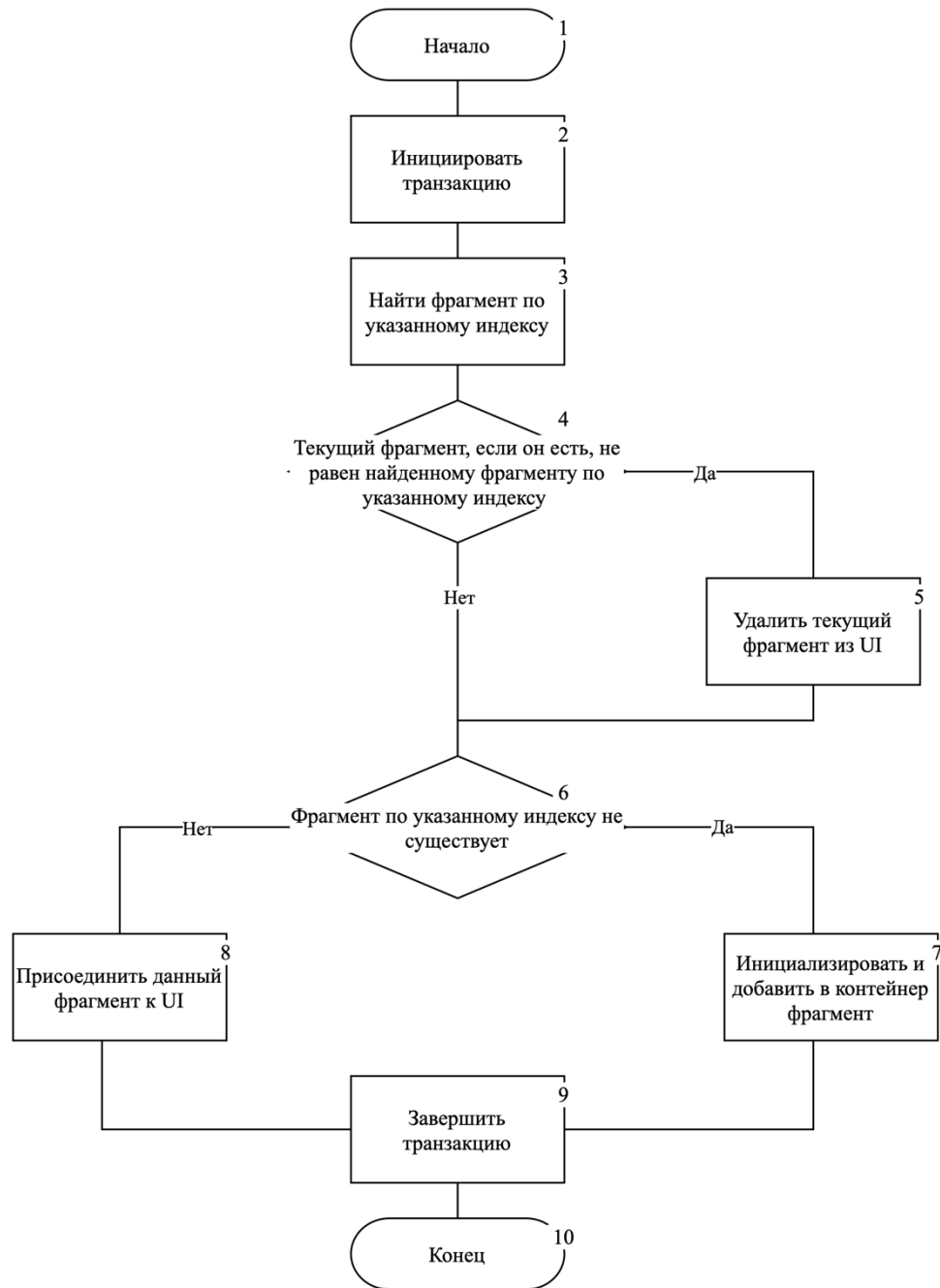


Рисунок 17 – Блок-схема тестируемого алгоритма

Как видно из полученной диаграммы, функция работает с менеджером фрагментов для сохранения состояния при переключении между шагами.

4.2.1 Метод покрытия операторов

Данный критерий подразумевает выполнение каждого оператора программного продукта, по крайней мере, один раз. Для выполнения каждого

оператора было составлено четыре теста, результаты которых приведены в таблице 7.

Таблица 7 – Покрытие операторов

№ оператора	Входные данные	Выходные данные
2, 3, 5, 7, 9	На экране представлен фрагмент, добавляемого фрагмента нет в контейнере	Текущий фрагмент удаляется из UI, затем инициализируется и присоединяется к UI новый фрагмент
2, 3, 7, 9	В контейнере нет фрагментов	Инициализируется и присоединяется к UI новый фрагмент
2, 3, 5, 8, 9	На экране представлен фрагмент, новый фрагмент уже есть в контейнере	Текущий фрагмент удаляется из UI, затем присоединяется новый фрагмент из контейнера
2, 3, 8, 9	Текущий и новый фрагменты имеют одинаковый тег	Ничего не происходит

В ходе тестирования были покрыты все операторы и все маршруты рассматриваемого алгоритма. Полученные результаты показали корректную работу алгоритма.

4.2.2 Метод покрытия условий

Суть метода покрытия условий заключается в том, чтобы составить число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз. Также должно соблюдаться условие выполнения каждого оператора, по крайней мере, один раз. Результаты приведены в таблице 8.

Таблица 8 – Покрытие условий

№ условия	Входные данные	Выходные данные
4 – Да 6 – Да	На экране представлен фрагмент, добавляемого фрагмента нет в контейнере	Текущий фрагмент удаляется из UI, затем инициализируется и присоединяется к UI новый фрагмент
4 – Да 6 – Нет	В контейнере нет фрагментов	Инициализируется и присоединяется к UI новый фрагмент
4 – Нет 6 – Да	На экране представлен фрагмент, новый фрагмент уже есть в контейнере	Текущий фрагмент удаляется из UI, затем присоединяется новый фрагмент из контейнера
4 – Нет 6 – Нет	Текущий и новый фрагменты имеют одинаковый тег	Ничего не происходит

В ходе тестирования были покрыты все условия. Полученные результаты показали корректную работы алгоритма.

4.2.3 Метод покрытия маршрутов

Для выполнения всех маршрутов был составлен граф маршрутов (рисунок 18). Результаты приведены в таблице 9.

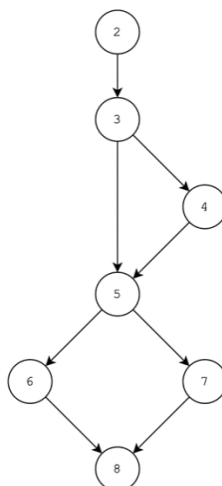


Рисунок 18 – Граф маршрутов

Таблица 9 – Покрытие маршрутов

№ маршрута	Входные данные	Выходные данные
1	На экране представлен фрагмент, добавляемого фрагмента нет в контейнере	Текущий фрагмент удаляется из UI, затем инициализируется и присоединяется к UI новый фрагмент
2	В контейнере нет фрагментов	Инициализируется и присоединяется к UI новый фрагмент
3	На экране представлен фрагмент, новый фрагмент уже есть в контейнере	Текущий фрагмент удаляется из UI, затем присоединяется новый фрагмент из контейнера
4	Текущий и новый фрагменты имеют одинаковый тег	Ничего не происходит

В ходе тестирования были покрыты все маршруты. Полученные результаты показали корректную работу алгоритма.

4.2.4 Классы эквивалентности

Технику разделения на классы эквивалентности в основном используют для сокращения числа тестов, обеспечивая эффективное покрытие тестами всех частей кода. Она подходит для всех типов данных, а потому получила широкое применение в тестировании программных продуктов.

Для всех целочисленных и дробных переменных логическое ограничение будет равно $[0; +\infty]$. Физическим ограничением для целочисленных переменных будет $[-2147483648; 2147483647]$. Для дробных чисел с плавающей запятой физическое ограничение примет вид $[-3.40282347E + 38; 3.40282347E + 38]$.

Для произвольных ограничений необходимо рассмотреть отдельно каждые переменные. Во-первых, возраст используется в расчете вероятности выживания:

$$S_0(\text{возраст}) = \exp(-\exp(\alpha)(\text{возраст} - 20)^p),$$

$$S_0(\text{возраст} + 10) = \exp(-\exp(\alpha)(\text{возраст} - 10)^p),$$

где S_0 – вероятность выживания;

возраст – входной параметр модели;

$$\exp(x) = e^x;$$

коэффициенты α и p берутся из таблицы.

Для возведения в степень необходимо неотрицательное число. Получаем логическое ограничение:

$$\text{возраст} \subseteq [20, +\infty).$$

Данная шкала менее корректно рассчитывает риск смерти о ССЗ для людей старше 90 лет. Поэтому произвольное ограничение получит вид:

$$\text{возраст} \subseteq [0, 90].$$

В итоге получаем класс эквивалентности для значений целого числа *возраст* (рисунок 19).

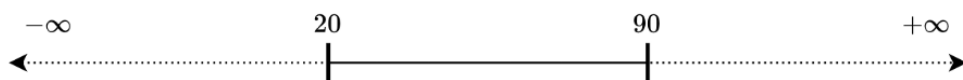


Рисунок 19 – Классы эквивалентности для целого числа *возраст*

Для всех остальных чисел произвольные переменные заданы согласно требованиям техническому заданию (рисунки 20 – 21).

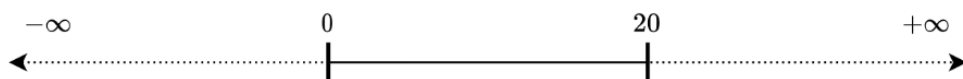


Рисунок 20 – Классы эквивалентности для дробного числа *холестерин*



Рисунок 21 – Классы эквивалентности для целого числа систолическое АД

Обобщим информацию о рассмотренных классах эквивалентности. Результаты представлены в таблице 10.

Таблица 10 – Классы эквивалентности

Входные условия	Правильные классы эквивалентности	Неправильные классы эквивалентности
Значение целого числа возраст	$[20,80]$	$(-\infty, 20) \cup (80, +\infty)$
Значение дробного числа холестерин	$[0,20]$	$(-\infty, 0) \cup (20, +\infty)$
Значение целого числа систолическое АД	$[80,220]$	$(-\infty, 80) \cup (220, +\infty)$
Значение целого числа количество выкуренных сигарет в день	$[0,99]$	$(-\infty, 0) \cup (99, +\infty)$

Таким образом, полученные классы эквивалентности дали понимание, что для тестирования указанной формулы необходимо использовать минимум четыре теста.

4.2.5 Функциональная диаграмма

Метод функциональных диаграмм используется для систематической выборки высоко результативных тестов, а сами функциональные диаграммы – это формальный язык, на который транслируется спецификация, написанная на естественном языке.

Для построения функциональной диаграммы необходимо определить причины и следствия рассматриваемых рабочих участков. Причина – это отдельное входное условие (класс эквивалентности входных условий), а

следствие – выходное условие (преобразование системы). Для определения причин и следствий необходимо последовательно вычитывать спецификации.

На рисунке 22 представлена функциональная диаграмма.

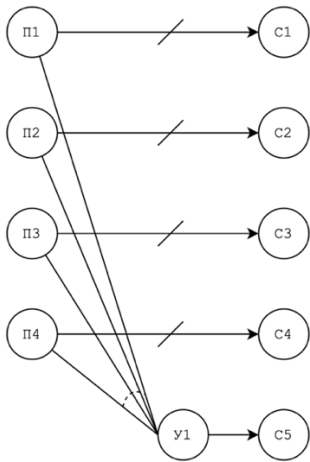


Рисунок 22 – Функциональная диаграмма

Описание причин и следствий представлено в таблицах 11 и 12 соответственно.

Таблица 11 – Описание причин

Причина	Описание
П1	Корректное значение целого числа возраст
П2	Корректное значение дробного числа холестерин
П3	Корректное значение целого числа систолическое АД
П4	Корректное значение целого числа количество выкуренных сигарет в день

Таблица 12 – Описание следствий

Следствие	Описание
С1	Сообщение: «Некорректное значение целого числа возраст»
С2	Сообщение: «Некорректное значение дробного числа холестерин
С3	Сообщение: «Некорректное значение целого числа систолическое АД»
С4	Сообщение: «Некорректное значение целого числа количество выкуренных сигарет в день
С5	Отправка формы на сервер

На основе функциональной диаграммы была составлена таблица решений (таблица 13). Набор тестов, на основе данной таблицы решений представлен в таблице 14.

Таблица 13 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
П1	1	0	1	1	1	0	0	0	1	0	1	1	1	0	0	0
П2	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0
П3	1	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0
П4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
С1	0	1	0	0	0	1	1	1	0	1	0	0	0	1	1	1
С2	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0
С3	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
С4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
С5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 14 – Набор тестов

№ теста	Входные данные	Ожидаемый результат
1	Возраст: 80 Холестерин: 20 Систолическое АД: 220 Кол-во выкуренных сигарет: 0	С5
2	Возраст: 19 Холестерин: 20 Систолическое АД: 220 Кол-во выкуренных сигарет: 0	С1
3	Возраст: 80 Холестерин: -0.0000001 Систолическое АД: 220 Кол-во выкуренных сигарет: 0	С2
4	Возраст: 80 Холестерин: 20 Систолическое АД: 221 Кол-во выкуренных сигарет: 0	С3
5	Возраст: 80 Холестерин: -0.0000001 Систолическое АД: 221 Кол-во выкуренных сигарет: 0	С2

Продолжение таблицы 14

№ теста	Входные данные	Ожидаемый результат
6	Возраст: 19 Холестерин: 20 Систолическое АД: 221 Кол-во выкуренных сигарет: 0	C1
7	Возраст: 19 Холестерин: -0.0000001 Систолическое АД: 220 Кол-во выкуренных сигарет: 0	C1
8	Возраст: 19 Холестерин: -0.0000001 Систолическое АД: 221 Кол-во выкуренных сигарет: 0	C1
9	Возраст: 80 Холестерин: 20 Систолическое АД: 220 Кол-во выкуренных сигарет: 100	C4
10	Возраст: 19 Холестерин: 20 Систолическое АД: 220 Кол-во выкуренных сигарет: 100	C1
11	Возраст: 80 Холестерин: -0.0000001 Систолическое АД: 220 Кол-во выкуренных сигарет: 100	C2
12	Возраст: 80 Холестерин: 20 Систолическое АД: 221 Кол-во выкуренных сигарет: 100	C3
13	Возраст: 80 Холестерин: -0.0000001 Систолическое АД: 221 Кол-во выкуренных сигарет: 100	C2
14	Возраст: 19 Холестерин: 20 Систолическое АД: 221 Кол-во выкуренных сигарет: 100	C1
15	Возраст: 19 Холестерин: -0.0000001 Систолическое АД: 220 Кол-во выкуренных сигарет: 100	C1
16	Возраст: 19 Холестерин: -0.0000001 Систолическое АД: 221 Кол-во выкуренных сигарет: 100	C1

В ходе функционального тестирования были выделены тесты, необходимые для проверки всех возможных ошибок.

4.3 Тестирование серверной части приложения

Для модульного тестирования NodeJs приложений существуют множество готовых инструментов. Самым популярным и простым является библиотека «Mocha». Описание тестов представлено в таблице 15.

Таблица 15 – Описание модульных тестов

№	Модуль	Входные данные	Результат
1	Расчет риска смерти от ССЗ по шкале SCORE	Некурящий мужчина 60 лет со ср. систолическим АД, равным 120 мм.рт.ст., и холестерином, равным 6 ммоль/л	Риск равен 4.34%. Успешно (рисунок Б.1)
2		Курящий мужчина 57 лет со ср. систолическим АД, равным 120 мм.рт.ст., и холестерином, равным 6 ммоль/л	Риск равен 6.65%. Успешно (рисунок Б.2)
3		Курящая женщина 45 лет со ср. систолическим АД, равным 105 мм.рт.ст., и холестерином, равным 4 ммоль/л	Риск равен 0.21%. Успешно (рисунок Б.3)
4		Некурящая женщина 60 лет со ср. систолическим АД, равным 110 мм.рт.ст., и холестерином, равным 5 ммоль/л	Риск равен 1.1%. Успешно (рисунок Б.4)
5		Курящий мужчина 65 лет со ср. систолическим АД, равным 180 мм.рт.ст., и холестерином, равным 8 ммоль/л	Риск равен 49.07%. Успешно (рисунок Б.5)
6	Расчет СКФ по шкале СКD-EPI	Мужчина 55 лет с креатинином, равным 60 мкмоль/л	СКФ равна 108 мл/мин/1,73 м². Успешно (рисунок Б.6)

Продолжение таблицы 15

№	Модуль	Входные данные	Результат
7	Расчет СКФ по шкале СКD-EPI	Мужчина 60 лет с креатинином, равным 90 мкмоль/л	СКФ равна 80 мл/мин/1,73 м². Успешно (рисунок Б.7)
8		Мужчина 50 лет с креатинином, равным 180 мкмоль/л	СКФ равна 37 мл/мин/1,73 м². Успешно (рисунок Б.8)
9		Мужчина 67 лет с креатинином, равным 96 мкмоль/л	СКФ равна 53 мл/мин/1,73 м². Успешно (рисунок Б.9)
10		Мужчина 76 лет с креатинином, равным 108 мкмоль/л	СКФ равна 43 мл/мин/1,73 м². Успешно (рисунок Б.10)
11	Расчет индекса массы тела	Человек с ростом 180 см и весом 93 кг	ИМТ равен 28.7. Успешно (рисунок Б.11)
12		Человек с ростом 170 см и весом 120 кг	ИМТ равен 41.5. Успешно (рисунок Б.12)

Результаты модульного тестирования представлены в приложении Б.

В результате модульного тестирования была проверена работоспособность подсистемы для расчета таких показателей, как: риск смерти от ССЗ по шкале SCORE, скорость клубочковой фильтрации по шкале СКD-EPI и индекс массы тела.

4.4 Метрики программного кода проекта

Для представления более полной информации по программному коду программных средств существует совокупность оценок ПО, или метрик

программного кода [14]. Они служат для определения сложности разрабатываемого программного продукта, оценки объема работы и стилистики написанного кода. Также они предоставляют более полную информацию о программном коде.

Для тестирования возьмем функцию, которая была представлена ранее в листинге 1.

4.4.1 Цикломатическая сложность

Для того чтобы оценить логическую и/или количественную сложность существует цикломатическая сложность, или сложность потока управления программы. Для ее вычисления необходимо построить управляющий граф программы, где узлы графа соответствуют командам программы и соединены ориентированными ребрами. На рисунке 23 представлен управляющий граф.

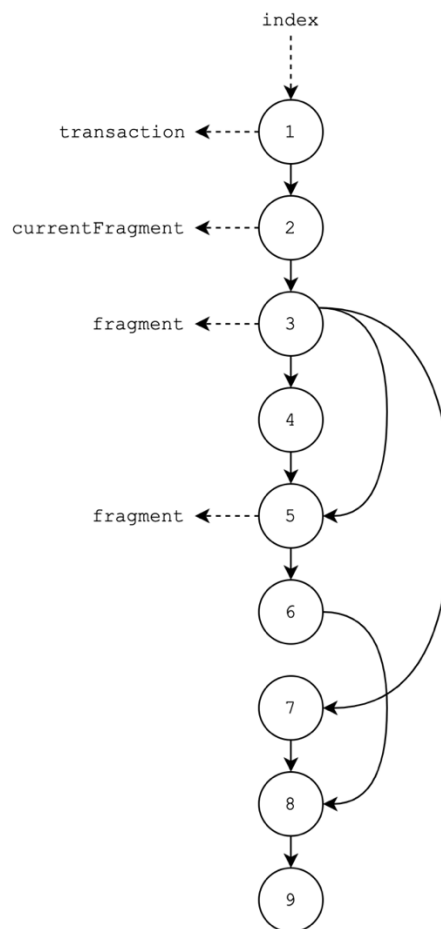


Рисунок 23 – Управляющий граф

Для представленного графа число дуг равно 11, число вершин 9. Для вычисления цикломатической сложности $Z(G)$ применяется формула

$$Z(G) = e - v + 2 \cdot p,$$

где e – число дуг ориентированного графа G ;

v – число вершин;

p – число компонентов связности графа.

Для простой программы или функции число компонентов связности графа всегда равно 1. Значение цикломатической сложности для текущего метода получилась 4, что означает, что метод имеет среднюю структуру сложности.

4.4.2 Метрики сложности потока управления данными

Для данного класса метрик популярным является метрика Чепина, суть которого состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода [15].

Формула для расчета метрики Чепина:

$$Q = a_1 \cdot P + a_2 \cdot M + a_3 \cdot C + a_4 \cdot T,$$

где P – вводимые переменные для расчетов и для обеспечения вывода;

M – модифицируемые или создаваемые внутри программы;

C – управляющие переменные;

T – не используемые в программе переменные;

a_1, a_2, a_3, a_4 – весовые коэффициенты.

С учетом весовых коэффициентов выражение примет вид:

$$Q = P + 2M + 3C + 0.5T = 1 + 2 \cdot 3 + 3 \cdot 0 + 0.5 \cdot 0 = 7$$

На основе полученного значения метрики Чепина уровень сложности данной программы считаем приемлемый.

5 Планирование и оценка бюджета проекта

Управление проектами является неотъемлемой частью проектов, в ходе которой намечаются и достигаются цели проекта. Для достижения положительного результата, учитывая риски и ограничения на трудозатраты, ресурсы и качество, необходимо составить план, рационально применить свои знания и инструментальные средства.

Существует несколько основных групп при планировании проектов:

- планирование проектных задач;
- планирование расписания проекта;
- планирование команды проекта;
- управление рисками.

При планировании разработки было выделено 27 задач. Перечень задач и их виды, предшественники и длительность каждой задачи представлены в таблице 16.

Таблица 16 – Перечень задач проекта

№	Название	Вид задачи	Предшественник	Длительность
1	Начало реализации проекта	Веха	–	–
2	Анализ требований и постановка задачи	Фаза	–	5
3	Постановка задачи	Задача	1	1
4	Анализ функциональных требований	Задача	3	2
5	Выбор средств разработки	Задача	4	2
6	Анализ требований завершен	Веха	5	–
7	Проектирование программных средств	Фаза	–	10
8	Функциональное моделирование	Задача	5	3
9	Проектирование базы данных	Задача	8	2

Продолжение таблицы 16

№	Название	Вид задачи	Предшественник	Длительность
10	Проектирование программных средств	Задача	9	4
11	Проектирование пользовательского интерфейса	Задача	9	5
12	Проектирование завершено	Веха	10;11	–
13	Реализация	Фаза	–	25
14	Реализация базы данных	Задача	12	3
15	Реализация пользовательского интерфейса	Задача	14	7
16	Реализация программных средств	Задача	15	15
17	Реализация завершена	Веха	16	–
18	Отладка и тестирование	Фаза	–	6
19	Отладка серверной части	Задача	17	3
20	Рефакторинг	Задача	19	2
21	Анализ качества кода	Задача	20	1
22	Отладка и тестирование завершена	Веха	21	–
23	Завершение проекта	Фаза	–	3
24	Составление документации	Задача	22	2
25	Сдача в эксплуатацию	Задача	24	1
26	Проект завершен	Веха	25	–
27	Конец проекта	Веха	26	–

Для реализации каждой задачи данного проекта необходимо определить ресурсы, а также их тип и стоимостную оценку. Поэтому при анализе всех необходимых ресурсов и затрат был составлен перечень всех ресурсов проекта, который представлен в таблице 17.

Таблица 17 – Ресурсы и затраты проекта

№	Название ресурса	Тип ресурса	Затраты		Начисление
			Стандартная ставка	Ставка сверхур.	
1	Программист	Трудовой	30 000 р./мес.	150 р./ч.	Пропорциональное
2	Аналитик	Трудовой	20 000 р./мес.	100 р./ч.	Пропорциональное
3	Тестировщик	Трудовой	15 000 р./мес.	150 р./ч.	Пропорциональное
4	Бумага	Материальный	300 р.	–	Пропорциональное
5	USB накопитель	Материальный	500 р.	–	Пропорциональное
6	Интернет	Затраты	–	–	Пропорциональное

Описанные выше ресурсы необходимо распределить по задачам. Распределение ресурсов по задачам приведено в таблице 18.

Таблица 18 – Назначение ресурсов

№	Название	Ресурсы	Единицы (затраты)
1	Начало реализации проекта	–	–
3	Постановка задачи	Программист	50%
4	Анализ функциональных требований	Программист	50%
		Аналитик	100%
		Интернет	350 р.
5	Выбор средств разработки	Программист	100%
		Интернет	350 р.
8	Функциональное моделирование	Программист	100%
9	Проектирование базы данных	Программист	100%
10	Проектирование программных средств	Программист	70%
11	Проектирование пользовательского интерфейса	Программист	30%
14	Реализация базы данных	Программист	100%
15	Реализация пользовательского интерфейса	Программист	100%

Продолжение таблицы 18

№	Название	Ресурсы	Единицы (затраты)
16	Реализация программных средств	Программист	100%
19	Отладка серверной части	Программист Тестировщик	50% 100%
20	Рефакторинг	Программист	100%
21	Анализ качества кода	Программист Тестировщик	50% 100%
24	Составление документации	Аналитик Программист Тестировщик Бумага USB накопитель Интернет	100% 100% 50% 1 уп. 1 шт. 350 р.
25	Сдача в эксплуатацию	Программист Тестировщик Аналитик	100% 100% 100%

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project. Проект был начат 12 ноября 2020 года. Результат внесения ресурсов в проект представлен на рисунке 24.

		Название ресурса	Тип	Стандартная ставка	Ставка сверхурочны	Начисление
1		Программист	Трудовой	30 000,00 ₽/мес	150,00 ₽/ч	Пропорциональное
2		Аналитик	Трудовой	20 000,00 ₽/мес	100,00 ₽/ч	Пропорциональное
3		Тестировщик	Трудовой	15 000,00 ₽/мес	150,00 ₽/ч	Пропорциональное
4		Бумага	Материальный	300,00 ₽		Пропорциональное
5		USB накопитель	Материальный	500,00 ₽		Пропорциональное
6		Интернет	Затраты			Пропорциональное

Рисунок 24 – Ресурсы проекта

Результаты назначений ресурсов в проекте представлены на рисунке 25.

	Название задачи ▾	Длительность ▾	Начало ▾	Окончание ▾	Предш. ▾	Названия ресурсов ▾
	Начало реализации проекта	0 дней	Чт 12.11.20	Чт 12.11.20		
	Анализ требований и постановка задачи	5 дней	Чт 12.11.20	Чт 19.11.20		
	Постановка задачи	1 день	Чт 12.11.20	Чт 12.11.20	1	Программист[50%]
	Анализ функциональных требований	2 дней	Пт 13.11.20	Пн 16.11.20	3	Программист[50%]; Аналитик; Интернет
	Выбор средств разработки	2 дней	Вт 17.11.20	Ср 18.11.20	4	Программист; Интернет
	Анализ требований завершен	0 дней	Чт 19.11.20	Чт 19.11.20	5	
	Проектирование программных средств	10 дней	Чт 19.11.20	Чт 03.12.20		
	Функциональное моделирование	3 дней	Чт 19.11.20	Пн 23.11.20	6	Программист
	Проектирование базы данных	2 дней	Вт 24.11.20	Ср 25.11.20	8	Программист
	Проектирование программных средств	4 дней	Чт 26.11.20	Вт 01.12.20	9	Программист[70%]
	Проектирование пользовательского интерфейса	5 дней	Чт 26.11.20	Ср 02.12.20	9	Программист[30%]
	Проектирование завершено	0 дней	Чт 03.12.20	Чт 03.12.20	10;11	
	Реализация	25 дней	Чт 03.12.20	Чт 07.01.21		
	Реализация базы данных	3 дней	Чт 03.12.20	Пн 07.12.20	12	Программист
	Реализация пользовательского	7 дней	Вт 08.12.20	Ср 16.12.20	14	Программист
	Реализация программ	15 дней	Чт 17.12.20	Ср 06.01.21	15	Программист
	Реализация завершена	0 дней	Чт 07.01.21	Чт 07.01.21	16	
	Отладка и тестирование	6 дней	Чт 07.01.21	Пт 15.01.21		
	Отладка серверной части	3 дней	Чт 07.01.21	Пн 11.01.21	17	Программист[50%]; Т
	Рефакторинг	2 дней	Вт 12.01.21	Ср 13.01.21	19	Программист
	Анализ качества кода	1 день	Чт 14.01.21	Чт 14.01.21	20	Программист[50%]; Тестировщик
	Отладка и тестирование	0 дней	Пт 15.01.21	Пт 15.01.21	21	
	Завершение проекта	3 дней	Пт 15.01.21	Ср 20.01.21		
	Составление документации	2 дней	Пт 15.01.21	Пн 18.01.21	22	Аналитик; Программист; Тестир
	Сдача в эксплуатацию	1 день	Вт 19.01.21	Вт 19.01.21	24	Программист; Тестир
	Проект завершен	0 дней	Ср 20.01.21	Ср 20.01.21	25	
	Конец проекта	0 дней	Ср 20.01.21	Ср 20.01.21	26	

Рисунок 25 – Назначение ресурсов задачам

В соответствии с занесенными задачами проекта, проанализированными ресурсами и соответствующими им затратами Microsoft Office Project автоматически построит диаграмму Ганта, которая иллюстрирует

календарный график выполнения работ по проекту. Диаграмма Ганта представлена на рисунке 26.

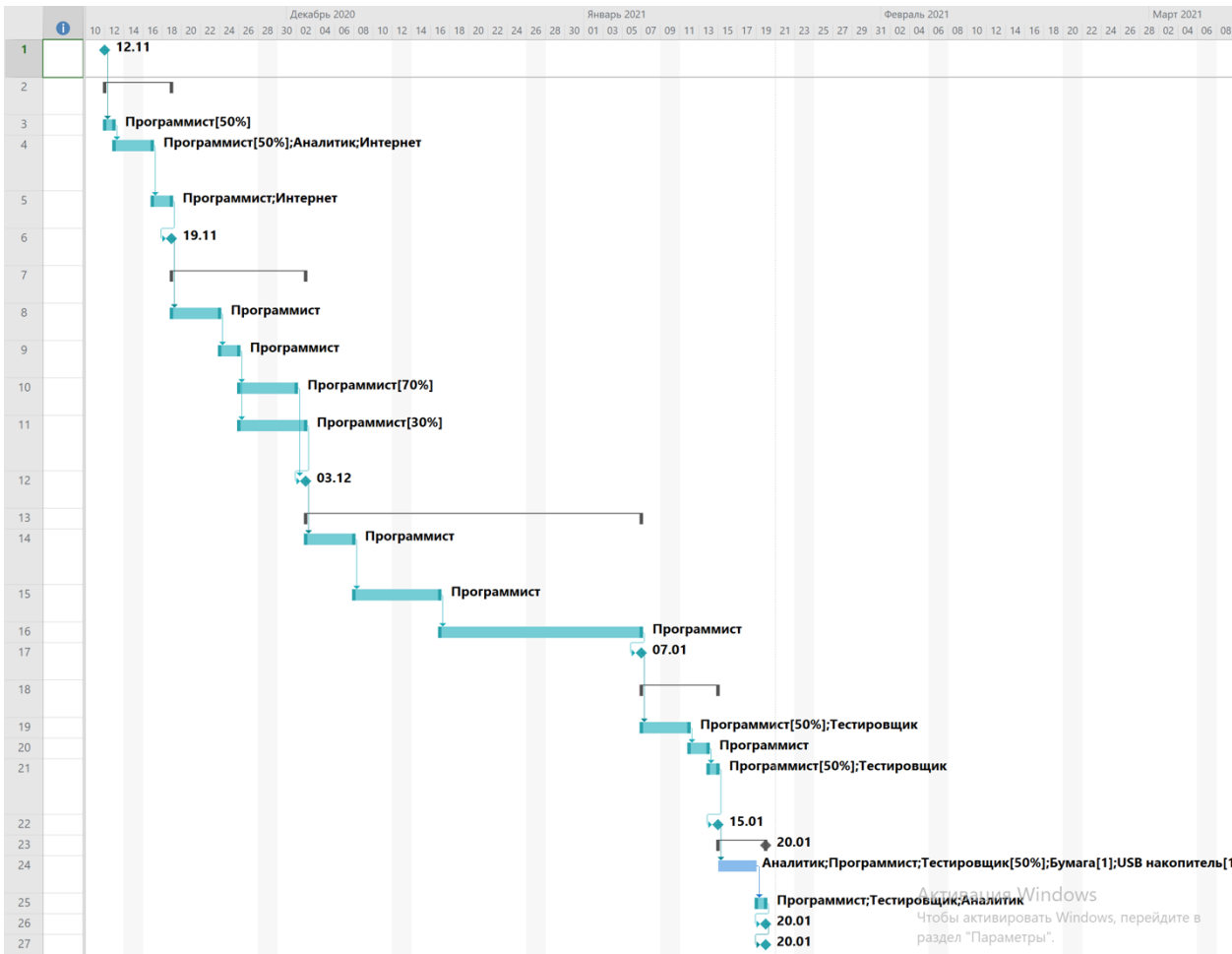


Рисунок 26 – Диаграмма Ганта

Также был сформирован обзорный отчет проекта планирования разработки программных средств сбора и анализа кардиологических данных, в котором содержатся данные о задаче и количестве затраченных средств на ее выполнение. Обзорный отчет проекта представлен на рисунке 27.

Обзорный отчет проекта					
Название	Начало	Окончание	Длительность	Трудозатраты	Затраты
Проект1	Чт 12.11.20	Ср 20.01.21	49 дней	446,4 ч	77 500,00 Р

Рисунок 27 – Обзорный отчет проекта

Как видно из отчета, представленного на рисунке 27, срок реализации проекта составил 49 рабочих дней с учетом выходных дней и праздников, а бюджет проекта был оценен в 77 500 рублей. Каждая задача была успешно выполнена в срок и завершена на 100%.

Помимо этого, программа позволяет построить отчет по движению денежных средств, представленный на рисунке 28, и отчет по исполнителям, представленный на рисунке 29.

Отчет о движении денежных средств	
Название	Затраты
Начало реализации проекта	0,00 Р
Постановка задачи	750,00 Р
Анализ функциональных требований	3 500,00 Р
Выбор средств разработки	3 000,00 Р
Анализ требований завершен	0,00 Р
Функциональное моделирование	4 500,00 Р
Проектирование базы данных	3 000,00 Р
Проектирование программных средств	4 200,00 Р
Проектирование пользовательского интерфейса	2 250,00 Р
Проектирование завершено	0,00 Р
Реализация базы данных	4 500,00 Р
Реализация пользовательского интерфейса	10 500,00 Р
Реализация программных средств	22 500,00 Р
Реализация завершена	0,00 Р
Отладка серверной части	4 500,00 Р
Рефакторинг	3 000,00 Р
Анализ качества кода	1 500,00 Р
Отладка и тестирование завершена	0,00 Р
Составление документации	6 550,00 Р
Сдача в эксплуатацию	3 250,00 Р
Проект завершен	0,00 Р
Конец проекта	0,00 Р

Рисунок 28 – Отчет о движении денежных средств

Программист				
Ид.	Название	Трудозатраты	Начало	Окончание
3	Постановка задачи	4 ч	Чт 12.11.20	Чт 12.11.20
4	Анализ функциональных требований	24 ч	Пт 13.11.20	Пн 16.11.20
5	Выбор средств разработки	16 ч	Вт 17.11.20	Ср 18.11.20
8	Функциональное моделирование	24 ч	Чт 19.11.20	Пн 23.11.20
9	Проектирование базы данных	16 ч	Вт 24.11.20	Ср 25.11.20
10	Проектирование программных средств	22,4 ч	Чт 26.11.20	Вт 01.12.20
11	Проектирование пользовательского интерфейса	12 ч	Чт 26.11.20	Ср 02.12.20
14	Реализация базы данных	24 ч	Чт 03.12.20	Пн 07.12.20
15	Реализация пользовательского интерфейса	56 ч	Вт 08.12.20	Ср 16.12.20
16	Реализация программных средств	120 ч	Чт 17.12.20	Ср 06.01.21
19	Отладка серверной части	36 ч	Чт 07.01.21	Пн 11.01.21
20	Рефакторинг	16 ч	Вт 12.01.21	Ср 13.01.21
21	Анализ качества кода	12 ч	Чт 14.01.21	Чт 14.01.21
24	Составление документации	40 ч	Пт 15.01.21	Пн 18.01.21
25	Сдача в эксплуатацию	24 ч	Вт 19.01.21	Вт 19.01.21

Аналитик				
Ид.	Название	Трудозатраты	Начало	Окончание
4	Анализ функциональных требований	24 ч	Пт 13.11.20	Пн 16.11.20
24	Составление документации	40 ч	Пт 15.01.21	Пн 18.01.21
25	Сдача в эксплуатацию	24 ч	Вт 19.01.21	Вт 19.01.21

Тестировщик				
Ид.	Название	Трудозатраты	Начало	Окончание
19	Отладка серверной части	36 ч	Чт 07.01.21	Пн 11.01.21
21	Анализ качества кода	12 ч	Чт 14.01.21	Чт 14.01.21
24	Составление документации	40 ч	Пт 15.01.21	Пн 18.01.21
25	Сдача в эксплуатацию	24 ч	Вт 19.01.21	Вт 19.01.21

Рисунок 29 – Отчет по делам исполнителей

По отчету, представленному на рисунке 29, можно выявить наиболее важных исполнителей для грамотного управленческого решения. Исходя из полученных отчетов полученная оценка бюджета и сроки являются экономически выгодными для разработки программных средств сбора и анализа кардиологических данных.

Заключение

В ходе выполнения выпускной квалификационной работы было разработаны программные средства сбора и анализа кардиологических данных.

На этапе анализа предметной области и требований были описаны методы реализации проекта, на основе которых были выработаны функциональные и нефункциональные требования к разрабатываемому продукту.

На этапе проектирования было осуществлено планирование разработки программных средств и оценка бюджета. По составленному отчету длительность проекта составила 49 дней, а затраты на реализацию – 77 500 рублей.

Также была спроектировано и реализовано мобильное приложение вместе с сервером и базой данных. После было проведено тестирование серверной части, на основе которого был сделан вывод о корректности работы программных средств.

Разработанные программные средства сбора и анализа кардиологических данных в результате выполнения выпускной квалификационной работы отвечают всем требованиям технического задания.

Список использованных источников

1. Митрохин, А.С. Информационные технологии в современной медицине / А.С. Митрохин // Актуальные проблемы гуманитарных и естественных наук. – 2010. – №9. – С. 355 – 359.
2. Наливаева, А.В. Информационные технологии в медицине: доказанные факты и нерешенные проблемы / А.В. Наливаева // Бюллетень медицинских интернет-конференций. – 2012. Том 2. – №11. – С. 894 – 897.
3. Инглис С., Кларк Р., Макалистер Ф. Структурированная поддержка по телефону или телемониторинга программ для пациентов с хронической сердечной недостаточностью // Кокрановская База данных Систематических Обзоров/ 2010. Вып. 8.
4. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание / Д. Арлоу, И. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.
5. Куперштейн, В. И. Microsoft Project в управлении проектами. / В. И. Куперштейн, А.В. Цветкова. – СПб.: БХВ-Петербург, 2011. – 415 с.
6. Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API // tproger.ru [Электронный ресурс]. – URL: <https://tproger.ru/articles/web-api/> (Дата обращения: 03.06.2021).
7. Слободан Стоянович, Александар Симович Бессерверные приложения на JavaScript / пер. с англ. А. Н. Киселева. – М.: ДМК Пресс, 2020. – 394 с.: ил.
8. Firebase: платформа разработки приложений.URL:// Сайт URL:разработчиков Firebase [Электронный <https://firebase.apptractor.ru> (дата обращения 18.05.2021).
9. Backdrop – Material Design // material.io [Электронный ресурс]. – URL: <https://material.io/components/backdrop/> (Дата обращения: 02.06.2021).
10. Дворянин, Д. М. К вопросу о понятиях UX UI дизайна / Д. М. Дворянин, К. С. Скубников // Аллея науки. – 2017. – С. 781 – 785.

11. Киммел, П. UML. Основы визуального анализа и проектирования / П. Киммел. – М.: НТ Пресс, 2008. – 272 с.
12. Майерс, Г. Искусство тестирования программ / Пер. с англ. под ред. Б.А. Позина. – Москва: Финансы и статистика, 1982. – 176 с.
13. Михалицын, М. М. Технология программирования. Проектирование и тестирование программ: Учеб. пособие / М. М. Михалицын – Новосибирск.: СГГА, 2003. – 73 с.
14. Ряжков Е. Программный код и его метрики // Habr [Электронный ресурс]. – URL: <https://habr.com/company/intel/blog/106082/> (дата обращения: 08.06.2021).
15. Средство оценки качества кода // habr.com [электронный ресурс]. – URL: <https://habr.com/ru/post/111524/> (09.06.2021).
16. Методология. Стандарт IDEF1 // Iteranet IT-журнал [Электронный ресурс]. – URL: <http://iteranet.ru/it-novosti/2013/10/16/metodologiya-standart-idef1/> (дата обращения: 22.05.2021).

Приложение А
ГЛОССАРИЙ
(справочное)

Программное обеспечение (ПО) – это комплекс программных средств для ЭВМ, которые обеспечивают деятельность, диагностику и тестирование их аппаратных средств.

Фельдшерско-акушерских пунктов (ФАП) – структурное подразделение лечебно-профилактического учреждения, осуществляющее доврачебную первичную медико-санитарную помощь в сельской местности.

Сердечно-сосудистые заболевания (ССЗ) – общее название заболеваний сердца и кровеносных сосудов.

UML (Unified Modeling Language) – унифицированный язык моделирования графического описания для создания абстрактной модели системы.

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

MVVM (Model-View-ViewModel) – шаблон проектирования архитектуры приложения, подразумевающий разделение модели на три части: модель (логика работы с данными), представление (графическая составляющая) и модель представления (обертка данных из модели, подлежащих связыванию).

Приложение Б
РЕЗУЛЬТАТЫ МОДУЛЬНОГО ТЕСТИРОВАНИЯ
(рекомендуемое)

✓ Риск равен 4.34%, если некурящий мужчина 60 лет, со ср. сист. АД 120 мм.рт.ст и холестерином 6 ммоль/л

Рисунок Б.1 – Результат теста №1

✓ Риск равен 6.65%, если курящий мужчина 57 лет, со ср. сист. АД 120 мм.рт.ст и холестерином 6 ммоль/л

Рисунок Б.2 – Результат теста №2

✓ Риск равен 0.21%, если курящая женщина 45 лет, со ср. сист. АД 105 мм.рт.ст и холестерином 4 ммоль/л

Рисунок Б.3 – Результат теста №3

✓ Риск равен 1.1%, если некурящая женщина 60 лет, со ср. сист. АД 110 мм.рт.ст и холестерином 5 ммоль/л

Рисунок Б.4 – Результат теста №4

✓ Риск равен 49.07%, если курящий мужчина 65 лет, со ср. сист. АД 180 мм.рт.ст и холестерином 8 ммоль/л

Рисунок Б.5 – Результат теста №5

✓ СКФ равна 108 мл/мин/1,73м², если мужчина 55 лет, с креатинином 60 мкмоль/л

Рисунок Б.6 – Результат теста №6

✓ СКФ равна 80 мл/мин/1,73м², если мужчина 60 лет, с креатинином 90 мкмоль/л

Рисунок Б.7 – Результат теста №7

✓ СКФ равна 37 мл/мин/1,73м², если мужчина 50 лет, с креатинином 180 мкмоль/л

Рисунок Б.8 – Результат теста №8

✓ СКФ равна 53 мл/мин/1,73м², если женщина 67 лет, с креатинином 96 мкмоль/л

Рисунок Б.9 – Результат теста №9

✓ СКФ равна 43 мл/мин/1,73м², если женщина 76 лет, с креатинином 108 мкмоль/л

Рисунок Б.10 – Результат теста №10

✓ Индекс массы тела равен 28.7, если человек с ростом 180 см и весом 93 кг

Рисунок Б.11 – Результат теста №11

✓ Индекс массы тела равен 41.5, если человек с ростом 170 см и весом 120 кг

Рисунок Б.12 – Результат теста №12

Приложение В
ЛИСТИНГ СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг В.1 – mainController.js

```

const admin = require('firebase-admin');
const db = admin.firestore();

const { SCORE, CKD_EPI, BMI } =
require('../services/evaluationService');

const dateFormat = require('dateformat');

const Attribute = require('../models/data');
const attrs = Attribute.attrs();
const Type = require('../models/type');
const httpError = require('http-errors');

exports.post = async (req) => {

  const { data, user } = req;

  const evaluations = {
    SCORE: SCORE({
      age: data[Attribute.Keys.AGE],
      sex: data[Attribute.Keys.SEX],
      systolic: data[Attribute.Keys.SYSTOLIC_BLOOD_PRESSURE],
      smoker: data[Attribute.Keys.SIGARETTES] > 0,
      cholesterol: data[Attribute.Keys.CHOLESTEROL],
    }),
    CKD_EPI: CKD_EPI({
      age: data[Attribute.Keys.AGE],
      sex: data[Attribute.Keys.SEX],
      creatinine: data[Attribute.Keys.CREATININE],
    }),
    BMI: BMI({
      weight: data[Attribute.Keys.WEIGHT],
      height: data[Attribute.Keys.HEIGHT],
    }),
  };
  evaluations.createdAt = new Date();
  evaluations.createdBy = {
    uid: user.uid,
    email: user.email,
  }
  evaluations.id = await save({ data, evaluations });

  return evaluations;
};

const save = async ({ data, evaluations }) => {
  var dataToAdd = {};
  Object.keys(data).forEach((id) => {
    let v;
    if (attrs[id].type === Type.SINGLE_CHOICE) {
      v = attrs[id].options[data[id]];

    } else if (attrs[id].type === Type.MULTIPLE_CHOICE) {
      v = []
      data[id].forEach((i) => {
        v.push(attrs[id].options[i]);
      });
    }
  });

```

```

    } else {
      v = data[id];
    }
    dataToAdd[id] = v;
  });

  const docRef = await db.collection('records').add({
    createdAt: evaluations.createdAt,
    createdBy: evaluations.createdBy,
    data: dataToAdd,
    evaluations: evaluations,
  });

  return docRef.id;
};

exports.rate = async (req, res) => {
  const id = req.query.id;
  const review = req.review;

  const recordRef = db.collection('records').doc(id);
  const record = await recordRef.get();
  if (!record.exists) {
    throw httpError(400, `Record with id ${id} does not exist.`);
  }
  if (record.data().createdBy.uid !== req.user.uid) {
    throw httpError(401, 'You cannot rate this evaluation.');
```

}

```

  await recordRef.update({ review });

};

const NOT_FILLED = 'Не заполнено';
const BASE_URL = 'https://us-central1-score-school.cloudfunctions.net/api/photo/';

exports.statement = async (req, res) => {

  headers = [
    'Пользователь',
    'Составлен',
    'SCORE',
    'СКФ по шкале CKD-EPI',
    'Индекс массы тела',
    'Знали о рекомендациях',
    'Рекомендации были полезны',
    'Оценка'
  ];

  Object.keys(attrs).forEach((id) => {
    e = attrs[id]
    headers.push(e.name);
  });

  const recordsRef = db.collection('records');
  const snapshot = await recordsRef.orderBy('createdAt',
'desc').get();
  if (snapshot.empty) {
```

```

    throw httpError(404, 'No matching documents');
  }

  rows = [];
  snapshot.forEach((doc) => {
    const data = doc.data();
    row = [];

    row.push(data.createdBy.email);

    const localDate = data.createdAt
      .toDate()
      .toLocaleString('en-US', { timeZone: 'Europe/Moscow' });
    row.push(dateFormat(new Date(localDate), 'dd.MM.yyyy HH:mm:ss'))

    row.push((data.evaluations.SCORE * 100).toFixed(2).replace('.',
    ', '));
    row.push(Math.round(data.evaluations.CKD_EPI));
    row.push((data.evaluations.BMI).toFixed(1).replace('.', ', '));
    if (data.review === undefined) {
      for (let i = 0; i < 3; i++) {
        row.push(NOT_FILLED);
      }
    } else {
      const review = data.review;
      row.push(review.aware ? 'Да' : 'Нет');
      row.push(review.helpful ? 'Да' : 'Нет');
      row.push(review.rate);
    }
  }

  Object.keys(attrs).forEach((id) => {
    const attr = attrs[id];
    let item = data.data[id];

    let cell;
    if (item === undefined) {
      cell = NOT_FILLED
    } else if (attr.type === Type.FLOAT) {
      cell = item.toFixed(3).replace('.', ', ');
    } else if (attr.type === Type.BOOLEAN) {
      cell = item ? 'Да' : 'Нет'
    } else if (attr.type === Type.MULTIPLE_CHOICE) {
      cell = item.join(', ');
    } else if (attr.type === Type.DATE) {
      cell = dateFormat(item.toDate(), 'dd.mm.yyyy');
    } else if (attr.type === Type.PHOTO) {
      newCell = [];
      item.forEach((photoId) => {
        newCell.push(BASE_URL + photoId)
      });
      cell = newCell.join(', ');
    } else {
      cell = item;
    }
  }

```

```

    }
    row.push(cell);
  });
  rows.push(row);
});

let str = headers.join(';') + '\n';
rows.forEach((row) => {
  str = str + row.join(';') + '\n';
});

return str;
};

```

Листинг В.2 – photoController.js

```

const admin = require('firebase-admin');
const db = admin.firestore();
const bucket = admin.storage().bucket();
const httpError = require('http-errors');

exports.uploadPhoto = async (req, res) => {
  if (!req.files) {
    throw httpError(400, 'You must send photo');
  }

  const photo = req.files.find(file => file.fieldname === 'photo');
  if (!photo) {
    throw httpError(400, 'You must send photo');
  }

  const path = `photos/${Date.now()}.jpg`;
  const file = bucket.file(path);

  return new Promise(async (resolve, reject) => {
    file.createWriteStream({
      metadata: {
        contentType: photo.mimetype
      },
      resumable: false
    })
    .on('error', (err) => reject(err))
    .on('finish', async () => {
      const photo = {
        createdAt: new Date(),
        createdBy: {
          uid: req.user.uid,
          email: req.user.email,
        },
        path: path,
      }
      const { id } = await db.collection('photos').add(photo);
      resolve({ id });
    })
    .end(photo.buffer);
  }).catch((err) => { throw err; });
}

```

```

exports.getPhoto = async (req, res) => {
  const id = req.params.photoId;
  const photoInfo = (await db.collection('photos').doc(id).get());
  if (!photoInfo.exists) {
    throw httpError(404, 'No matching photo found');
  }
  const file = bucket.file(photoInfo.data().path);
  const signedUrls = await file.getSignedUrl({
    action: 'read',
    expires: new Date(Date.now() + 1000 * 60 * 60),
  });

  return signedUrls[0];
};

```

Листинг В.3 – helpers/index.js

```

exports.awareOfAsyncErrors = (handler) => {
  return async (req, res, next) => {
    try {
      return await handler(req, res, next);
    } catch (e) {
      return next(e);
    }
  };
};

```

Листинг В.4 – auth.js

```

const admin = require('firebase-admin');
const httpError = require('http-errors');
const { awareOfAsyncErrors } = require('../helpers');

exports.authenticate = awareOfAsyncErrors(async (req, res, next) => {
  if (!req.headers.authorization ||
    !req.headers.authorization.startsWith('Bearer ')) {
    throw httpError.Unauthorized();
  }

  const token = req.headers.authorization.split('Bearer ')[1];
  try {
    const decodedToken = await admin.auth().verifyIdToken(token);
    req.user = decodedToken;

    return next();
  } catch (e) {
    throw httpError.Unauthorized();
  }
});

exports.isAdmin = awareOfAsyncErrors(async (req, res, next) => {
  const user = await admin.auth().getUser(req.user.user_id);
  if (!user.customClaims.admin) {
    throw httpError(403, 'You do not have privileges');
  }

  return next();
});

```

Листинг В.5 – errorHandler.js

```
const HttpsErrors = require('http-errors');

module.exports = (err, req, res, next) => {
  console.log(err);

  if (err instanceof HttpsErrors.HttpError) {
    res.status(err.status).json({ msg: err.message });
  } else {
    res.status(500);
    res.json({
      msg: err.message,
      stacktrace: err.stacktrace,
    });
    // if (process.env.NODE_ENV === 'development') {
    //   res.json({
    //     msg: err.message,
    //     stacktrace: err.stacktrace,
    //   });
    // } else {
    //   res.json({
    //     msg: 'INTERNAL',
    //   });
    // }
  }
};
```

Листинг В.6 – validator.js

```
const Type = require('../models/type');
const dataModel = require('../models/data');
const attrs = dataModel.attrs();
const httpError = require('http-errors');
const { awareOfAsyncErrors } = require('../helpers');

exports.validateDataForRecord = awareOfAsyncErrors((req, res, next) => {
  {
    const { data } = req.body;
    if (data === undefined || !Array.isArray(data)) {
      throw httpError.BadRequest();
    }

    const ids = Object.keys(attrs);

    const result = {};
    data.forEach((e) => {

      const attr = attrs[e.id];
      if (attr === undefined) {
        throw httpError.BadRequest();
      }
      const value = e.value;

      var v;
      if (attr.type === Type.PHOTO) {
```

```

    v = value;

    } else if (attr.type === Type.TEXT) {
        v = value;

    } else if (attr.type === Type.INTEGER) {
        v = parseInt(value);
        if (isNaN(v)) {
            throw httpError(400, `Attribute with id ${e.id} must be of
type integer.`);
        }

    } else if (attr.type === Type.FLOAT) {
        v = parseFloat(value);
        if (isNaN(v)) {
            throw httpError(400, `Attribute with id ${e.id} must be of
type float.`);
        }

    } else if (attr.type === Type.DATE) {
        let timestamp = parseInt(value);
        if (!timestamp) {
            throw httpError(400, `Attribute with id ${e.id} must be of
type timestamp.`);
        }
        v = new Date(timestamp);

    } else if (attr.type === Type.BOOLEAN) {
        v = Boolean(value);

    } else if (attr.type === Type.SINGLE_CHOICE) {
        v = parseInt(value);
        if (isNaN(v)) {
            throw httpError(400, `Attribute with id ${e.id} must be an
index.`);
        }
        if (v >= attr.options.length) {
            throw httpError(400, `Index of attribute with id ${e.id} must
be within the range [0..${attr.options.length - 1}]`);
        }

    } else if (attr.type === Type.MULTIPLE_CHOICE) {
        if (!(value instanceof Array)) {
            throw httpError(400, `Attribute with id ${e.id} must be of
type array.`);
        }
        value.sort()
        v = []
        value.forEach((i) => {
            const n = parseInt(i);
            if (isNaN(n)) {
                throw httpError(400, `Attribute with id ${e.id} must contain
array if indexes.`);
            }
            if (v >= attr.options.length) {
                throw httpError(400, `All indexes of attribute with id
${e.id} must be within the range [0..${attr.options.length - 1}]`);
            }
            v.push(n);

```

```

    });

    } else {
        throw httpError(500, `Attribute with id ${e.id} has undefined
type.`);
    }

    result[e.id] = v;
});
ids.forEach((id) => {
    let attr = attrs[id]
    if (attr.required) {
        if (!data.find(e => e.id === id)) {
            throw httpError(400, `Attribute with id ${e.id} does not seem
to be filled.`);
        }
    }
});

req.data = result;

return next();
});

exports.validateReview = awareOfAsyncErrors(async (req, res, next) =>
{
    const id = req.query.id;
    let { aware, helpful, rate } = req.body;
    if (id === undefined || aware === undefined || helpful === undefined
|| rate === undefined) {
        throw httpError.BadRequest();
    }
    rate = parseInt(rate);
    if (isNaN(rate) || rate < 1 || rate > 5) {
        throw httpError.BadRequest();
    }

    req.review = {
        aware: Boolean(aware),
        helpful: Boolean(helpful),
        rate: rate,
    };

    return next();
});

```

Листинг В.7 – data.js

```

const Type = require('./type');

const steps = [
    {
        id: "VVKN6k4LsWKOr9s6aTIZ",
        name: "Общие данные",
    },
    {
        id: "JCHiKzc4TSuZRJ8v62HC",
        name: "Жалобы",
    },

```

```

    {
      id: "RCD7VTgc04QQAMv2Pwqf",
      name: "EQ-5D-3L",
    },
    {
      id: "mR0Knakta8fsTCShtJom",
      name: "Анамнез",
    },
    {
      id: "zeEzYWza8CRcOD6EcLCy",
      name: "Наличие болезней",
    },
    {
      id: "H7AEj92vFmaCoJmdr167",
      name: "Принимаемые препараты",
    },
    {
      id: "3UDmXl59PXgjmb1T8jA5",
      name: "ОАК",
    },
    {
      id: "KbEQGPMcOl7WJJTVJNpd",
      name: "БАК",
    },
    {
      id: "WA12G0V7GgzoOdwZ73dS",
      name: "ЭКГ",
    },
    {
      id: "9oqrRtcFYqxWpYYfffcKy",
      name: "ЭХОЭКГ",
    },
    {
      id: "oQsZxAGWhkcXFSqtieft",
      name: "ХМЭКГ",
    }
  ];

const attrs = [
  {
    name: "Фамилия волонтера",
    type: Type.TEXT,
    required: true,
    id: "KzTi18pqbvTaBmUTX1TT",
    stepId: "VVKN6k4LsWKO9s6aTIZ"
  },
  {
    name: "Имя волонтера",
    type: Type.TEXT,
    required: true,
    id: "52tOTfbNw4neUw99BNah",
    stepId: "VVKN6k4LsWKO9s6aTIZ"
  },
  {
    name: "Отчество волонтера",
    type: Type.TEXT,
    required: true,
    id: "GHBwfYeLlPAHlHkCF3x3",
    stepId: "VVKN6k4LsWKO9s6aTIZ"
  }
];

```

```

},
{
  name: "Возраст",
  type: Type.INTEGER,
  required: true,
  id: "7VWqa7NcyKcwos4jTZkz",
  stepId: "VVKN6k4LsWKOOr9s6aTIZ"
},
{
  name: "Пол",
  type: Type.SINGLE_CHOICE,
  options: [
    "Мужской",
    "Женский"
  ],
  required: true,
  id: "zbm9VRuOkCdX5BkwmyyD",
  stepId: "VVKN6k4LsWKOOr9s6aTIZ"
},
{
  name: "Рост",
  type: Type.INTEGER,
  required: true,
  id: "8xVn3IP8LUN7TTGr1fn8",
  stepId: "VVKN6k4LsWKOOr9s6aTIZ"
},
{
  name: "Вес",
  type: Type.INTEGER,
  required: true,
  id: "kZqXoXv9az0n03fCfmgR",
  stepId: "VVKN6k4LsWKOOr9s6aTIZ"
},
{
  name: "Среднее значение систолического АД за последние две недели",
  type: Type.INTEGER,
  required: true,
  id: "Rtpr6Q1ZgQxj8lgfllip",
  stepId: "VVKN6k4LsWKOOr9s6aTIZ"
},
{
  name: "Среднее значение диастолического АД за последние две недели",
  type: Type.INTEGER,
  required: true,
  id: "V2FjQDvZxUPJgldMefBV",
  stepId: "VVKN6k4LsWKOOr9s6aTIZ"
},
{
  name: "Жалобы за последний месяц",
  type: Type.MULTIPLE_CHOICE,
  options: [
    "Нет жалоб",
    "Боль в сердце",
    "Перебои в работе сердца",
    "Одышка",
    "Кашель",
    "Отделение мокроты",

```

```

        "Головокружение",
        "Головная боль",
        "Шум в ушах",
        "Онемение ног, рук",
        "Бессонница"
    ],
    id: "U7TBhMGoINXbgNoRzMxZ",
    stepId: "JCHiKzc4TSuZRJ8v62HC"
},
{
    name: "Подвижность",
    type: Type.SINGLE_CHOICE,
    options: [
        "Я не испытываю трудностей при ходьбе",
        "Я испытываю некоторые трудности при ходьбе",
        "Я прикован к постели"
    ],
    required: true,
    id: "dBxm0h1lexSddGJTULRX",
    stepId: "RCD7VTgc04QQAMv2Pwqf"
},
{
    name: "Уход за собой",
    type: Type.SINGLE_CHOICE,
    options: [
        "Я не испытываю трудностей при уходе за собой",
        "Я испытываю некоторые трудности с мытьем или одеванием",
        "Я не в состоянии самостоятельно мыться или одеваться"
    ],
    required: true,
    id: "wJWxmd7U78C1cnbwSdzS",
    stepId: "RCD7VTgc04QQAMv2Pwqf"
},
{
    name: "Повседневная деятельность",
    type: Type.SINGLE_CHOICE,
    options: [
        "Я не испытываю трудностей в моей привычной повседневной деятельности",
        "Я испытываю некоторые трудности в моей привычной повседневной деятельности",
        "Я не в состоянии заниматься своей привычной повседневной деятельностью"
    ],
    required: true,
    id: "oSm4ZEZDQkSpMe5WVEPw",
    stepId: "RCD7VTgc04QQAMv2Pwqf"
},
{
    name: "Боль/Дискомфорт",
    type: Type.SINGLE_CHOICE,
    options: [
        "Я не испытываю боли или дискомфорта",
        "Я испытываю умеренную боль или дискомфорт",
        "Я испытываю крайне сильную боль или дискомфорт"
    ],
    required: true,
    id: "R4wnG90AJnJ7c6GIHO8w",
    stepId: "RCD7VTgc04QQAMv2Pwqf"
}

```

```

},
{
  name: "Древога/Депрессия",
  type: Type.SINGLE_CHOICE,
  options: [
    "Я не испытываю тревоги или депрессии",
    "Я испытываю умеренную тревогу или депрессию",
    "Я испытываю крайне сильную тревогу или депрессию"
  ],
  required: true,
  id: "IQRIndqYqVDy7OcFMH2Y",
  stepId: "RCD7VTgc04QQAMv2Pwqf"
},
{
  name: "Семейное положение",
  type: Type.SINGLE_CHOICE,
  options: [
    "В браке",
    "Не в браке"
  ],
  id: "FfhX2bkktiVGvcSeNJxi",
  stepId: "mR0Knakta8fsTCShTjom"
},
{
  name: "Наличие детей",
  type: Type.INTEGER,
  id: "n8Jp7wEU6gJjS2pmwG8f",
  stepId: "mR0Knakta8fsTCShTjom"
},
{
  name: "Просмотр телевизора или компьютера",
  type: Type.INTEGER,
  id: "THIgtPBWg9luk6xI2QaI",
  stepId: "mR0Knakta8fsTCShTjom"
},
{
  name: "Ходьба на улице",
  type: Type.INTEGER,
  id: "JTvGlb40EMbcy0SK2Dgb",
  stepId: "mR0Knakta8fsTCShTjom"
},
{
  name: "Черта личности",
  type: Type.SINGLE_CHOICE,
  options: [
    "Интроверт",
    "Экстраверт"
  ],
  id: "85jDi2J3ZZXUELdw3cqV",
  stepId: "mR0Knakta8fsTCShTjom"
},
{
  name: "Количество выкуранных сигарет в день",
  type: Type.INTEGER,
  id: "smvS1V2rNa4tcOXR23To",
  stepId: "mR0Knakta8fsTCShTjom"
},
{
  name: "Работающий",

```

```

    type: Type.BOOLEAN,
    id: "vA9xM8tEIKf0XjtBW8ka",
    stepId: "mR0Knakta8fsTCShtJom"
  },
  {
    name: "Наличие домашних животных",
    type: Type.BOOLEAN,
    id: "O2uoJcKpmKmpFJNz8JEW",
    stepId: "mR0Knakta8fsTCShtJom"
  },
  {
    name: "Стрессоустойчивость",
    type: Type.BOOLEAN,
    id: "2EYq4TXloQw2zZZSgcSd",
    stepId: "mR0Knakta8fsTCShtJom"
  },
  {
    name: "Метеозависимость",
    type: Type.BOOLEAN,
    id: "xWGebDtFmIQ08RMelVx6",
    stepId: "mR0Knakta8fsTCShtJom"
  },
  {
    name: "Артериальная гипертензия",
    type: Type.BOOLEAN,
    id: "51HmPz5OXa8W5X3XiJa0",
    stepId: "zeEzYWza8CRcOD6EcLCy"
  },
  {
    name: "Стаж заболевания",
    type: Type.INTEGER,
    id: "T5WPQRmCLYq3DdnPKsq6",
    stepId: "zeEzYWza8CRcOD6EcLCy"
  },
  {
    name: "Фибрилляция предсердий (мерцательная аритмия)",
    type: Type.BOOLEAN,
    id: "ZO0h0Yt1awKxrMWIMpuQ1",
    stepId: "zeEzYWza8CRcOD6EcLCy"
  },
  {
    name: "Другие нарушения ритма сердца (кроме фибрилляции предсердий)",
    type: Type.BOOLEAN,
    id: "aM3u4LWNASxW7deMweoy",
    stepId: "zeEzYWza8CRcOD6EcLCy"
  },
  {
    name: "Стаж заболевания",
    type: Type.INTEGER,
    id: "9azwY983Fv93Zic2e0MH",
    stepId: "zeEzYWza8CRcOD6EcLCy"
  },
  {
    name: "ИБС. Стенокардия",
    type: Type.BOOLEAN,
    id: "u86OKigx8etodvcwf05I",
    stepId: "zeEzYWza8CRcOD6EcLCy"
  },

```

```

{
  name: "Стаж заболевания",
  type: Type.INTEGER,
  id: "rCovcsb9wjyP2UTwiVit",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "ПИКС (перенесенный инфаркт миокарда)",
  type: Type.BOOLEAN,
  id: "6N2YSIw9LQkMcS3VhOGt",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Стаж заболевания",
  type: Type.INTEGER,
  id: "Qp5eMhC11Hu5BRTFWw7D",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Сахарный диабет",
  type: Type.BOOLEAN,
  id: "YfUCA3jP5K3mCF3jKo9n",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Стаж заболевания",
  type: Type.INTEGER,
  id: "74Mz7aVuTX0zWdSuUQKu",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "ОНМК (мозговой инсульт)",
  type: Type.BOOLEAN,
  id: "5nQMrASGxTv5LZTq6oJw",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Болезни щитовидной железы",
  type: Type.BOOLEAN,
  id: "A9bxqnW6RJQqoabhQByg",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Болезни почек",
  type: Type.BOOLEAN,
  id: "jeI0FCsJIjGPqtI9AEN7",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Болезни легких (бронхиальная астма, ХОБЛ)",
  type: Type.BOOLEAN,
  id: "ch1YMGrfxP21eaQCsAEw",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Болезни печени (гепатит, цирроз)",
  type: Type.BOOLEAN,
  id: "OOXQx22KUsAAp9PBnPcX",
  stepId: "zeEzYWza8CRcOD6EcLCy"
}

```

```

},
{
  name: "COVID-19",
  type: Type.BOOLEAN,
  id: "XBV4kYvWDHjnwDeAkzcv",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Дата заболевания",
  type: Type.DATE,
  id: "sWhcBl9OfPwE81LRxzkB",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Процент поражения легочной ткани ",
  type: Type.INTEGER,
  id: "Np3dP0tMawSzvIpxAoKW",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Проводилось ли лечение в стационаре",
  type: Type.BOOLEAN,
  id: "GDUJ5vqGnj6B6WrO7Jrs",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Жалобы на момент заболевания",
  type: Type.MULTIPLE_CHOICE,
  options: [
    "кашель",
    "температура выше 37",
    "температура выше 38",
    "одышка",
    "потеря обоняния",
    "потеря вкуса"
  ],
  id: "RvezyFB9BE1lpsc2Tkc5",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Онкологические заболевания",
  type: Type.BOOLEAN,
  id: "fsJlqbDqD56SsZDCajlE",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Стаж заболевания",
  type: Type.INTEGER,
  id: "Vll7Lt9cVKBzV3Jughtk",
  stepId: "zeEzYWza8CRcOD6EcLCy"
},
{
  name: "Ингибиторы АПФ / сартаны",
  type: Type.SINGLE_CHOICE,
  options: [
    "зофеноприл",
    "каптоприл",
    "лизиноприл",
    "периндоприл",

```

```

        "рамиприл",
        "фозиноприл",
        "эналаприл",
        "азилсартан",
        "валсартан",
        "ирбесартан",
        "кандесертан",
        "лозартан",
        "олмесартан",
        "телмисартан",
        "эпросартан"
    ],
    id: "LMRRxIpQdcME3h5R53z7",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "IJSvi7JejufUPclttYsA",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Антагонисты кальция",
    type: Type.SINGLE_CHOICE,
    options: [
        "амлодипин",
        "верапамил",
        "дилтиазем",
        "лерканидипин",
        "нимодипин",
        "нитрендипин",
        "нифедипин",
        "фелодипин",
        "циннаризин"
    ],
    id: "7sFEVieZbrEbjy4g8k7m",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "UjDFhrhLBrNjS1NqGsYF",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Диуретики",
    type: Type.SINGLE_CHOICE,
    options: [
        "гипотиазид",
        "индапамид",
        "спиронолактон или верошпирон",
        "торасемид с любым названием",
        "фуросемид или лазикс",
        "эплеренон или эспиро"
    ],
    id: "VADuUzJnpXqMTEE6AYWK",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{

```

```

    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "nebEXePpDLv5zdU4nl0a",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Бета-блокаторы",
    type: Type.SINGLE_CHOICE,
    options: [
      "атенолол",
      "бисопролол",
      "метопролол",
      "небиволол",
      "пропранолол или анаприлин"
    ],
    id: "BqRIzK99ydeT2KJSZnPZ",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "E7U3F5NIX8dur4R2AKzV",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Нитраты",
    type: Type.INTEGER,
    id: "JtxPLIUvy2LY1lB3drPm",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Антикоагулянты",
    type: Type.SINGLE_CHOICE,
    options: [
      "варфарин",
      "ксарелто",
      "прадакса",
      "эликвис"
    ],
    id: "zNlQG70afYdCpNDAopC6",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "r5alW9XmMqHCWuP7v64v",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Последний уровень МНО, если пациент принимает варфарин",
    type: Type.FLOAT,
    id: "XBS0LEiGy9LvZ1UmCZYb",
    stepId: "H7AEj92vFmaCoJmdr167"
  },
  {
    name: "Статины",
    type: Type.SINGLE_CHOICE,
    options: [
      "аторвастатин",

```

```

        "розувастатин",
        "симвастатин"
    ],
    id: "e23U1qGD5jx54NEEmd3X",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "1TFw7feKvLkFfpR4AEpb",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Антиаритмические препараты",
    type: Type.SINGLE_CHOICE,
    options: [
        "аллапинин",
        "амиодарон или кордарон",
        "новокаиномид",
        "пропафенон или пропанорм",
        "соталол или сотагексал",
        "этакцин"
    ],
    id: "P43wRnULhiZ2gK1elOTM",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Суточная дозировка препарата",
    type: Type.FLOAT,
    id: "5reCmOCToZDjRXXMz3VC",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Дигоксин",
    type: Type.BOOLEAN,
    id: "fIYyaZ5jXgYBBH4aEmj3",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Антиагрегенты",
    type: Type.MULTIPLE_CHOICE,
    options: [
        "ацетилсалициловая кислота с любым названием",
        "клопидогрел с любым названием",
        "брилинта",
        "эффиент"
    ],
    id: "wSXubuz3AgSWVOAzrwK2",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Прочие средства",
    type: Type.MULTIPLE_CHOICE,
    options: [
        "моксонидин",
        "предуктал",
        "антациды",
        "ингибиторы протонной помпы",
        "спазмолитики",

```

```

        "бронходилататоры",
        "тирозол",
        "левотироксин или эутирокс",
        "сахароснижающие препараты",
        "инсулин",
        "успокоительные",
        "феназепам",
        "антидепрессанты",
        "корвалол или валокордин",
        "валериана",
        "валидол",
        "глицин"
    ],
    id: "Q23a58DefJkUskS7KirF",
    stepId: "H7AEj92vFmaCoJmdr167"
},
{
    name: "Прикрепить фото ОАК",
    type: Type.PHOTO,
    id: "GrHJFgZxjtGQSJXKotff",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{
    name: "Дата",
    type: Type.DATE,
    id: "bYArwTkM2nRp3M9iIjD7",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{
    name: "Лейкоциты",
    type: Type.FLOAT,
    id: "zJDsnnq2qZfJ0HFycKXn",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{
    name: "Эритроциты",
    type: Type.FLOAT,
    id: "AmAMi6X3tJ1524MHxvgl",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{
    name: "Гемоглобин",
    type: Type.INTEGER,
    id: "UzplteauXQJTEEbUM4md",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{
    name: "Тромбоциты",
    type: Type.INTEGER,
    id: "xMxqTgfQeExNxyX12nbc",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{
    name: "CO2",
    type: Type.INTEGER,
    id: "sO2FuHjVGwbbzmR7sgUg",
    stepId: "3UDmXl59PXgjmb1T8jA5"
},
{

```

```

    name: "Прикрепить фото БАК",
    type: Type.PHOTO,
    id: "uDmLEDcNGybdARcBKbRG",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Дата",
    type: Type.DATE,
    id: "y2EicctXdBWym7WrTaGS",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Белок",
    type: Type.FLOAT,
    id: "yZpcLIgdkhWymFJHlTWK",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Мочевина",
    type: Type.FLOAT,
    id: "1TWxzUXOJcoYwcIDLAFy",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Креатинин",
    type: Type.INTEGER,
    required: true,
    id: "YzD2yQzzXjWa8x8SfHUs",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Глюкоза",
    type: Type.FLOAT,
    required: true,
    id: "dH8WXzotJaTooyqUqVUB",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "АСТ",
    type: Type.FLOAT,
    id: "xFrjRqlsz6RZ0RPufMHI",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "АЛТ",
    type: Type.FLOAT,
    id: "GDuUsVg52LWAOouj4nts",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Холестерин",
    type: Type.FLOAT,
    required: true,
    id: "kbLLuIAN86QvBlqZ1jZX",
    stepId: "KbEQGPMc017WJJTVJNpd"
  },
  {
    name: "Общий билирубин",
    type: Type.FLOAT,

```

```

    id: "D0JICfXKghjMdgrnD55U",
    stepId: "KbEQGPMcOl7WJJTVJNpd"
  },
  {
    name: "ТТГ",
    type: Type.FLOAT,
    id: "L8rzPq20D5qliQ6gSz9t",
    stepId: "KbEQGPMcOl7WJJTVJNpd"
  },
  {
    name: "Прикрепить фото ЭКГ",
    type: Type.PHOTO,
    id: "AKQhb8kDRsXK6CtBeaUU",
    stepId: "WA12G0V7GgzoOdWZ73ds"
  },
  {
    name: "Прикрепить фото ЭХОКГ",
    type: Type.PHOTO,
    id: "Za2RTYJM2qJa9RjfpULt",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Дата",
    type: Type.DATE,
    id: "iuAg6Yl5FsgXSY2bGAm7",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Левое предсердие, продольный размер",
    type: Type.INTEGER,
    id: "5HjrszMVsuTj2oJyrhsP",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Левое предсердие, поперечный размер",
    type: Type.INTEGER,
    id: "hxSIuVaOgGq2ZEx2u56Y",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "КДР",
    type: Type.INTEGER,
    id: "sitZK8DmQu9TRreEuGIA",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "КСП",
    type: Type.INTEGER,
    id: "GSDh7ZTn5wWkT6ziRakG",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "КДО",
    type: Type.INTEGER,
    id: "rL0fPrZK9eWjIFoESrnG",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "КСО",

```

```

    type: Type.INTEGER,
    id: "RyNUQTRpFtcgbLmLnYgI",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "ФВ",
    type: Type.INTEGER,
    id: "tGXmP0WhluKvGDjjqYgm",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "МЖП в диастолу",
    type: Type.INTEGER,
    id: "ZDtqTIwIenBdlY7CJlWO",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "ЗСЛЖ в диастолу",
    type: Type.INTEGER,
    id: "BoEkcod5dda2fOIUr2Kg",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "ММЛЖ",
    type: Type.INTEGER,
    id: "hcKifAtT5QN221r7x0cc",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "ОТС",
    type: Type.FLOAT,
    id: "BnuMc4BT3IRuYlctySG7",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Правое предсердие, продольный размер",
    type: Type.INTEGER,
    id: "xfyTdxizpR2RhrTuwpXg",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Правый желудочек, продольный размер",
    type: Type.INTEGER,
    id: "lLdc5LyMdrFgiy3EoIvw",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Правый желудочек, толщина стенки",
    type: Type.INTEGER,
    id: "BVFd80lAxsNzEygnnzRF",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Легочная артерия, диаметр ствола",
    type: Type.INTEGER,
    id: "mulEqBfC6hMFvTQETHNX",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {

```

```

    name: "СДЛА",
    type: Type.INTEGER,
    id: "rsIjKAnTWsmXuWDUxojg",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Нижняя полая вена, диаметр",
    type: Type.INTEGER,
    id: "4xIUnlbqXpHqnOEIJEGI",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Наличие диастолической дисфункции ЛЖ",
    type: Type.BOOLEAN,
    id: "3KF9gBrkQoHG8mqxEIve",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Регургитация на митральном клапане, степень",
    type: Type.SINGLE_CHOICE,
    options: [
      "1",
      "2",
      "3",
      "4"
    ],
    id: "goGDdExwjmRdGINxdXxj",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Регургитация на аортальном клапане, степень",
    type: Type.SINGLE_CHOICE,
    options: [
      "1",
      "2",
      "3",
      "4"
    ],
    id: "fh13IeXfgv1BIuoPdqxp",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Регургитация на трикуспидальном клапане, степень",
    type: Type.SINGLE_CHOICE,
    options: [
      "1",
      "2",
      "3",
      "4"
    ],
    id: "NaGU2CQKvOWIS0LsJzoN",
    stepId: "9oqrRtcFYqxWpYYffcKy"
  },
  {
    name: "Регургитация на клапане легочной артерии, степень",
    type: Type.SINGLE_CHOICE,
    options: [
      "1",
      "2",

```

```

        "3",
        "4"
    ],
    id: "igImv4nd7hst8vPz72rd",
    stepId: "9oqrRtcFYqxWpYYffcKy"
},
{
    name: "Легочная гипертензия, степень",
    type: Type.SINGLE_CHOICE,
    options: [
        "1",
        "2",
        "3",
        "4"
    ],
    id: "8pCdcIJJeJTk1KyPTjUKr",
    stepId: "9oqrRtcFYqxWpYYffcKy"
},
{
    name: "Прикрепить фото ХМЭКГ",
    type: Type.PHOTO,
    id: "vmEJilfZZMA7hXHKBRCz",
    stepId: "oQsZxAGWhkcXFSqtieft"
},
{
    name: "Дата",
    type: Type.DATE,
    id: "hDsYu60ACRmSRc0MChps",
    stepId: "oQsZxAGWhkcXFSqtieft"
},
{
    name: "ЧСС минимальная за сутки",
    type: Type.INTEGER,
    id: "u0sJdwE5iqImd8WWBxMD",
    stepId: "oQsZxAGWhkcXFSqtieft"
},
{
    name: "ЧСС максимальная за сутки",
    type: Type.INTEGER,
    id: "BxdAiPwp4HmplnDFVahv",
    stepId: "oQsZxAGWhkcXFSqtieft"
},
{
    name: "ЧСС средняя за сутки",
    type: Type.INTEGER,
    id: "XclboLJeCJDsd7UcLA2A",
    stepId: "oQsZxAGWhkcXFSqtieft"
},
{
    name: "Циркадный индекс (ЦИ)",
    type: Type.FLOAT,
    id: "KS96fd9RUslcCWQLvhM6",
    stepId: "oQsZxAGWhkcXFSqtieft"
},
{
    name: "Количество наджелудочковых экстрасистол всего",
    type: Type.INTEGER,
    id: "DNO0115Cg1OPbK42IVC4",
    stepId: "oQsZxAGWhkcXFSqtieft"
}

```

```

    },
    {
      name: "Количество желудочковых экстрасистол всего",
      type: Type.INTEGER,
      id: "TqlA1ow39xbqxGes8cB3",
      stepId: "oQsZxAGWhkcXFSqtieft"
    }
  ];

exports.steps = () => {
  result = {};
  steps.forEach((e) => {
    result[e.id] = e.name;
  });

  return result;
};

exports.attrs = () => {
  result = {};
  attrs.forEach((e) => {
    var clone = Object.assign({}, e);
    delete clone.id;
    result[e.id] = clone;
  });

  return result;
};

exports.Keys = {
  SEX: "zbm9VRuOkCdX5BkwmyyD",
  AGE: "7VWqa7NcyKcwos4jTZkz",
  CHOLESTEROL: "kbLLuIAN86QvBlqZ1jZX",
  SYSTOLIC_BLOOD_PRESSURE: "Rtpr6Q1ZgQxj8lgfllip",
  SIGARETTES: "smvS1V2rNa4tcOXR23To",
  CREATININE: "YzD2yQzzXjWa8x8SfHUs",
  WEIGHT: "kZqXoXv9az0n03fCfmgR",
  HEIGHT: "8xVn3IP8LUN7TTGr1fn8",
};

```

Листинг В.8 – type.js

```

module.exports = {
  PHOTO: 1,
  TEXT: 2,
  DATE: 3,
  INTEGER: 4,
  FLOAT: 5,
  BOOLEAN: 6,
  SINGLE_CHOICE: 7,
  MULTIPLE_CHOICE: 8,
};

```

Листинг В.9 – mainRoute.js

```

'use-strict';

const express = require('express');

```

```

const router = express.Router();
const mainController = require('../controllers/mainController');
const auth = require('../middleware/auth');
const validator = require('../middleware/validator');
const dateFormat = require('dateformat');
const iconv = require('iconv-lite');
const { awareOfAsyncErrors } = require('../helpers');

router.post(
  '/post',
  [auth.authenticate, validator.validateDataForRecord],
  awareOfAsyncErrors(async (req, res) => res.json(await
mainController.post(req, res)))
);

router.post(
  '/rate',
  [auth.authenticate, validator.validateReview],
  awareOfAsyncErrors(async (req, res) => res.json(await
mainController.rate(req, res)))
);

router.get(
  '/statement',
  [auth.authenticate, auth.isAdmin],
  awareOfAsyncErrors((req, res, next) => {
    mainController.statement(req, res)
      .then((result) => iconv.encode(result, 'win1251'))
      .then((buf) => {
        const prettyDate = dateFormat(new Date(), 'yyyy-mm-dd_hh-MM-
ss');
        const filename = `${prettyDate}.csv`;
        res.attachment(filename);
        res.charset = 'win1251';
        return res.send(buf);
      })
      .catch((e) => {
        next(e);
      });
  })
);

module.exports = router;

```

Листинг В.10 – photoRoute.js

```

'use-strict';

const express = require('express');
const router = express.Router();
const photoController = require('../controllers/photoController');
const auth = require('../middleware/auth');
const { awareOfAsyncErrors } = require('../helpers');

router.post(
  '/photo',
  auth.authenticate,
  awareOfAsyncErrors(async (req, res) => res.json(await
photoController.uploadPhoto(req, res)))

```

```
);

router.get(
  '/photo/:photoId',
  awareOfAsyncErrors(async (req, res) => res.redirect(await
photoController.getPhoto(req, res)))
);

module.exports = router;
```

Листинг В.11 – evaluationService.js

```
exports.SCORE = ({ age, sex, cholesterol, systolic, smoker }) => {
  const alpha = [-21.0, -28.7, -25.7, -30.0];
  const p = [4.62, 6.23, 5.47, 6.42];

  const smokerd = [0.71, 0.63];
  const chol = [0.24, 0.02];
  const sbp = [0.018, 0.022];

  let s0 = [];
  for (let i = 0; i <= 2; i += 2) {
    s0.push(Math.exp(-(Math.exp(alpha[i + sex]) * Math.pow(age - 20,
p[i + sex]))));
    s0.push(Math.exp(-(Math.exp(alpha[i + sex]) * Math.pow(age - 10,
p[i + sex]))));
  }

  let w = [];
  for (let i = 0; i < 2; i++) {
    let v = chol[i] * (cholesterol - 6) + sbp[i] * (systolic - 120) +
smokerd[i] * (smoker ? 1 : 0);
    w.push(v);
    w.push(v);
  }

  let s = [];
  for (let i = 0; i < 4; i++) {
    s.push(Math.pow(s0[i], Math.exp(w[i])));
  }

  let s10 = [];
  for (let i = 0; i <= 2; i += 2) {
    s10.push(s[i + 1] / s[i]);
  }

  let risk10 = [];
  for (let i = 0; i < 2; i++) {
    risk10.push(1 - s10[i]);
  }

  let result = risk10[0] + risk10[1];

  return result;
};

exports.CKD_EPI = ({ age, sex, creatinine }) => {
  let a = 0;
  let k = 0;
```

```

let alpha = 0.0;
if (sex === 0) { // Male
  a = 141;
  k = 0.9;
  alpha = creatinine > 80 ? -1.209 : -0.411;
} else { // Female
  a = 144;
  k = 0.7;
  alpha = creatinine > 62 ? -1.209 : -0.329;
}

return a * Math.pow(creatinine / 88.4 / k, alpha) * Math.pow(0.993,
age);
};

exports.BMI = ({ weight, height }) => {
  return weight / Math.pow(height / 100, 2);
};

```

Листинг В.12 – index.js

```

'use-strict';

const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const cors = require('cors')({ origin: true });
const fileParser = require('express-multipart-file-parser');

const admin = require('firebase-admin');
const functions = require('firebase-functions');
const serviceAccount = require('./score-school-firebase-adminsdk-
c72i5-493ccf3b8b.json');
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  storageBucket: 'score-school.appspot.com',
});
const bucket = admin.storage().bucket();
const dataModel = require('./models/data');
const attrs = dataModel.attrs();
const Type = require('./models/type');
const db = admin.firestore();

process.env['NODE-ENV'] = 'development';

const mainRoute = require('./routes/mainRoute');
const photoRoute = require('./routes/photoRoute');
const errorHandler = require('./middleware/errorHandler');

app.use(cors);
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(fileParser);
app.use(mainRoute);
app.use(photoRoute);

app.use(errorHandler);

exports.api = functions.https.onRequest(app);

```

```

exports.onRecordDeleted = functions.firestore
  .document('records/{recordId}')
  .onDelete(async (snap, context) => {

    const data = snap.data().data;
    Object.keys(attrs).forEach(async (id) => {
      if (attrs[id].type === Type.PHOTO && data[id] !== undefined) {
        data[id].forEach(async (photoId) => {
          await db.collection('photos').doc(photoId).delete();
        });
      }
    });
  });

exports.onPhotoDeleted = functions.firestore
  .document('photos/{photoId}')
  .onDelete(async (snap, context) => {

    const photoInfo = snap.data();
    const file = bucket.file(photoInfo.path);
    await file.delete();

  });

exports.scheduledStorageCleanup = functions.pubsub
  .schedule('every day 03:00')
  .onRun(async (context) => {

    const records = await db.collection('records').get();
    const photos = await db.collection('photos').get();

    photos.forEach(async (photo) => {
      const createdAt = photo.data().createdAt.toDate().getTime();
      let shoudDelete = createdAt < (Date.now() - 3 * 60 * 60 * 24 *
1000); // if photo > 3 days old

      if (shoudDelete) {
        records.forEach((record) => {
          const data = record.data().data;
          Object.keys(attrs).forEach((id) => {
            if (attrs[id].type === Type.PHOTO && data[id] !==
undefined && data[id].indexOf(photo.id) !== -1) {
              shoudDelete = false
            }
          });
        });
      }

      if (shoudDelete) {
        await db.collection('photos').doc(photo.id).delete();
      }
    });
  });

```

Листинг В.13 – test/index.js

```
/* eslint-disable prefer-arrow-callback */
```

```

const assert = require('assert');
const { SCORE, CKD_EPI, BMI } =
require('../app/services/evaluationService');

describe('Расчеты', () => {
  describe('SCORE', () => {
    it('Риск равен 4.34%, если некурящий мужчина 60 лет, со ср. сист. АД 120 мм.рт.ст и холестерином 6 ммоль/л', () => {
      const result = SCORE({
        age: 60,
        sex: 0,
        systolic: 120,
        smoker: false,
        cholesterol: 6,
      });
      assert.strictEqual((result * 100).toFixed(2), '4.34');
    });

    it('Риск равен 6.65%, если курящий мужчина 57 лет, со ср. сист. АД 120 мм.рт.ст и холестерином 6 ммоль/л', () => {
      const result = SCORE({
        age: 57,
        sex: 0,
        systolic: 120,
        smoker: true,
        cholesterol: 6,
      });
      assert.strictEqual((result * 100).toFixed(2), '6.65');
    });

    it('Риск равен 0.21%, если курящая женщина 45 лет, со ср. сист. АД 105 мм.рт.ст и холестерином 4 ммоль/л', () => {
      const result = SCORE({
        age: 45,
        sex: 1,
        systolic: 105,
        smoker: true,
        cholesterol: 4,
      });
      assert.strictEqual((result * 100).toFixed(2), '0.21');
    });

    it('Риск равен 1.1%, если некурящая женщина 60 лет, со ср. сист. АД 110 мм.рт.ст и холестерином 5 ммоль/л', () => {
      const result = SCORE({
        age: 60,
        sex: 1,
        systolic: 110,
        smoker: false,
        cholesterol: 5,
      });
      assert.strictEqual((result * 100).toFixed(2), '1.10');
    });

    it('Риск равен 49.07%, если курящий мужчина 65 лет, со ср. сист. АД 180 мм.рт.ст и холестерином 8 ммоль/л', () => {
      const result = SCORE({
        age: 65,
        sex: 0,

```

```

        systolic: 180,
        smoker: true,
        cholesterol: 8,
    });
    assert.strictEqual((result * 100).toFixed(2), '49.07');
});
});

describe('CKD-EPI', () => {
    it('СКФ равна 108 мл/мин/1,73м2, если мужчина 55 лет, с креатинином 60 мкмоль/л', () => {
        const result = CKD_EPI({
            age: 55,
            sex: 0,
            creatinine: 60,
        });
        assert.strictEqual(Math.round(result), 108);
    });

    it('СКФ равна 80 мл/мин/1,73м2, если мужчина 60 лет, с креатинином 90 мкмоль/л', () => {
        const result = CKD_EPI({
            age: 60,
            sex: 0,
            creatinine: 90,
        });
        assert.strictEqual(Math.round(result), 80);
    });

    it('СКФ равна 37 мл/мин/1,73м2, если мужчина 50 лет, с креатинином 180 мкмоль/л', () => {
        const result = CKD_EPI({
            age: 50,
            sex: 0,
            creatinine: 180,
        });
        assert.strictEqual(Math.round(result), 37);
    });

    it('СКФ равна 53 мл/мин/1,73м2, если женщина 67 лет, с креатинином 96 мкмоль/л', () => {
        const result = CKD_EPI({
            age: 67,
            sex: 1,
            creatinine: 96,
        });
        assert.strictEqual(Math.round(result), 53);
    });

    it('СКФ равна 43 мл/мин/1,73м2, если женщина 76 лет, с креатинином 108 мкмоль/л', () => {
        const result = CKD_EPI({
            age: 76,
            sex: 1,
            creatinine: 108,
        });
        assert.strictEqual(Math.round(result), 43);
    });
});

```

```
describe('BMI', () => {
  it('Индекс массы тела равен 28.7, если человек с ростом 180 см и
весом 93 кг', () => {
    const result = BMI({
      weight: 93,
      height: 180,
    });
    assert.strictEqual(result.toFixed(1), '28.7');
  });

  it('Индекс массы тела равен 41.5, если человек с ростом 170 см и
весом 120 кг', () => {
    const result = BMI({
      weight: 120,
      height: 170,
    });
    assert.strictEqual(result.toFixed(1), '41.5');
  });
});
```

Приложение Г
ЛИСТИНГ КЛИЕНТСКОЙ ЧАСТИ
(рекомендуемое)

Листинг Г.1 – LoginRepository.kt

```
package com.example.medicalapp.data

import com.example.medicalapp.data.model.Volunteer
import com.google.firebase.auth.FirebaseUser
import java.lang.Exception

/**
 * Class that requests authentication and user information from the
 * remote data source and
 * maintains an in-memory cache of login status and user credentials
 * information.
 */

object LoginRepository {

    private val dataSource = LoginDataSource()

    // in-memory cache of the loggedInUser object
    var user: FirebaseUser? = null
        private set

    val isLoggedIn: Boolean
        get() = user != null

    init {
        // If user credentials will be cached in local storage, it is
        recommended it be encrypted
        // @see
        https://developer.android.com/training/articles/keystore
        user = null
    }

    fun logout() {
        user = null
        dataSource.logout()
    }

    suspend fun login(username: String, password: String): Volunteer {
        val user = dataSource.login(username, password)

        user?.let {
            setLoggedInUser(it)
            return Volunteer(user.uid, user.displayName)
        }

        throw Exception("lox pidor")
    }

    private fun setLoggedInUser(loggedInUser: FirebaseUser) {
        this.user = loggedInUser
        // If user credentials will be cached in local storage, it is
        recommended it be encrypted
        // @see
        https://developer.android.com/training/articles/keystore
    }
}
```

Листинг Г.2 – Repository.kt

```

package com.example.medicalapp.data

import com.example.medicalapp.data.model.ResultResponse
import com.google.android.gms.tasks.Tasks
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import java.io.IOException
import com.example.medicalapp.data.Result.Success
import com.example.medicalapp.data.Result.Error
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.RequestBody.Companion.asRequestBody
import retrofit2.http.*
import java.io.File

object Repository {

    suspend fun post(values: Map<String, Any>): Result<ResultResponse>
    {
        val data = mutableListOf<HashMap<String, *>>()
        values.forEach { input ->
            data.add(hashMapOf("id" to input.key, "value" to
input.value))
        }
        val request = DataRequest(data)
        val response = API.apiService.postData(request)
        if (response.isSuccessful) {
            val result = response.body() ?: return
Error(Exception("null"))
            return Success(result)
        } else {
            val gson = Gson()
            val type = object : TypeToken<ErrorResponse>() {}.type
            val ErrorResponse: ErrorResponse? =
gson.fromJson(response.errorBody()?.charStream(), type)
            return Error(Exception(ErrorResponse?.msg ?: "Error
occurred"))
        }
    }

    suspend fun sendPhoto(photoPath: String): Result<String> {
        val file = File(photoPath)
        val body = MultipartBody.Part.createFormData(
            "photo",
            file.name,
            file.asRequestBody("image/*".toMediaTypeOrNull())
        )

        val response = API.apiService.sendPhoto(body)
        if (response.isSuccessful) {

```

```

        val result = response.body()?.id ?: return
        Error(Exception("null"))
        return Success(result)
    } else {
        val gson = Gson()
        val type = object : TypeToken<ErrorResponse>() {}.type
        val errorResponse: ErrorResponse? =
gson.fromJson(response.errorBody()?.charStream(), type)
        return Error(Exception(errorResponse?.msg ?: "Error
occurred"))
    }
}

suspend fun sendReview(recordId: String, aware: Boolean, helpful:
Boolean, rate: Int): Result<Unit> {
    val response = API.apiService.sendReview(recordId,
ReviewRequest(aware, helpful, rate))
    if (response.isSuccessful) {
        val result = response.body() ?: return
        Error(Exception("null"))
        return Success(result)
    } else {
        val gson = Gson()
        val type = object : TypeToken<ErrorResponse>() {}.type
        val errorResponse: ErrorResponse? =
gson.fromJson(response.errorBody()?.charStream(), type)
        return Error(Exception(errorResponse?.msg ?: "Error
occurred"))
    }
}

}

data class DataRequest(val data: List<Map<String, *>>)

data class ErrorResponse(val msg: String)

data class PhotoResponse(val id: String)

data class ReviewRequest(val aware: Boolean, val helpful: Boolean, val
rate: Int)

object API {
    private const val API_BASE_URL = "https://us-central1-score-
school.cloudfunctions.net/api/"
    // private const val API_BASE_URL =
"http://192.168.1.151:5001/score-school/us-central1/api/"

    private val httpClient = OkHttpClient.Builder().apply {
        addInterceptor(
            HttpLoggingInterceptor().apply { level =
HttpLoggingInterceptor.Level.HEADERS }
        )
        addNetworkInterceptor(AuthInterceptor())
    }

    private val builder: Retrofit.Builder = Retrofit.Builder()
        .baseUrl(API_BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())

```

```

        private val retrofit: Retrofit = builder
            .client(httpClient.build())
            .build()

        val apiService: ApiService =
            retrofit.create(ApiService::class.java)

        interface ApiService {
            @POST("post")
            suspend fun postData(@Body request: DataRequest):
                retrofit2.Response<ResultResponse>

            @Multipart
            @POST("photo")
            suspend fun sendPhoto(@Part photo: MultipartBody.Part):
                retrofit2.Response<PhotoResponse>

            @POST("rate")
            suspend fun sendReview(@Query("id") id: String, @Body request:
                ReviewRequest): retrofit2.Response<Unit>
        }

        class AuthInterceptor : Interceptor {

            private val auth: FirebaseAuth by lazy { FirebaseAuth }

            override fun intercept(chain: Interceptor.Chain): Response {
                var request = chain.request()

                try {
                    val user = auth.currentUser ?: throw Exception("user
is not logged in.")
                    val tokenResult =
                        Tasks.await(user.getIdToken((false)))
                    val token = tokenResult.token ?: throw
                        Exception("idToken is null.")

                    request = request
                        .newBuilder()
                        .header("Authorization", "Bearer $token")
                        .build()

                    return chain.proceed(request)
                } catch (e: Exception) {
                    throw IOException(e.message)
                }
            }
        }
    }
}

```

Листинг Г.3 – Prefs.kt

```

package com.example.medicalapp.data

import android.content.Context
import android.content.SharedPreferences

```

```

class Prefs(context: Context) {

    private val preferences: SharedPreferences
        = context.getSharedPreferences(APP_PREF_NAME,
Context.MODE_PRIVATE)

    var firstName: String
        get() = preferences.getString(FIRST_NAME_KEY, "")!!
        set(value) = preferences.edit().putString(FIRST_NAME_KEY,
value).apply()

    var lastName: String
        get() = preferences.getString(LAST_NAME_KEY, "")!!
        set(value) = preferences.edit().putString(LAST_NAME_KEY,
value).apply()

    var patronymic: String
        get() = preferences.getString(PATRONYMIC_KEY, "")!!
        set(value) = preferences.edit().putString(PATRONYMIC_KEY,
value).apply()

    companion object {

        private const val APP_PREF_NAME = "CardioguidePref"

        private const val FIRST_NAME_KEY = "firstName"
        private const val LAST_NAME_KEY = "lastName"
        private const val PATRONYMIC_KEY = "patronymic"

    }
}

```

Листинг Г.4 – Result.kt

```

package com.example.medicalapp.data

import com.example.medicalapp.data.Result.Success

/**
 * A generic class that holds a value with its loading status.
 * @param <T>
 */
sealed class Result<out R> {

    data class Success<out T : Any>(val data: T) : Result<T>()
    data class Error(val exception: Exception) : Result<Nothing>()

    override fun toString(): String {
        return when (this) {
            is Success<*> -> "Success[data=$data]"
            is Error -> "Error[exception=$exception]"
        }
    }
}

val Result<*>.succeeded
    get() = this is Success && data != null

```

Листинг Г.5 – Task.kt

```

package com.example.medicalapp.data

import com.google.android.gms.tasks.Task
import kotlinx.coroutines.CancellationException
import kotlinx.coroutines.suspendCancellableCoroutine
import kotlin.coroutines.resume
import kotlin.coroutines.resumeWithException

suspend fun <T> Task<T>.await(): T {
    if (isComplete) {
        val e = exception
        return if (e == null) {
            if (isCanceled) {
                throw CancellationException("Task $this was cancelled
normally.")
            } else {
                result!!
            }
        } else {
            throw e
        }
    }

    return suspendCancellableCoroutine { cont ->
        addOnCompleteListener {
            val e = exception
            if (e == null) {
                if (isCanceled) cont.cancel() else
cont.resume(result!!)
            } else {
                cont.resumeWithException(e)
            }
        }
    }
}

```

Листинг Г.6 – BackdropBehavior.kt

```

package com.example.medicalapp.ui.form.backdrop

import android.content.Context
import android.os.Bundle
import android.os.Parcelable
import android.util.AttributeSet
import android.view.MotionEvent
import android.view.View
import android.view.ViewGroup
import android.view.ViewPropertyAnimator
import android.view.animation.AccelerateDecelerateInterpolator
import androidx.annotation.IdRes
import androidx.appcompat.widget.Toolbar
import androidx.coordinatorlayout.widget.CoordinatorLayout
import androidx.core.view.children
import com.example.views.R

```

```

import com.example.views.margin

class BackdropBehavior : CoordinatorLayout.Behavior<View> {

    enum class DropState {
        OPEN,
        CLOSE
    }

    interface OnDropListener {

        fun onDrop(dropState: DropState, fromUser: Boolean)
    }

    companion object {
        private const val DEFAULT_DURATION = 200L
        private const val WITHOUT_DURATION = 0L
        private val DEFAULT_DROP_STATE = DropState.CLOSE

        private const val ARG_DROP_STATE = "arg_drop_state"
    }

    private var utils = BackdropUtils()

    private var toolbarId: Int? = null
    private var backLayoutId: Int? = null

    private var toolbar: Toolbar? = null
    private var backLayout: ViewGroup? = null
    private var frontLayout: ViewGroup? = null

    private var topPosition: Int = 0
    private var bottomPosition: Int = 0
    private var topInset: Int = 0
    private var bottomInset: Int = 0

    private var iconRes: Int = R.drawable.ic_close

    private var dropState: DropState = DEFAULT_DROP_STATE

    private var needToInitializing = true

    private var frontViewDown = false
    private var frontViewMove = false

    private var dropListeners = mutableListOf<OnDropListener>()

    constructor() : super()

    constructor(context: Context, attrs: AttributeSet) :
super(context, attrs)

    override fun onSaveInstanceState(parent: CoordinatorLayout, child:
View): Parcelable {
        return Bundle().apply {
            putSerializable(ARG_DROP_STATE, dropState)
        }
    }
}

```

```

        override fun onRestoreInstanceState(parent: CoordinatorLayout,
            child: View, state: Parcelable) {
            super.onRestoreInstanceState(parent, child, state)

            dropState =
                (state as? Bundle)?.getSerializable(ARG_DROP_STATE) as?
                DropState ?: DEFAULT_DROP_STATE
        }

        override fun layoutDependsOn(
            parent: CoordinatorLayout,
            child: View,
            dependency: View
        ): Boolean {
            if (toolbarId == null && backLayoutId == null) return false

            return when (dependency.id) {
                toolbarId -> true
                backLayoutId -> true
                else -> false
            }
        }

        override fun onInterceptTouchEvent(
            parent: CoordinatorLayout,
            child: View,
            event: MotionEvent
        ): Boolean {
            if (dropState == DropState.CLOSE) {
                return false
            }

            if (event.action == MotionEvent.ACTION_DOWN) {
                frontLayout?.let {
                    frontViewDown = event.y >= it.y
                }
            }

            return frontViewDown
        }

        override fun onTouchEvent(parent: CoordinatorLayout, child: View,
            event: MotionEvent): Boolean {
            when (event.action) {
                MotionEvent.ACTION_UP, MotionEvent.ACTION_CANCEL -> {
                    if (frontViewDown && !frontViewMove) {
                        close()
                    }

                    frontViewMove = false
                    frontViewDown = false
                }
                MotionEvent.ACTION_MOVE -> {
                    frontViewMove = true
                }
            }

            return frontViewDown
        }

```

```

    }

    override fun onDependentViewChanged(
        parent: CoordinatorLayout,
        child: View,
        dependency: View
    ): Boolean {

        this.frontLayout = child as? ViewGroup
            ?: throw IllegalArgumentException("BackLayout must extend
a ViewGroup")

        when (dependency.id) {
            toolbarId -> toolbar = dependency as? Toolbar
                ?: throw IllegalArgumentException("toolbarId doesn't
match Toolbar")

            backLayoutId -> {
                backLayout = dependency as? ViewGroup
                    ?: throw IllegalArgumentException("backLayoutId
doesn't match back Layout")

                // TODO (next release): remove this conditional
                if (toolbarId == null) {
                    toolbar = utils.findToolbar(backLayout!!)
                    ?: throw
IllegalArgumentException("AppBarLayout mast contain a Toolbar!")
                }
            }
        }

        if (toolbar != null && frontLayout != null && backLayout !=
null && needToInitializing) {
            initViews(parent, frontLayout!!, toolbar!!, backLayout!!)
        }

        val a = super.onDependentViewChanged(parent, child,
dependency)

        return a
    }

    fun setIcon(@IdRes iconRes: Int) {
        this.iconRes = iconRes
    }

    fun setTopInset(topInset: Int) {
        this.topInset = topInset
        needToInitializing = true
    }

    fun setBottomInset(bottomInset: Int) {
        this.bottomInset = bottomInset
        needToInitializing = true
    }

    /**
     * Attach back layout to Backdrop.
     * BackDropLayout must contain a [Toolbar]

```

```

    */
    fun attachBackLayout(@IdRes appBarLayoutId: Int) {
        this.backLayoutId = appBarLayoutId
    }

    /**
     * @deprecated – use [BackdropBehavior.attachBackLayout]. This
     method will be removed in version 0.1.7+
     */
    @Deprecated("Use BackdropBehavior.attachBackLayout")
    fun attachToolbar(@IdRes toolbarId: Int) {
        this.toolbarId = toolbarId
    }

    /**
     * @deprecated – use [BackdropBehavior.attachBackLayout]. This
     method will be removed in version 0.1.7+
     */
    @Deprecated("Use BackdropBehavior.attachBackLayout")
    fun attachBackContainer(@IdRes backContainerId: Int) {
        this.backLayoutId = backContainerId
    }

    fun addOnDropListener(listener: OnDropListener) {
        dropListeners.add(listener)
    }

    fun removeDropListener(listener: OnDropListener) {
        dropListeners.remove(listener)
    }

    fun open(withAnimation: Boolean = true): Boolean = if (dropState
    == DropState.OPEN) {
        false
    } else {
        dropState = DropState.OPEN
        if (backLayout != null && toolbar != null && frontLayout !=
        null) {
            drawDropState(frontLayout!!, toolbar!!, backLayout!!,
            withAnimation)
        } else {
            throw IllegalArgumentException("Toolbar and backContainer
            must be initialized")
        }
        notifyListeners(false)
        true
    }

    fun close(withAnimation: Boolean = true): Boolean = if (dropState
    == DropState.CLOSE) {
        false
    } else {
        dropState = DropState.CLOSE
        if (backLayout != null && toolbar != null && frontLayout !=
        null) {
            drawDropState(frontLayout!!, toolbar!!, backLayout!!,
            withAnimation)
        } else {

```

```

        throw IllegalArgumentException("Toolbar and backContainer
must be initialized")
    }
    notifyListeners(false)
    true
}

private fun initViews(
    parent: CoordinatorLayout,
    frontLayout: ViewGroup,
    toolbar: Toolbar,
    backLayout: ViewGroup
) {

    toolbar.margin(top = topInset)

    topPosition = backLayout.top + toolbar.height + topInset
    bottomPosition = backLayout.top + backLayout.height
    if (bottomPosition > parent.height - bottomInset) {
        bottomPosition = parent.height - bottomInset
        if (this.frontLayout is FrontLayout) {
            bottomPosition -= (this.frontLayout!! as
FrontLayout).getSubheaderHeight()
        }
    }
    backLayout.layoutParams.height = bottomPosition -
backLayout.top
    frontLayout.layoutParams.height = parent.height - topPosition
    drawDropState(frontLayout, toolbar, backLayout, false)

    with(toolbar) {
        setNavigationIcon(iconRes)
        setNavigationOnClickListener {
//            dropState = when (dropState) {
//                DropState.CLOSE -> DropState.OPEN
//                DropState.OPEN -> DropState.CLOSE
//            }
            dropState = DropState.OPEN
            drawDropState(frontLayout, toolbar, backLayout)
            notifyListeners(true)
        }
    }

    needToInitializing = false
}

private fun drawDropState(
    frontLayout: ViewGroup,
    toolbar: Toolbar,
    backContainer: ViewGroup,
    withAnimation: Boolean = true
) {
    when (dropState) {
        DropState.CLOSE -> {
            drawClosedState(frontLayout, backContainer, toolbar,
withAnimation)
//            toolbar.setNavigationIcon(closedIconId)
        }
        DropState.OPEN -> {

```

```

        drawOpenedState(frontLayout, backContainer, toolbar,
withAnimation)
//        toolbar.setNavigationIcon(openedIconRes)
    }
}

private fun drawClosedState(
    frontLayout: ViewGroup,
    backLayout: ViewGroup,
    toolbar: Toolbar,
    withAnimation: Boolean = true
) {
    val position = topPosition.toFloat()
    val duration = if (withAnimation) DEFAULT_DURATION else
WITHOUT_DURATION

    frontLayout
        .animate()
        .setInterpolator(AccelerateDecelerateInterpolator())
        .withDim()
        .y(position)
        .setDuration(duration)
        .start()

    utils.fadeIn(toolbar, duration)
    for (child in backLayout.children) {
        if (child !is Toolbar) {
            utils.fadeOut(child, duration)
        }
    }

    if (frontLayout is FrontLayout) {
        frontLayout.backgroundDimAlpha = 0.0f
    }
}

private fun drawOpenedState(
    frontLayout: ViewGroup,
    backLayout: ViewGroup,
    toolbar: Toolbar,
    withAnimation: Boolean = true
) {
    val position = bottomPosition.toFloat()
    val duration = if (withAnimation) DEFAULT_DURATION else
WITHOUT_DURATION

    frontLayout
        .animate()
        .setInterpolator(AccelerateDecelerateInterpolator())
        .withDim()
        .y(position)
        .setDuration(duration)
        .start()

    utils.fadeOut(toolbar, duration)
    for (child in backLayout.children) {
        if (child !is Toolbar) {
            utils.fadeIn(child, duration)
        }
    }
}

```

```

        }
    }
}

private fun ViewPropertyAnimator.withDim(): ViewPropertyAnimator {
    if (frontLayout is FrontLayout) {
        val view = frontLayout as FrontLayout
        val top = topPosition
        val bottom = bottomPosition
        setUpdateListener {
            val currentPosition = view.y
            val alpha = 0.6f * ((currentPosition - top) / (bottom
- top))
            view.backgroundDimAlpha = alpha
        }
    }

    return this
}

// private fun getBottomPosition(): Float = bottomPosition -

private fun notifyListeners(fromUser: Boolean) {
    dropListeners.forEach { listener ->
        listener.onDrop(dropState, fromUser)
    }
}
}

```

Листинг Г.7 – BottomSheet.kt

```

package com.example.medicalapp.ui.form

import android.os.Build
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.annotation.IdRes
import androidx.annotation.LayoutRes
import androidx.core.view.children
import com.example.medicalapp.R
import com.example.medicalapp.dpToPx
import com.example.medicalapp.padding
import
com.google.android.material.bottomsheet.BottomSheetDialogFragment
import kotlinx.android.synthetic.main.bottom_sheet.*

class BottomSheet(@LayoutRes val resource: Int) :
    BottomSheetDialogFragment() {

    lateinit var itemClickListener : (view: View) -> Unit

    override fun onCreateView(inflater: LayoutInflater, container:
    ViewGroup?, savedInstanceState: Bundle?): View {
        return (inflater.inflate(resource, container) as
    ViewGroup).apply {
            children.forEach { view ->
                view.setOnClickListener {

```

```

        itemClickListener(it)
        dialog?.dismiss()
    }
}

}

}

@Suppress("DEPRECATION")
override fun onStart() {
    super.onStart()
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        dialog?.window?.apply {

findViewById<View>(com.google.android.material.R.id.container).setOnAp
plyWindowInsetsListener { _, insets ->
            content.padding(bottom =
insets.systemWindowInsetBottom + 8.dpToPx())

            insets.consumeSystemWindowInsets()
        }
        val decorView = decorView
        decorView.systemUiVisibility =
decorView.systemUiVisibility or
            View.SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR
        }
    }
}
}

```

Листинг Г.8 – ContentFragment.kt

```

package com.example.medicalapp.ui.form

import android.app.Activity.RESULT_OK
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.os.Environment
import android.provider.MediaStore
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.content.FileProvider
import androidx.core.os.bundleOf
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProviders
import com.example.medicalapp.*
import kotlinx.android.synthetic.main.content_form.*
import kotlinx.android.synthetic.main.content_form.view.*
import java.io.File
import java.io.IOException
import java.util.*

class ContentFragment : Fragment() {

    private lateinit var viewModel: FormViewModel

    private var photoInputId: String? = null

```

```

        private var photoFileName: String? = null

        override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View? {
            return inflater.inflate(R.layout.content_form, container,
false).apply {
                applyInsets(this)
            }
        }

        override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
            super.onViewCreated(view, savedInstanceState)

            val factory =
FormViewModelFactory(requireActivity().application, this,
savedInstanceState)
            viewModel = ViewModelProviders.of(requireActivity(),
factory).get(FormViewModel::class.java)

            val selected = viewModel.selectedStep.value ?: 0
            val data = viewModel.data[selected]

            val adapter = ControlAdapter(requireActivity(),
data.attributes) { id, value ->
                viewModel.onInputChanged(id, value)
            }
            adapter.onAddPhotoButtonClickListener = { id ->
                photoInputId = id
                showBottomSheet(id)
            }
            adapter.onRemovePhotoButtonClickListener = { id, photo ->
                viewModel.removePhoto(id, photo)
            }
            flowLayout.adapter = adapter

            viewModel.photos.observe(viewLifecycleOwner) { photos ->
                photos.forEach {
                    adapter.addPhoto(it.key, it.value)
                }
            }

            //            val isLast = viewModel.data.size - 1 == selected
            //            if (isLast) {
            //                nextBtn.visibility = View.GONE
            //            }
            //            nextBtn.setOnClickListener {
            //                viewModel.nextStep()
            //            }
        }

        private fun dispatchTakePictureIntent(filename: String) {
            Intent(MediaStore.ACTION_IMAGE_CAPTURE).also {
                takePictureIntent ->

                takePictureIntent.resolveActivity(requireContext().packageManager)?.al
so {
                    val photoFile: File? = try {
                        createImageFile(filename, requireContext())

```

```

    } catch (ex: IOException) {
        // Error occurred while creating the File
        null
    }

    photoFile?.also {
        photoFileName = it.absolutePath

        val photoUri = FileProvider.getUriForFile(
            requireContext(),
            "com.example.serviceland.fileprovider",
            it
        )

        takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri)
        startActivityResult(takePictureIntent,
            ControlAdapter.REQUEST_IMAGE_CAPTURE
        )
    }
}

}

}

private fun createImageFile(filename: String, context: Context):
File {
    val storageDir =
context.getExternalFilesDir(Environment.DIRECTORY_PICTURES)
    return File.createTempFile(
        filename,
        ".jpg",
        storageDir
    )
}

private fun dispatchSelectPhotoIntent() {
    Intent(Intent.ACTION_GET_CONTENT).apply {
        putExtra(Intent.EXTRA_ALLOW_MULTIPLE, false)
        type = "image/*"
        startActivityResult(Intent.createChooser(this, "Выбрать
фото"), ControlAdapter.REQUEST_SELECT_IMAGE)
    }
}

private fun showBottomSheet(id: String) {
    val sheet = BottomSheet(R.layout.photo_bottom_sheet)
    sheet.itemClickListener = { view ->
        when (view.id) {
            R.id.takePhoto -> {
                dispatchTakePictureIntent(id + Date().time)
            }
            R.id.selectPhoto -> {
                dispatchSelectPhotoIntent()
            }
        }
    }
    sheet.show(requireActivity().supportFragmentManager,
        PHOTO_BOTTOM_SHEET_TAG)
}

```

```

        override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
            val id = photoInputId ?: return

            when(requestCode) {
                ControlAdapter.REQUEST_IMAGE_CAPTURE -> {
                    if (resultCode == RESULT_OK) {
                        val filename = photoFileName ?: return
                        viewModel.addPhoto(id, filename)
                        photoFileName = null
                    }
                }
                ControlAdapter.REQUEST_SELECT_IMAGE -> {
                    if (resultCode == RESULT_OK) {
//                        viewModel.addPhoto(id, data?.data.toString())
                        val selectedPhotoUri = data?.data ?: return
                        val input =
requireContext().contentResolver.openInputStream(selectedPhotoUri) ?:
return
                        val photoFile = createImageFile(id + Date().time,
requireContext())
                        val photoUri = FileProvider.getUriForFile(
                            requireContext(),
                            "com.example.serviceland.fileprovider",
                            photoFile
                        )
                        val output =
requireContext().contentResolver.openOutputStream(photoUri) ?: return
//                        val output =
FileOutputStream(createImageFile(filename, requireContext()))
                        input.copyTo(output, 1024)
                        input.close()
                        output.close()

                        viewModel.addPhoto(id, photoFile.absolutePath)
                    }
                }
            }
        }

        @Suppress("DEPRECATION")
        private fun applyInsets(view: View) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT_WATCH)
            {
                view.flowLayout.setOnApplyWindowInsetsListener { _, insets
->
                    view.padding(
                        bottom = insets.systemWindowInsetBottom +
8.dpToPx(),
                        left = insets.systemWindowInsetLeft,
                        right = insets.systemWindowInsetRight
                    )
                    insets.copy(bottom = 0)
                }
            }
        }

        companion object {

```

```

        private const val ID_ARG = "id_arg"
        private const val RECYCLER_VIEW_STATE_ARG =
"recycler_view_state"
        private const val PHOTO_BOTTOM_SHEET_TAG =
"photo_bottom_sheet"

        fun newInstance(id: Int) = ContentFragment().apply {
            arguments = bundleOf(ID_ARG to id)
        }
    }
}

```

Листинг Г.9 – FromActivity.kt

```

package com.example.medicalapp.ui.form

import android.app.Activity
import android.content.Intent
import android.graphics.Color
import android.os.Build
import android.os.Bundle
import android.text.Spannable
import android.text.SpannableString
import android.text.style.ForegroundColorSpan
import android.view.Menu
import android.view.MenuItem
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProviders
import com.example.medicalapp.*
import com.example.medicalapp.ui.login.LoginActivity
import com.example.medicalapp.ui.report.ReportActivity
import com.example.medicalapp.ui.form.backdrop.BackdropBehavior
import com.example.medicalapp.ui.form.backdrop.StepsAdapter
import kotlinx.android.synthetic.main.activity_form.*
import kotlinx.android.synthetic.main.activity_form.frontLayout
import kotlinx.android.synthetic.main.back_layer.*
import kotlinx.android.synthetic.main.step_item.view.*

class FormActivity : AppCompatActivity() {

    companion object {
        private const val FRAGMENT_CONTAINER = R.id.content
        private const val FRAGMENT_TAG_PREFIX = "content_"
        private const val BOTTOM_SHEET_TAG = "bottom_sheet"
        private const val SELECTED_STEP_ARG = "selected_step"

        private const val INPUTS_ARG = "inputs"
        private const val PHOTOS_ARG = "photos"
    }

    private lateinit var viewModel: FormViewModel

    private lateinit var backdropBehavior: BackdropBehavior
    private lateinit var stepsAdapter: StepsAdapter

```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    setDecorBehindSystemWindows()

    setContentView(R.layout.activity_form)
    setSupportActionBar(toolbar)
    applyInsets()

    val factory = FormViewModelFactory(application, this,
savedInstanceState)
    viewModel = ViewModelProviders.of(this,
factory).get(FormViewModel::class.java)

    //      var selected = 0
    //      savedInstanceState?.let {
    //          selected = it.getInt(SELECTED_STEP_ARG)
    //          viewModel.selectStep(selected)
    //      }

    savedInstanceState?.let {
        viewModel.inputs = (it.getSerializable(INPUTS_ARG) as
HashMap<String, Any>?) ?: return@let
        val photos = (it.getSerializable(PHOTOS_ARG) as
HashMap<String, List<Pair<String, String?>>?) ?: return@let
        viewModel.setPhotos(photos)
    }

    val selected = viewModel.selectedStep.value ?: 0
    stepsAdapter = StepsAdapter(this, viewModel.data, selected) {
index ->
        backdropBehavior.close()
        viewModel.selectStep(index)
    }
    with(steps) {
        adapter = stepsAdapter
        addItemDecoration(stepsAdapter.JoinItemDecoration())
    }

    backdropBehavior = frontLayout.findBehavior()
    backdropBehavior.attachBackLayout(R.id.backLayout)

    viewModel.selectedStep.observe(this) { index ->
        if (index == -1) {
            return@observe resetFragmentManager()
        }
        stepsAdapter.selected = index
        val total = viewModel.data.size
        supportActionBar?.title = "Шаг ${index + 1} из $total"

        val step = viewModel.data[index]
        if (step.containsRequired) {
            val text = step.name
            val spannable = SpannableString("$text*")
            spannable.setSpan(ForegroundColorSpan(Color.RED),
text.length, text.length + 1, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE)
            frontLayout.subheader.setText(spannable,
TextView.BufferType.SPANNABLE)

```

```

        } else {
            frontLayout.subheader.text = step.name
        }

        showPage(index)
    }

    viewModel.errorMessage.observe(this) {
        toast(it)
    }

    //        viewModel.result.observe(this) {
    //            startActivity(ReportActivity.getIntent(this, it))
    //        }

    rollUp.setOnClickListener {
        backdropBehavior.close()
    }

}

override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)

    outState.putSerializable(INPUTS_ARG, viewModel.inputs)
    outState.putSerializable(PHOTOS_ARG, viewModel.photos.value)
    outState.putInt(SELECTED_STEP_ARG,
viewModel.selectedStep.value ?: 0)
}

@Suppress("DEPRECATION")
private fun applyInsets() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT_WATCH)
    {
        backdrop.setOnApplyWindowInsetsListener { _, insets ->
//            view.padding(top = insets.systemWindowInsetTop)
            val top = insets.systemWindowInsetTop + 8.dpToPx()
            steps.padding(top = top)
            rollUp.margin(top = top)

backdropBehavior.setTopInset(insets.systemWindowInsetTop)

backdropBehavior.setBottomInset(insets.systemWindowInsetBottom)
            insets.copy(top = 0)
        }
    }
}

override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    when (requestCode) {
        ReportActivity.REPORT_REQUEST -> {
            if (resultCode == Activity.RESULT_OK) {
//                clearInput()
            }
        }
    }
}

```

```

    }

    private fun clearInput() {
        viewModel.clear()
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.menu_form, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item.itemId) {
            R.id.more -> {
                showBottomSheet()
            }
        }
        return true
    }

    private fun showBottomSheet() {
        val sheet = BottomSheet(R.layout.bottom_sheet)
        sheet.itemClickListener = { view ->
            when (view.id) {
                R.id.complete -> {
                    if (viewModel.canComplete()) {
                        viewModel.saveDraft()
                        startActivityForResult(
                            ReportActivity.getIntent(this,
viewModel.inputs),
                            ReportActivity.REPORT_REQUEST
                        )
                    }
                }
                R.id.clear -> {
                    clearInput()
                }
                R.id.exit -> {
                    logout()
                }
            }
        }
        sheet.show(supportFragmentManager, BOTTOM_SHEET_TAG)
    }

    private fun logout() {
        viewModel.logout()
        startActivity(Intent(this, LoginActivity::class.java))
        finish()
    }

    private fun resetFragmentManager() {
        val transaction = supportFragmentManager.beginTransaction()

        repeat(viewModel.data.count()) { i ->
            val fragment =
supportFragmentManager.findFragmentByTag(FRAGMENT_TAG_PREFIX + i)
            if (fragment != null) {
                transaction.remove(fragment)
            }
        }
    }

```

```

        }
    }
    transaction.commitNow()

    viewModel.selectStep(0)
}

private fun showPage(index: Int) {
    val transaction = supportFragmentManager.beginTransaction()
    transaction.setCustomAnimations(R.animator.fade_in,
R.animator.fade_out)

    val currentFragment =
supportFragmentManager.primaryNavigationFragment
    var fragment =
supportFragmentManager.findFragmentByTag(FRAGMENT_TAG_PREFIX + index)
    if (currentFragment != null && (fragment == null ||
currentFragment != fragment)) {
        transaction.detach(currentFragment)
    }
    if (fragment == null) {
        fragment =
ContentFragment.newInstance(viewModel.selectedStep.value ?: 0)
        transaction.add(FRAGMENT_CONTAINER, fragment,
FRAGMENT_TAG_PREFIX + index)
    } else {
        transaction.attach(fragment)
    }

    transaction.setPrimaryNavigationFragment(fragment)
    transaction.commitNow()
}
}

```

Листинг Г.10 – FormViewModel.kt

```

package com.example.medicalapp.ui.form

import android.app.Application
import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.SavedStateHandle
import com.example.medicalapp.App
import com.example.medicalapp.Step
import com.example.medicalapp.TextInput
import com.example.medicalapp.data.LoginRepository

import com.example.medicalapp.data.MainRepository
import com.example.medicalapp.data.Prefs
import com.example.medicalapp.data.Repository
import com.example.medicalapp.data.model.ResultResponse
import kotlinx.coroutines.*
import kotlinx.serialization.decodeFromString
import kotlinx.serialization.json.Json
import com.example.medicalapp.data.Result.Success
import com.example.medicalapp.data.Result.Error

```

```

class FormViewModel(
    application: Application,
    private val loginRepository: LoginRepository,
    private val mainRepository: MainRepository,
    private val savedStateHandle: SavedStateHandle
) : AndroidViewModel(application) {

    companion object {
        private const val SELECTED_STEP = "selected_step"
    }

    private val job = Job()
    private val viewModelScope = CoroutineScope(Dispatchers.Main +
job)

    private val prefs: Prefs = (application as App).prefs

    //     private val _isDataValid = MutableLiveData<Boolean>()
    //     val isDataValid: LiveData<Boolean> = _isDataValid

    val selectedStep: LiveData<Int> =
savedStateHandle.getLiveData(SELECTED_STEP, 0)

    var inputs = HashMap<String, Any>()

    private val _photos: MutableLiveData<HashMap<String,
List<Pair<String, String?>>>> = MutableLiveData()
    val photos: LiveData<HashMap<String, List<Pair<String, String?>>>>
= _photos

    fun clear() {
        _photos.value = HashMap()
        inputs = HashMap()

        //         inputs["52tOTfbNw4neUw99BNah"] = prefs.firstName
        //         inputs["KzTi18pqbvTaBmUTX1TT"] = prefs.lastName
        //         inputs["GHBwfYeLlPAHIHkCF3x3"] = prefs.patronymic

        (data[0].attributes[0] as TextInput).input = prefs.lastName
        (data[0].attributes[1] as TextInput).input = prefs.firstName
        (data[0].attributes[2] as TextInput).input = prefs.patronymic

        selectStep(-1)
    }

    fun addPhoto(id: String, uri: String) {
        val copy = HashMap<String, List<Pair<String, String?>>>>()
        val initial = _photos.value
        if (initial != null) {
            copy.putAll(initial)
        }
        val photoList = copy[id]?.toMutableList() ?: mutableListOf()
        photoList.add(Pair(uri, null))
        copy[id] = photoList

        sendPhoto(id, uri)

        _photos.value = copy
    }
}

```

```

fun removePhoto(id: String, photo: Pair<String, String?>) {
    val copy = HashMap<String, List<Pair<String, String?>>>()
    val initial = _photos.value
    if (initial != null) {
        copy.putAll(initial)
    }
    val photoList = copy[id]?.toMutableList() ?: mutableListOf()
    val photoToRemove = photoList.find { it.first == photo.first }
    if (photoToRemove != null) {
        photoList.remove(photoToRemove)
    }
    copy[id] = photoList
    _photos.value = copy

    if (inputs.containsKey(id)) {
        val remotePaths = (inputs[id] as List<String>)
        val newRemotePaths = mutableListOf<String>()
        remotePaths.forEach {
            if (it != photo.second) {
                newRemotePaths.add(it)
            }
        }
        if (newRemotePaths.isEmpty()) {
            inputs.remove(id)
        } else {
            inputs[id] = newRemotePaths
        }
    }
    Log.d("test", inputs.toString())
}

fun setPhotos(new: HashMap<String, List<Pair<String, String?>>>) {
    _photos.value = new
}

private val _result = MutableLiveData<ResultResponse>()
val result: LiveData<ResultResponse> = _result

private val _errorMessage = MutableLiveData<String>()
val errorMessage: LiveData<String> = _errorMessage

val data: List<Step> =
application.assets.open("inputs.json").bufferedReader().use {
    Json.decodeFromString(it.readText())
}

init {
    val attrs = data[0].attributes
    (attrs[0] as TextInput).input = prefs.lastName
    (attrs[1] as TextInput).input = prefs.firstName
    (attrs[2] as TextInput).input = prefs.patronymic
}

fun onInputChanged(id: String, value: Any?) {
    if (value != null) {
        inputs[id] = value
    } else {
        inputs.remove(id)
    }
}

```

```

    }
    Log.d("test", inputs.toString())
}

private fun sendPhoto(id: String, localPath: String) =
viewModelScope.launch {
    when (val response = Repository.sendPhoto(localPath)) {
        is Success -> setRemotePath(id, localPath, response.data)
        is Error -> _errorMessage.value =
response.exception.message
    }
}

private fun setRemotePath(id: String, localPath: String,
remotePath: String) {
    val photos = _photos.value?.get(id)
    if (photos != null) {
        val photo = photos.find { it.first == localPath }
        if (photo != null) {
            val newPhotos = photos.toMutableList()
            newPhotos.remove(photo)
            newPhotos.add(Pair(localPath, remotePath))

            val newValue = mutableListOf<String>()
            newPhotos.forEach {
                if (it.second != null) {
                    newValue.add(it.second!!)
                }
            }

            val copy = HashMap<String, List<Pair<String,
String?>>>>()
            val initial = _photos.value
            if (initial != null) {
                copy.putAll(initial)
            }
            copy[id] = newPhotos
            _photos.value = copy
            onInputChanged(id, newValue)
        }
    }
}

fun canComplete(): Boolean {
    photos.value?.forEach {
        it.value.forEach { l ->
            if (l.second == null) {
                _errorMessage.value = "Фото загружаются"
                return false
            }
        }
    }
    data.forEach { step ->
        step.attributes.forEach { attribute ->
            if (attribute.isRequired() &&
!inputs.containsKey(attribute.getIdd())) {
                _errorMessage.value = "Заполните все необходимые
данные"
                return false
            }
        }
    }
}

```

```

        }
    }
}

return true
}

fun saveDraft() {
    prefs.firstName = inputs["52tOTfbNw4neUw99BNah"].toString()
    prefs.lastName = inputs["KzTi18pqbvTaBmUTX1TT"].toString()
    prefs.patronymic = inputs["GHBwfYeLlPAHIHkCF3x3"].toString()
}

fun complete() {
    //      var canComplete = true
    //      data.forEach { step ->
    //          step.attributes.forEach { attribute ->
    //              if (attribute.required) {
    //                  canComplete = canComplete &&
attribute.isFilled()
    //              }
    //          }
    //      }
    //      if (!canComplete) {
    //          _errorMessage.value = "You must fill required fields"
    //          return
    //      }
    //      val handler = CoroutineExceptionHandler { _, exception ->
    //          Log.d("Network", "Caught $exception")
    //      }

    viewModelScope.launch {
        when (val response = Repository.post(inputs)) {
            is Success -> {
                _result.value = response.data
            }
            is Error -> {
                _errorMessage.value = response.exception.message
            }
        }
    }
}

fun logout() {
    loginRepository.logout()
    prefs.firstName = ""
    prefs.lastName = ""
    prefs.patronymic = ""
}

fun selectStep(stepIndex: Int) {
    savedStateHandle.set(SELECTED_STEP, stepIndex)
}

fun nextStep(): Boolean {
    val selected = selectedStep.value ?: 0
    if (selected == data.size - 1) return false

    savedStateHandle.set(SELECTED_STEP, selected + 1)
}

```

```
        return true
    }

    override fun onCleared() {
        job.cancel()
    }
}
```