

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление 09.03.01 «Информатика и вычислительная техника»

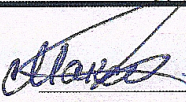
подготовки:

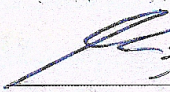
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ
ПОИСКА МЕДИЦИНСКИХ УЧРЕЖДЕНИЙ**

Студент  31.05.22. Максимова Д.Р.
(подпись, дата)

Руководитель  31.05.22 Митрохин М.А.
(подпись, дата)

Нормоконтролёр  31.05.22 Бычков А.С.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 № 13)

Заведующий кафедрой



(подпись)

М.А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК от 15.06.2022. № 11)

Секретарь ГЭК



(подпись)

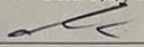
Е.И. Гурин

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«26» апреля 2022 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Максимова Дарья Ренатовна, гр.18ВВ1 факультета вычислительной техники.
2. Тема работы: «Мобильное приложение для поиска медицинских учреждений».

Тема утверждена приказом ПГУ №0357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы д.т.н., профессор, зав. каф. ВТ ПГУ Митрохин М.А.

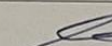
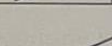
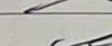
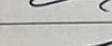
4. Задание на работу

Разработать мобильное приложение, предназначенное для поиска медицинских учреждений, первичной диагностики заболеваний, доступа к указаниям по оказанию первой медицинской помощи.

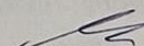
5. Перечень подлежащих разработке вопросов

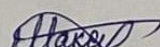
Требуется провести анализ предметной области, разработать архитектуру приложения, выбрать инструменты для разработки, реализовать клиент-серверное приложение.

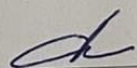
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10%	26.04.2022 – 28.04.2022	
2.	Выбор функционального состава устройства	10%	29.04.2022 – 5.05.2022	
3.	Разработка UML диаграмм	10%	6.05.2022 – 9.05.2022	
4.	Разработка интерфейса приложения	15%	10.05.2022 – 18.05.2022	
5.	Разработка программной части	30%	19.05.2022 – 23.05.2022	
6.	Отладка и тестирование программной части	10%	24.05.2022 – 25.05.2022	
7.	Выполнение задания по БЖД	5%	26.05.2022 – 27.05.2022	
8.	Оформление выпускной работы	10%	28.05.2022 – 31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы  26.04.22 Митрохин М. А.
(подпись, дата)

Задание к исполнению принял студент  26.04.22 Максимова Д. Р.
(подпись, дата)

Работу к защите допустить
Декан факультета  14.06.2022 Фионова Л. Р.
(подпись, дата)

Реферат

Работа содержит 87 листов, 23 рисунка, 2 таблицы, 5 источников, 3 приложения.

ANDROID, ПРИЛОЖЕНИЕ, КАРТЫ, МАРШРУТ, ЗАБОЛЕВАНИЕ, *JAVA*, *PYTHON*, *PYTORCH*, *FLASK*, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, НЕЙРОННАЯ СЕТЬ, *MAPKIT*, ЯНДЕКС, *RETROFIT*.

Цель ВКР – разработка приложения для поиска медицинских учреждений с возможностью первичной диагностики заболевания. Поиск медицинских учреждений осуществляется с помощью *SDK MapKit*. Первичная диагностика осуществляется с помощью анализа изображения, которое клиент отправляет на сервер, где нейронная сеть анализирует полученную картинку и отправляет назад ответ с информацией о предполагаемом заболевании.

В ходе выполнения ВКР было разработано клиент-серверное приложение, которое удовлетворяет поставленной задаче.

Графическая часть выпускной квалификационной работы содержит *UML*-диаграммы.

				ПГУ 1.090301.23.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Максимова Д.Р.					
Пров.		Митрохин М.А.					
Н. Контр.		Бычков А.С.					
Утв.		Митрохин М.А.					
Мобильное приложение для поиска медицинских учреждений					Лист	Лист	Листов
						2	87
Пояснительная записка					ФВТ, каф. ВТ гр. 18ВВ1		

Содержание

Введение.....	5
1 Мобильное приложение для поиска медицинских учреждений	6
1.1 Анализ предметной области для решения поставленной задачи.....	6
1.1.1 Анализ темы	6
1.1.2 Анализ аналогов	6
1.1.3 Постановка задачи	10
1.2 Архитектура приложения.....	11
1.2.1 Выбор архитектуры приложения	11
1.2.2 Общая структура проекта.....	12
1.2.1 Разработка <i>UML</i> -диаграмм.....	12
1.3 Выбор инструментов разработки	14
1.3.1 Выбор языков программирования.....	14
1.3.2 Выбор сервиса для отображения карт	15
1.3.3 Выбор фреймворка машинного обучения	16
1.3.4 Выбор среды программирования	17
1.4 Разработка программы.....	18
1.4.1 Работа с картами	18
1.4.2 Определение заболевания по изображению.....	25
1.4.3 Определение заболевания с помощью опроса	35
1.4.4 Рекомендации по оказанию медицинской помощи.....	37
1.4.5 Персонализация приложения.....	39
1.4.6 Разработка сервера.....	41
2 Безопасность жизнедеятельности.....	43

2.1 Характеристика опасных факторов при работе с ПЭВМ	43
2.2 Классификация и категорирование помещения	45
2.3 Санитарно-гигиеническая характеристика.....	46
2.4 Электробезопасность	47
2.5 Пожарная безопасность	49
2.6 Расчет количества светильников в помещении с ПЭВМ.....	50
Заключение	52
Список использованных источников	53
Приложение А – <i>UML</i> -диаграммы	54
Приложение Б – Листинг программы	63
Приложение В – Презентация.....	82

Введение

Доступ к информации всегда был одной из самых главных потребностей человека. В тяжелых ситуациях трудно сориентироваться и нет времени для поиска и проверки информации. Поэтому в рамках выпускной квалификационной работы было разработано приложение, решающее эту проблему. Необходимо разработать приложение, с помощью которого можно будет осуществлять легкий поиск по медицинским учреждениям. При необходимости получить первичную диагностику заболеваний с помощью современных технологий. Дать пользователю необходимую информацию по оказанию первой помощи.

					<i>ПГУ 1.090301.23.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум	Подпись	Дата		

1 Мобильное приложение для поиска медицинских учреждений

1.1 Анализ предметной области для решения поставленной задачи

1.1.1 Анализ темы

В современном мире люди обладают свободным доступом к информации. Иногда, от владения информацией может зависеть жизнь и благополучие человека. В регионах с труднодоступной инфраструктурой проблематично быстро получить медицинскую помощь. В таких ситуациях удобно использовать мобильное приложение. С помощью мобильного приложения можно получить не только первичную диагностику заболевания, но и найти актуальную информацию о действующих больницах и поликлиниках, указания по оказанию первой помощи. В ситуации, когда нет быстрого доступа к медицинской помощи, наиболее быстрым решением проблемы будет использование мобильного приложения, содержащего в себе всю необходимую информацию.

Разрабатываемое приложение будет важно для людей, кто не знаком с окружающей местностью, у кого нет возможности быстро получить первичную диагностику заболевания, если необходимо осуществить первую медицинскую помощь самостоятельно для себя или окружающих

1.1.2 Анализ аналогов

Перед разработкой приложения был проведен сравнительный анализ аналогичных приложений в *Google Play*, которые могут помочь пользователям в экстренных ситуациях, связанных с оказанием первой медицинской помощи и поиска медицинских учреждений. Был проведен анализ следующих приложений: «*Hospital Finder*», «*Nearest Hospital*», «*Australian Hospital Finder*».

Приложение «*Hospital Finder*» предоставляет пользователю информацию о медицинских учреждениях, находящихся поблизости. Имеется возможность осуществлять поиск по штату и городу, в котором находится больница. Есть функция поиска по названиям больниц. Каждое медицинское учреждение имеет подробное описание. Благодаря этому приложению пользователь может получить информацию о часах работы медицинского учреждения, контактные данные. Добавив медицинские учреждения в избранное, с помощью функции “*Favourite*”, пользователь может в любой момент открыть вкладку с информацией о медицинском учреждении, сохранённого в избранное. Функции, доступные пользователю представлены на рисунке 1.1.

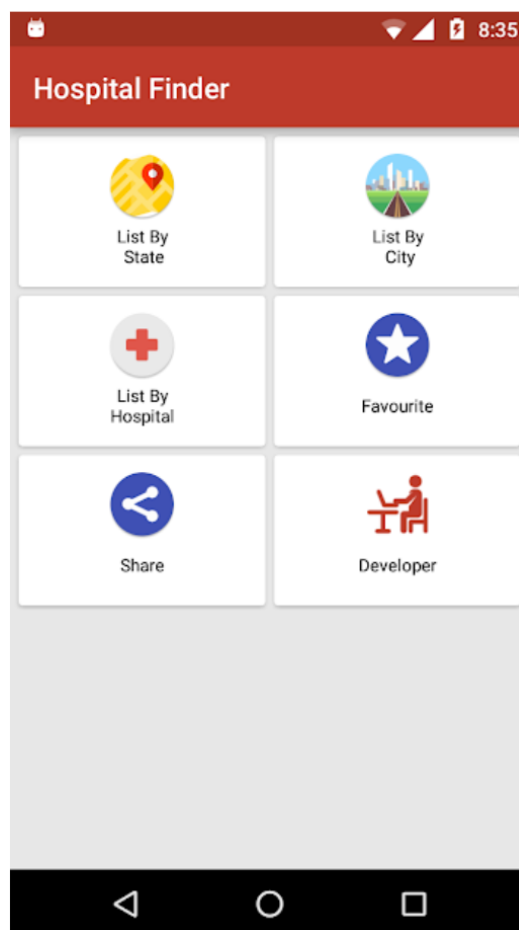


Рисунок 1.1 – Функционал приложения «*Hospital Finder*»

Приложение «*Nearest Hospital*» обладает только одной функцией - составление пути от текущей геопозиции до медицинского учреждения. На карте отображается текущее местоположение пользователя, есть визуальный и звуковой навигатор. Пользователь может осуществлять поиск как с помощью ввода строки с клавиатуры, так и с помощью голосового поиска. Интерфейс приложения представлен на рисунке 1.2.

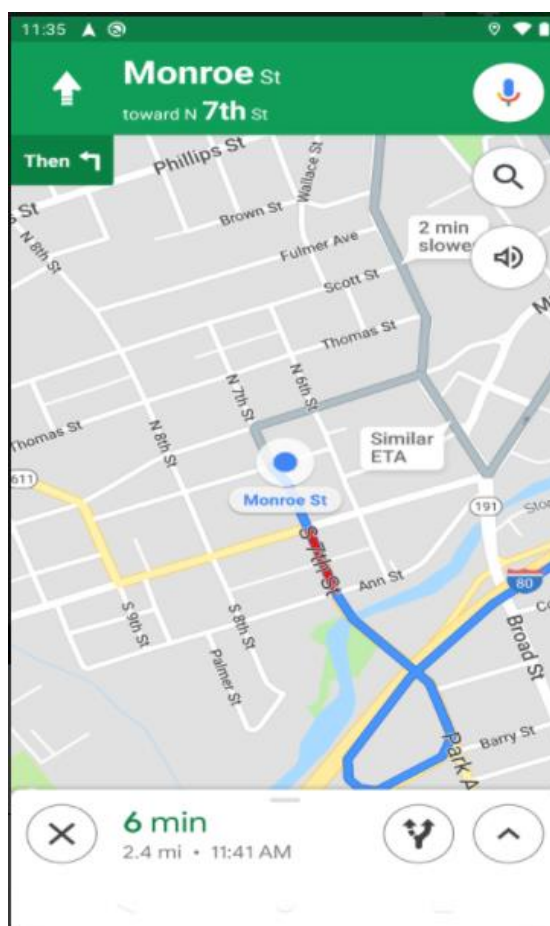


Рисунок 1.2 – Интерфейс приложения «*Nearest Hospital*»

С помощью приложения «*Australian Hospital Finder*» пользователь имеет доступ к поиску медицинских учреждений не только, когда у него есть интернет, но даже и без доступа к нему. Это является большим плюсом при использовании приложения в районах со слаборазвитой инфраструктурой. Пользователю доступна краткая информация о больницах, находящихся поблизости. Интерфейс приложения представлен на рисунке 1.3.

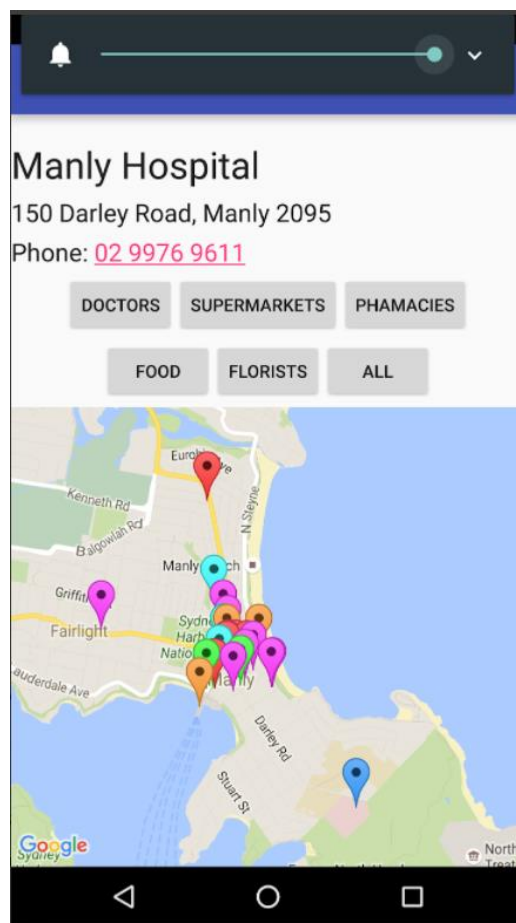


Рисунок 1.3 – Интерфейс приложения «*Australian Hospital Finder*»

В ходе сравнительного анализа было установлено, что приложения имеют очень небольшой набор функций. Интерфейс является непонятным интуитивно [1], что может значительно осложнить работу с приложением в стрессовых для пользователя ситуациях. Ни в одном из приложений нет первичной диагностики заболеваний, отсутствуют советы по оказанию первой медицинской помощи. Исходя из этого анализа становится ясно, что в магазине приложений *Google Play* не существует полноценного приложения по поиску медицинских учреждений, а существующие имеют ограниченный функционал. Именно поэтому появилась необходимость разработать приложение, имеющее набор всех необходимых функций.

Сравнение функционала вышеописанных приложений представлен в таблице 1.1.

Таблица 1.1 – Результат сравнительного анализа приложений.

Функция	Название приложения		
	<i>Hospital Finder</i>	<i>Nearest Hospital</i>	<i>Australian Hospital Finder</i>
Составление кратчайшего пути до медицинского учреждения	+	+	+
Первичная диагностика заболевания	-	-	-
Советы по оказанию первой медицинской помощи	-	-	-
Диагностика заболевания по фото	-	-	-
Интуитивный интерфейс	+	-	-

1.1.3 Постановка задачи

ОС *Android* является самой популярной операционной системой для мобильных устройств с долей рынка в 72%. Поэтому принято решение разработать *Android* приложение со следующим функционалом:

- составление кратчайшего пути до медицинского учреждения;
- первичная диагностика заболевания с помощью фото;
- первичная диагностика заболевания с помощью опроса пользователя;
- советы по оказанию первой медицинской помощи.

Для полноценного функционирования приложения необходимо будет использовать версию *Android* 5 и выше.

Для полноценного использования приложения, необходимо получить от пользователя доступ к определению текущего местонахождения, доступ в интернет, доступ к медиафайлам и камере.

1.2 Архитектура приложения

1.2.1 Выбор архитектуры приложения

Существует достаточно много подходов для построение сложных систем, все они задают способы разбиения приложения на модули. Хорошая архитектура должна обладать следующими свойствами:

- возможность тестирования системы;
- независимость архитектуры от любого окружения (база данных, интерфейс и другие службы).

Клиент-серверная архитектура удовлетворяет этим требованиям.

Клиент-серверная архитектура заключается в разделении системы на две категории программ: клиентская программа и сервер.

Клиентская программа подключается к серверу и выполняет определенную задачу. В частности, это может потребовать предоставления некоторых данных с сервера.

Сервер - это программа или процесс, осуществляющий прием и обработку запросов, отправку результата обработки данных, которые передал клиент.

Связь между сервером и клиентской программой определяется протоколами. Протокол - это язык общения между программами, в частности, между клиентской программой и сервером.

Для создания сервера будет использован фреймворк *Flask*.

Для взаимодействия между сервером и клиентом будет использоваться программная библиотека с открытым исходным кодом *Retrofit*.

1.2.2 Общая структура проекта

Для визуализации элементов и компонентов программы была составлена структурная схема проекта. Клиент отправляет запрос на сервер через *REST API*. Сервер отправляет запрос для нейронной сети на обработку изображения, в случае положительного ответа сервер передает данные клиенту через *REST API*. Схема представлена на рисунке 1.4.

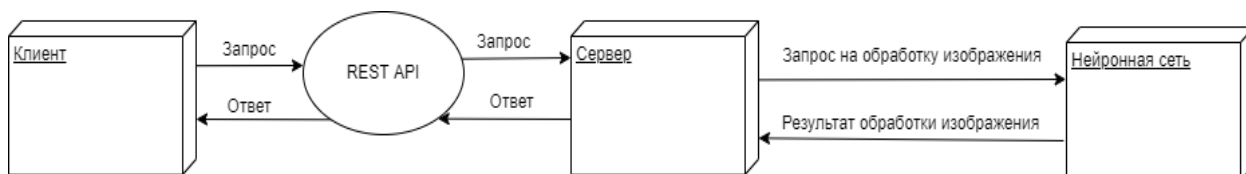


Рисунок 1.4 – Общая структура проекта

1.2.1 Разработка *UML*-диаграмм

Все описанные диаграммы представлены в приложении Б.

В мобильном приложении будет реализована клиент-серверная архитектура. Клиент будет взаимодействовать с сервером для того, чтобы производить анализ изображений с помощью нейронной сети. Взаимодействие осуществляется через интернет. *UML*-диаграмма развертывания наглядно показывает взаимодействие клиента и сервера через интернет.

Затем была составлена *UML*-диаграмма вариантов использования. С помощью этой диаграммы можно определить заранее функционал приложения, что несомненно облегчает разработку будущего приложения.

Определение заболевания по изображению осуществляется с помощью нейронной сети. Работа нейронной сети происходит на сервере. Клиент отправляет изображение на сервер, после чего сервер обрабатывает изображение. Сервер отправляет клиенту обратно название заболевания, вероятность этого заболевания, координаты пораженной заболеванием области. Взаимодействие клиента и сервера наглядно продемонстрировано на *UML*-диаграмме последовательности.

Более подробная логика работы сервера представлена на *UML*-диаграмме деятельности сервера. После получения изображения сервер с помощью нейронной сети ищет заболевание. Если заболевание найдено, то сервер формирует вероятность заболевания, затем название, после чего определяет область, пораженную заболеванием. Если заболевание не было обнаружено, то пользователю отправляется пустой ответ.

В приложении наиболее важными функциями являются поиск медицинского учреждения на карте, а также определение заболевания по изображению. Была составлена *UML*-диаграмма деятельности клиента для этих функций. Если пользователь выбрал работу с картами, то он воспользоваться несколькими функциями:

- изменение масштаба карты;
- составление маршрута;
- поиск по карте;
- отображение медицинских учреждений;
- определение текущей геопозиции.

Если была выбрана функция диагностики заболевания по фото, то пользователю необходимо загрузить изображение в приложение. После чего программа отправляет сообщение с изображением на сервер, после получения ответа результат отображается на экране. Если ответ не пришел, то пользователь получает сообщение об ошибке. В случае ошибки пользователю необходимо загрузить изображение в мобильное приложение заново.

UML-диаграмма классов показывает, как классы взаимодействуют между собой, а также описывает их внутреннюю структуру: поля, методы. Для осуществления функции определения заболевания было разработано три класса: *ImageFragment*, *Prediction*, *Square*. Класс *Square* содержит в себе координаты области, пораженной заболеванием, координаты передает в класс *Prediction*. Класс *Prediction* хранит в себе информацию об определенных

нейронной сетью заболеваний. Класс *ImageFragment* осуществляет вычисление заболевания пользователя по картинке.

Класс *RouteFragment* осуществляет работу с картой. Благодаря чему можно осуществлять поиск по карте, составлять маршруты, поиск по определенным категориям, поиск по конкретной строке, определение текущего местоположения пользователя и другие манипуляции с картой.

Первичная диагностика заболевания с помощью экспертной системы осуществляется с помощью классов: *SymptomsFragment*, *Question*, *Disease*. Класс *Disease* содержит в себе подробную информацию для каждого заболевания:

- название заболевания;
- описание;
- рекомендации по оказанию первой помощи;
- картинки, наглядно иллюстрирующие заболевание;
- иконку, для отображения в списке.

Информация, описанная в классе *Disease*, передается в класс *Question*. Класс *Question* содержит в себе:

- список из болезней, информацию которых получает из класса *Disease*;
- вопросы, для прохождения опроса.

В классе *SymptomsFragment* происходит подсчет результатов опроса, пройденного пользователем.

1.3 Выбор инструментов разработки

1.3.1 Выбор языков программирования

Сейчас трудно представить себе разработку мобильных приложений без *Java* и *Kotlin*. У этих языков есть ряд преимуществ и недостатков, которые повлияли на выбор языка программирования для разработки:

- *java* обладает большим сообществом разработчиков, большим количеством готовых библиотек и инструментов, а также много готового кода;
- *java* быстро компилируется (на 12-15% быстрее чем *Kotlin*);
- сообщество *java* примерно в пять раз больше, чем *Kotlin*. Большое сообщество позволяет быстро найти информацию о том, как решить ту или иную проблему;
- *UI ОС Android* написан на *Java*, есть много *SDK*;
- для начинающих разработчиков лучше выбрать язык программирования *Java*, потому что *Kotlin* является оберткой над *Java*, т.е. для разработки на *Kotlin* необходимо знать *Java*.

Учитывая очевидные преимущества был выбран язык программирования *Java*.

Нейронную сеть было решено писать на языке *Python*. *Python* – популярный язык для работы с нейронными сетями и обладает большим количеством фреймворков. Для реализации функций приложения самым удобным и простым в использовании является фреймворк *PyTorch*.

Чтобы не переписывать готовый код для анализа фотографий на другой язык, было принято решение использовать язык *Python* для написания сервера.

1.3.2 Выбор сервиса для отображения карт

Для разработки приложения с картами необходимо использовать *SDK*. Существует несколько популярных *SDK*, обеспечивающие доступ к картам и работе с ними. Самые популярные библиотеки для карт:

- *Google Maps*, разработанный корпорацией *Google*;
- *MapKit* от компании Яндекс;
- *2GIS SDK* от компании 2ГИС.

Google Maps обладают большой базой данных и большим функционалом. Многие сервисы, необходимые для разработки приложений с картами, *Google Maps* платные. Для получения *API* ключа необходима регистрация, которая недоступна для разработчиков некоторых стран. Из-за

этого разработка программы с использованием *Google Maps* стала невозможной.

Компания Яндекс предоставляет доступ к работе со своими картами благодаря программной библиотеке *MapKit*. Бесплатная версия *SDK MapKit* обладает практически аналогичным функционалом, что и платная, но с небольшими ограничениями, которые никак не повлияют на разработку приложения. В день можно делать 25000 бесплатных запросов, что достаточно для разрабатываемого приложения, можно пользоваться составлением и отображением маршрутов, поиском по карте и многими другими функциями.

SDK от компании 2ГИС обладает очень широким функционалом, но в бесплатной версии разработчику дается очень небольшое количество запросов, нет возможности осуществлять полноценный геокодинг. 2ГИС *SDK* удобнее использовать для других целей: планирование, решение *B2B*-задач, *B2L*-маркетинг, поиск клиентов для бизнеса и т.д. 2ГИС предоставляет бесплатный доступ к картам в зависимости от потребностей разработчика. Мне дали бесплатный доступ на 3 месяца с ограничениями: лимит 1000 запросов, ограниченный функционал карт.

Самым оптимальным вариантом для использования в разрабатываемом приложении является *MapKit*, потому что Яндекс предоставляет доступ к большому количеству запросов и широкому функционалу, что позволит распространить приложение среди большого количества пользователей.

1.3.3 Выбор фреймворка машинного обучения

Tensor Flow - это фреймворк с открытым исходным кодом для программирования для разработки нейронных сетей. Фреймворк сложен в использовании.

Keras - это фреймворк нейронных сетей с открытым исходным кодом, написанный на *Python*. *Keras* обладает относительно низкой производительностью, из-за чего фреймворк желательно использовать только для небольших наборов данных. Эти особенности *Keras* могут стать

существенными недостатками при дальнейшей разработке и возможного расширения функционала программы.

PyTorch - это фреймворк машинного обучения с открытым исходным кодом для *Python*, основанная на *torch*. Фреймворк был разработан исследовательской группой *Facebook* по искусственному интеллекту. *PyTorch* обладает удобной отладкой программы, удобно использовать для высокопроизводительных моделей и больших наборов данных, требующих высокой скорости выполнения. На официальном сайте *PyTorch* есть огромное количество обучающих уроков для новичков, готовых моделей нейронных сетей.

Для разрабатываемого приложения самым подходящим фреймворком является *PyTorch*.

1.3.4 Выбор среды программирования

Выбор среды программирования был между *Eclipse* и *Android Studio*.

Android Studio – официальная среда программирования, разработанная компанией *Google*. *Android Studio* обладает удобным отладчиком с большим количеством инструментов, большим количеством плагинов, которые помогают ускорить разработку проекта в разы. При разработке очень удобно использовать эмулятор *Android* устройства для визуализации приложения.

Eclipse — это стандартный пакет, который хорошо подходит для разработки *Java* и плагинов, а также для добавления новых плагинов. *Eclipse* имеет собственное рабочее пространство и расширяемую систему подключаемых модулей для настройки среды. *Eclipse* имеет короткое время сборки. Предпочтительнее для разработки небольших приложений.

IntelliJ IDEA – среда программирования разработанная компанией *JetBrains*. *IDE* была специально разработана для программирования на языке *Java*. Позволяет писать не только приложения для *Android*, но и консольные.

Был произведен сравнительный анализ, результаты которого представлены в таблице 1.2.

Таблица 1.2 – Результат сравнительного анализа сред разработки

Среда разработки	Достоинства	Недостатки
<i>Android Studio</i>	Поддерживает <i>Android</i> напрямую без сторонних расширений; происходят регулярные обновления; существует удобный эмулятор мобильного устройства, который облегчает разработку; большое количество инструментов для разработки	Расходует много ресурсов; сбои в программе
<i>Eclipse</i>	Существует огромное количество плагинов, который позволяют упростить процесс разработки	Поддерживает <i>Android</i> через сторонние расширения
<i>IntelliJ IDEA</i>	Позволяет разрабатывать кроссплатформенные приложения на языке <i>Java</i> ; удобная отладка кода	Есть платные функции; мало инструментов для <i>Android</i> разработки

Из результатов анализа ясно, что лучше всего использовать среду разработки *Android Studio*. Именно поэтому в проекте было решено использовать ее.

1.4 Разработка программы

1.4.1 Работа с картами

Для работы с *SDK MapKit* [2] необходимо получить уникальный *API* ключ на сайте Яндекса. Ключ автоматически генерируется, для этого разработчику необходимо только зарегистрироваться на сервисе и зайти в личный кабинет. После чего можно использовать *MapKit* в соответствии с документацией. Интерфейс кабинета разработчика представлен на рисунке 1.5.

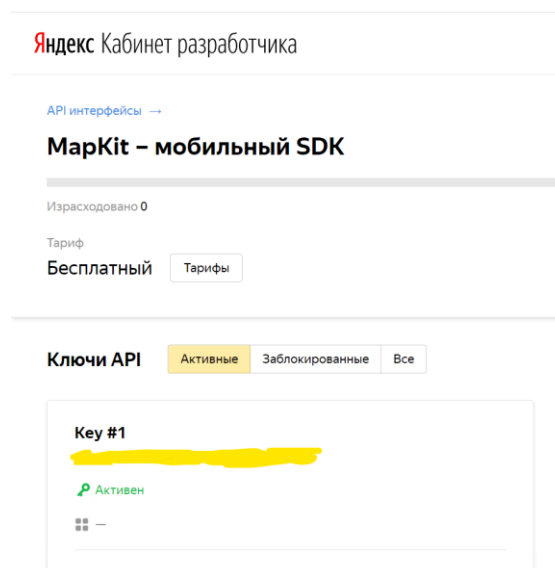


Рисунок 1.5 – Кабинет разработчика

При первом запуске приложения для работы с картами приложение запрашивает доступ к текущей геопозиции рисунок 1.6.

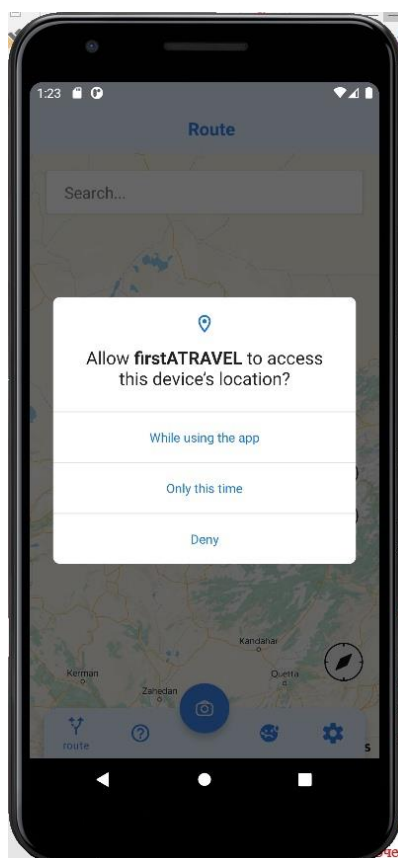


Рисунок 1.6 – Запрос на разрешение доступа к текущей геопозиции
ПОЛЬЗОВАТЕЛЯ

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

19

После получения доступа к геопозиции пользователь может видеть свое текущее местонахождение на карте. Оно отмечено синей точкой рисунок 1.7.

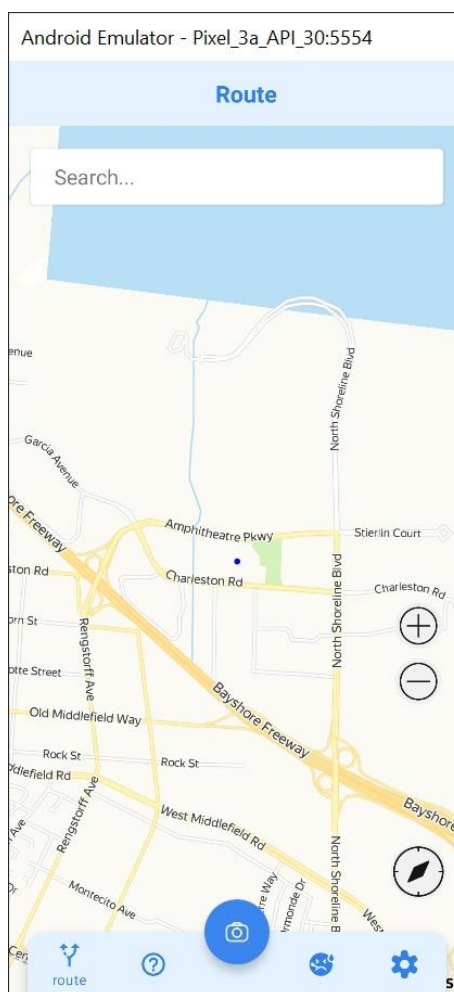


Рисунок 1.7 – Карты с отмеченной геопозицией

Для того, чтобы перейти к текущей геопозиции необходимо получить долготу и широту. Пользователю для этого ничего не надо делать, программа это сделает автоматически, т.к. ранее был получен доступ к текущей геопозиции. Приближение к текущему местоположению осуществляется с помощью метода *CameraPosition*. Фрагмент кода представлен ниже.

```
@Override
public void onLocationUpdated(@NonNull Location location) {
    Log.d("TagCheck", "LocationUpdated " +
        location.getPosition().getLongitude());
    Log.d("TagCheck", "LocationUpdated " +
        location.getPosition().getLatitude());
}
```

```

        curLoc = location;
        binding.mapview.getMap().move(
            new CameraPosition(location.getPosition(), 14.0f,
0.0f, 0.0f),
            new Animation(Animation.Type.SMOOTH, 1),
            null);

        binding.mapview.getMap().getMapObjects().addPlacemark(new
Point(location.getPosition().getLatitude(),
location.getPosition().getLongitude()));
    }

```

Пользователь может перемещаться по карте, изменять масштаб картами как с помощью жестов, так и с помощью кнопок увеличения и уменьшения масштаба. В любой момент, нажав кнопку компаса, пользователь может переместиться к текущему местонахождению на карте. Увеличение и уменьшение масштаба происходит аналогичным образом, как и перемещение к текущей геопозиции.

Одной из самых важных функций приложения является поиск медицинских учреждений. Все доступные медицинские учреждения отмечены на карте синим маркером рисунок 1.8.

Для того, чтобы отображались только медицинские учреждения, поиск по умолчанию осуществляется по ключевым словам: больница, поликлиника и т.д. Для этого прямо в коде были указаны эти ключевые слова. Фрагмент кода представлен ниже.

```

@Override
public void onSearchResponse(@NonNull Response response) {
    MapObjectCollection mapObjects =
binding.mapview.getMap().getMapObjects();
    mapObjects.clear();

    for (GeoObjectCollection.Item searchResult :
response.getCollection().getChildren()) {

```

```

        Point resultLocation =
searchResult.getObj().getGeometry().get(0).getPoint();
        if (resultLocation != null) {
            mapObjects.addPlacemark(
                resultLocation,
                ImageProvider.fromResource(requireContext(),
R.drawable.search_result));
        }
    }
}

```

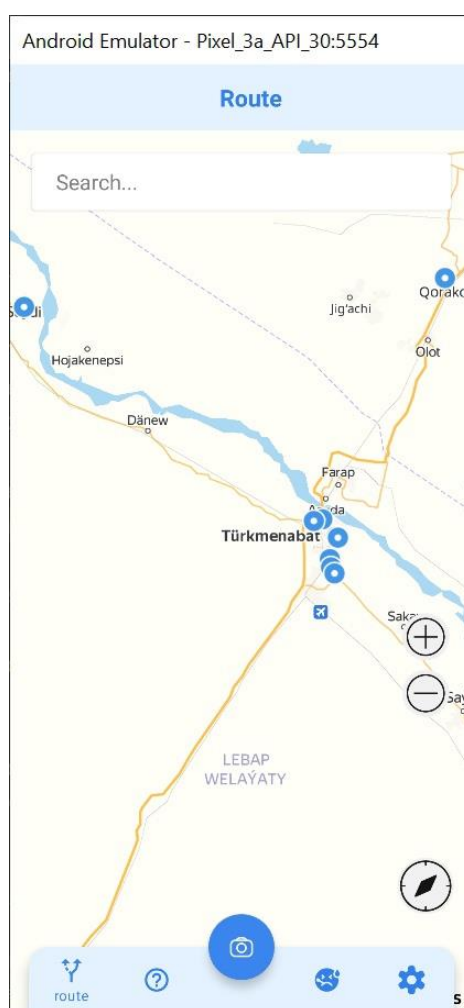


Рисунок 1.8 – Доступные медицинские учреждения на карте

Пользователь может осуществлять поиск медицинских учреждений не только с помощью карты, но и с помощью строки “*Search...*”. Для поиска с

помощью строки пользователю необходимо ввести название медицинского учреждения рисунок 1.9.

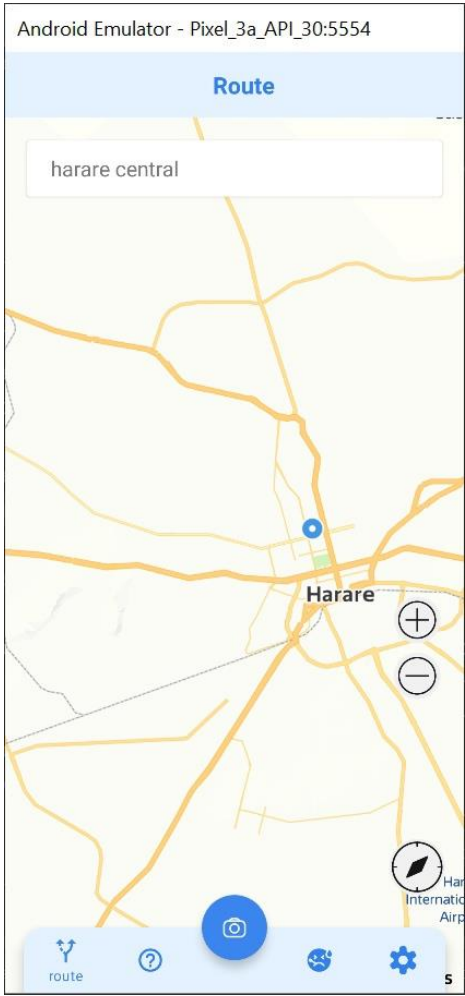


Рисунок 1.9 – Поиск медицинского учреждения с помощью окна “Search...”

Поиск по конкретным медицинским учреждениям осуществляется аналогичным образом, как и отображение всех медицинских учреждений, только по строчке, которую задал пользователь.

Если есть необходимость составления маршрута от текущего местоположения до госпиталя, то пользователю необходимо нажать на нужное медицинское учреждение на карте или ввести его название с помощью поискового окна. Пример маршрута на рисунке 1.10.

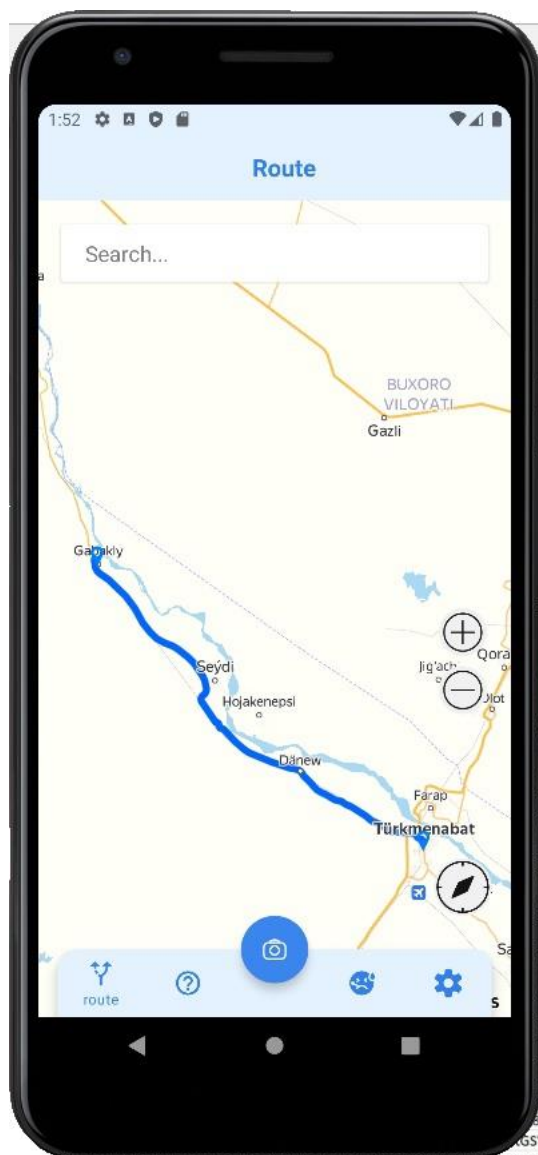


Рисунок 1.10 – Маршрут от текущего местоположения до медицинского учреждения

Для того, чтобы нарисовать маршрут необходимо получить текущее местонахождение пользователя, это будет начальной точкой и конечную точку маршрута. Далее с помощью полилиний маршрут отображается в приложении. *MapKit* предоставляет доступ к автомобильным маршрутам. Фрагмент кода представлен ниже.

```
private void submitRequest() {
```

```
    DrivingOptions drivingOptions = new DrivingOptions();
```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

24

```

VehicleOptions vehicleOptions = new VehicleOptions();
ArrayList<RequestPoint> requestPoints = new ArrayList<>();

requestPoints.add(new RequestPoint(
    ROUTE_START_LOCATION,
    RequestPointType.WAYPOINT,
    null));
requestPoints.add(new RequestPoint(
    ROUTE_END_LOCATION,
    RequestPointType.WAYPOINT,
    null));

drivingSession = drivingRouter.requestRoutes(requestPoints,
drivingOptions, vehicleOptions, this);
}

```

1.4.2 Определение заболевания по изображению

Нейронная сеть была написана с помощью фреймворка *PyTorch* [3]. Нейронная сеть выполняет поиск объектов на изображении. Данный процесс представляет собой классификацию и определение части изображения, в которой находится тот или иной класс [4]. Перед началом ее создания, нужно собрать данные для обучения и тренировки нейронной сети. Для этого были вручную найдены изображения для следующих классов: герпес, укус ядовитой змеи, укус ядовитого паука, меланомы, глаукомы. Для каждого класса было найдено минимум 70 изображений. После сбора для каждой картинки вручную была создана аннотация, включающая в себя место локализации класса, т.е. прямоугольник, в котором присутствует пораженная заболеванием область. Все аннотации были экспортированы в CSV файл, пример файла для одного из заболеваний представлен на рисунке 1.11. В первой колонке находится название болезни, во второй колонке расположены координаты пораженной болезнью области, третья колонка содержит в себе название изображения, которое использовалось для обучения нейронной сети, в

четвертой колонке находятся начальные значения длины и ширины изображения.

	A	B	C	D	E	F	G	H	I	J
1	venomous_spider_bite,338,973,173,169,-									
2	venomous_spider_bite,266,751,169,180,1V8UT4cRdEcDDqQmmK6QAHMIwBnOAPOMg17QtslLejgZDU-X6S									
3	venomous_spider_bite,420,857,131,113,2M_gsWqSjqcF7iDGT0JRhgW8dEanpzyr5WMTx8vDktqbth5VMOI									
4	venomous_spider_bite,353,841,83,94,37GWHJGzg83hVTXH8TONdogAHPExxNJUrLREtc20_J7okGv2Y4awBz									
5	venomous_spider_bite,300,766,270,286,3fs7Bc71dDp2ual9fdkYi2wRGxgvQbrAAONTgyGPzYzMA41u7NLUB7									
6	venomous_spider_bite,136,206,50,46,542598dc7e802_Untitled1.jpg,620,418									
7	venomous_spider_bite,327,796,147,143,6Ub6zwdntr_anLYgZjVQdUeYd6U-spdCIVhqM5OSjWOYEFwYrQyI									
8	venomous_spider_bite,353,879,135,150,6gkYvYuLL-tCsKNWlY9RSysLxEESVYgs4nxmzAigz_kYPfdOwqqafRu									
9	venomous_spider_bite,242,270,145,177,735245167c74ba5d6acad49a622079b2.jpeg,617,1024									
10	venomous_spider_bite,251,909,169,210,8nxyXX1G-w60THr09K1qop2NDmZnfnFaxhQHBZ0RrtQFfq5iMaby									
11	venomous_spider_bite,338,736,353,349,BP8b2ZWZVDAGs50L1g5Rxx9MkUOBKRHEyZOPdrTRaRAu3Cij1DN									
12	venomous_spider_bite,293,1003,150,147,CIJOTEQURWNQACGN8I7QzTsXByTFPGvXQJFDhqtbyLbKNBK3MA									
13	venomous_spider_bite,281,875,222,252,DWGbfyllps4m9WCy0aP_TeB41wmuMgHpjcW0OxWs5zT_eJ0uuI									
14	venomous_spider_bite,590,860,120,154,FQ_9Jdk7Utv82P7T2tj_kdCq8K3SrorP8I4NZi8jAXMpPWmBzAdKW									
15	venomous_spider_bite,428,924,105,105,HwMbleTp4kAutFZP5ducZsumhVo57eXiHilxY8sdu9glX-pHLe5jOV									
16	venomous_spider_bite,270,186,85,102,Hole-hand-necrosis-spider-bite-741363.jpg,590,776									
17	venomous_spider_bite,362,224,38,39,IMG-20140318-WA0000.jpg,800,600									
18	venomous_spider_bite,199,947,71,83,J0PSCfRsF9-ve9jGxk2v5E5tGSYLpu7sUpt2gC1aFmheY6z4VHm-ZNaj8									
19	venomous_spider_bite,240,804,312,248,LoBqgnE0VsnPomhivnCesTSrMDj_7TDxOQcsnMP-28LgzDFNH7G									
20	venomous_spider_bite,428,706,116,124,N5MukMANZra2BHI7n65b3W3Bd-QpZz3tA8OXfXHsZHHvfAxpGx1									
21	venomous_spider_bite,439,124,254,182,Not-a-spider-1100-x-480-1024x447.jpg,1024,447									
22	venomous_spider_bite,300,826,135,113,S6m9wt4UspXdt23p0aGOrAhsFZ05gM1SAR4sizGHRhYCZ2fDZ7Ng									
23	venomous_spider_bite,398,857,120,120,SxzMNNdNA2PnWtfaQI5XVI-cLUKWCRAllyzMVomoldio6LFICJS96m									

Рисунок 1.11 – CSV файл

Ниже представлен фрагмент кода, который описывает данные с различными болезнями. Для начала необходимо привести все изображения к единому формату и тензорам, которые необходимы для работы нейронной сети. Код представлен ниже.

```
class HerpesDataset(torch.utils.data.Dataset):

    def __init__(self, root, annotation, w, h, transforms=None):

        self.root = root

        self.transforms = transforms

        self.ann = annotation
```

```

        self.width = w

        self.height = h

    def __getitem__(self, index):

        item_ann = self.ann[index]

        label = 0#item_ann[0]

        start_x = int(item_ann[1])

        start_y = int(item_ann[2])

        end_x = int(item_ann[3]) + start_x

        end_y = int(item_ann[4]) + start_y

        print(end_x)

        path = item_ann[5]

        img = cv2.imread(os.path.join(self.root, path))

        img_rgb = cv2.cvtColor(img,
cv2.COLOR_BGR2RGB).astype(np.float32)

        img_height, img_width, img_channels = img.shape

        img_res = cv2.resize(img_rgb, (self.width, self.height),
cv2.INTER_AREA)

        img_res /= 255.0

        boxes = []

        boxes.append([start_x / (img_width / self.width),

                        start_y / (img_height / self.height),

                        end_x / (img_width / self.width),

                        end_y / (img_height / self.height)])

```

```

categories = []

categories.append(1)

boxes = torch.as_tensor(boxes, dtype=torch.float32)

area = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] -
boxes[:, 0])

categories = torch.as_tensor(categories,
dtype=torch.int64)

iscrowd = torch.zeros((boxes.shape[0],),
dtype=torch.int64)

```

Для использования аннотаций необходимо задать параметры файла с аннотациями. Код представлен ниже.

```

my_annotation = {}

my_annotation["boxes"] = boxes

my_annotation["labels"] = categories

my_annotation["iscrowd"] = iscrowd

my_annotation["area"] = area

my_annotation["size"] = iscrowd

image_id = torch.tensor([index])

my_annotation["image_id"] = image_id

if self.transforms:

    sample = self.transforms(image = img_res,

                             bboxes = my_annotation['boxes'],

                             labels = categories)

    img_res = sample['image']

```

```

        my_annotation['boxes'] =
torch.Tensor(sample['bboxes'])

        return img_res, my_annotation

def __len__(self):

    return len(self.ann)

#Далее необходимо определить каталог файлов и каталог
тестирования

train_dir = '/content/pics/herpes_image/'

test_dir = '/content/pics/test/photo/'

#После чего проводится проверка набора данных.

dataset = HerpesDataset(train_dir, data, 600, 400)

print('Length of dataset:', len(dataset), '\n')

img, target = dataset[0]

print('Label example:', target)

```

База данных необходима для того, чтобы в дальнейшем была возможность итерироваться по собранным картинками в процессе обучения нейронной сети.

Архитектура нейронной сети – *Faster R-CNN*. Данная модель используется для поиска объектов на изображении и распознавания образов.

Была взята готовая модель нейронной сети от компании *Microsoft*, которая используется для обнаружения объектов на изображении. Код модели нейронной сети представлен ниже.

```
def get_object_detection_model(num_classes):
```

```

#Загрузка предварительно обученной модели:
model =
torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=
True)

#Получение количества входных признаков для классификатора:

in_features =
model.roi_heads.box_predictor.cls_score.in_features

#Замена предварительно подготовленных заголовков на новые:

model.roi_heads.box_predictor = FastRCNNPredictor(in_features,
num_classes)

return model

```

Были определены гиперпараметры, которые используются в дальнейшем для обучения нейронной сети. *Batch size* отвечает за количество фотографий, которое будет подаваться в нейронную сеть.

```

data_loader = torch.utils.data.DataLoader(

    dataset,

    batch_size=2,

    shuffle=True,

    num_workers=2,

    collate_fn=utils.collate_fn,

)

data_loader_test = torch.utils.data.DataLoader(

    dataset_test,

    batch_size=2,

```

```

shuffle=False,

num_workers=2,

collate_fn=utils.collate_fn,

)

img, target = dataset_test[0]

plot_img_bbox(torch_to_pil(img), target)

```

Результат работы нейронной сети представлен на рисунке 1.12.

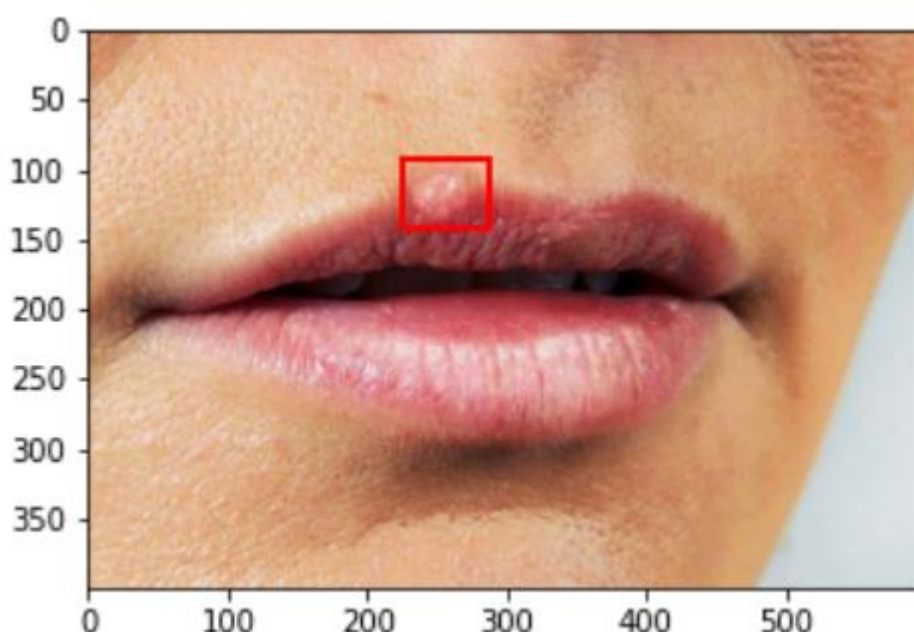


Рисунок 1.12 – Результат работы нейронной сети

Обучение проходит на сервисе *GOOGLE COLAB* с использованием технологии *CUDA*, которая распараллеливает вычисления, благодаря чему обучение происходит гораздо быстрее. Рисунок 1.13 включение *CUDA* технологий.

Данная технология была создана компанией *NVIDIA*, что послужило сильным скачком в развитии нейронных сетей и систем искусственного интеллекта.

Настройки блокнота

Аппаратный ускоритель

GPU 

Чтобы максимально эффективно использовать Colab, старайтесь не применять ускоритель (GPU), если он вам не нужен. [Подробнее...](#)

☐ Выполнение в фоновом режиме

Хотите, чтобы ваш код выполнялся, даже когда вы закроете браузер? [Перейти на Colab Pro+](#)

☐ Исключить выходные данные кодовой ячейки при сохранении блокнота

Отмена

Сохранить

Рисунок 1.13 – Настройка блокнота

В фреймворк *PyTorch* входит готовая функция, которая используется для тренировки одной эпохи нейронной сети – *train_one_epoch*. После 5 эпох был получен результат, представленный на рисунке 1.14.

```
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.225
Average Precision (AP) @[ IoU=0.50      | area= all | maxDets=100 ] = 0.654
Average Precision (AP) @[ IoU=0.75      | area= all | maxDets=100 ] = 0.169
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.266
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.013
Average Recall    (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.236
Average Recall    (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.336
Average Recall    (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.336
Average Recall    (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall    (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.383
Average Recall    (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.050
```

Рисунок 1.14 – Результат тренировки нейронной сети

Нейронная сеть используется для анализа изображения, которое пользователь загружает в приложение. Изображения, проанализированные

нейронной сетью, нигде не сохраняются. Для безопасности данных пользователя важно, что приложение не собирает и не хранит никакой личной информации, которую пользователь загружает в приложение.

При нажатии на иконку в круглой рамочке пользователю необходимо добавить изображение для анализа рисунок 1.15. Первичная диагностика осуществляется по кожным и глазным заболеваниям:

- герпес;
- укус ядовитой змеи;
- укус ядовитого паука;
- меланома;
- глаукома.

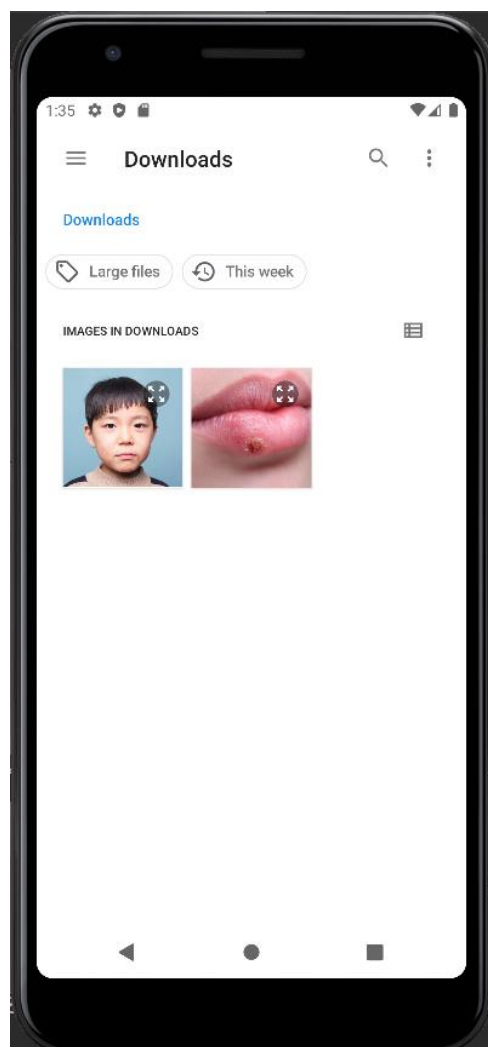


Рисунок 1.15 – Выбор изображения пользователем

После диагностики заболевания на странице “*Prediction by image*” пользователь видит вероятность заболевания, которое было найдено с помощью программы, название заболевания. После чего можно перейти по кнопке “*Get advice*” на страницу с подробным описанием болезни и рекомендациями по оказанию первой медицинской помощи. Страница с рекомендациями и описанием заболевания подробно описана в пункте 1.4.4 Рекомендации по оказанию первой медицинской помощи. Результат работы программы представлен на рисунке 1.16.

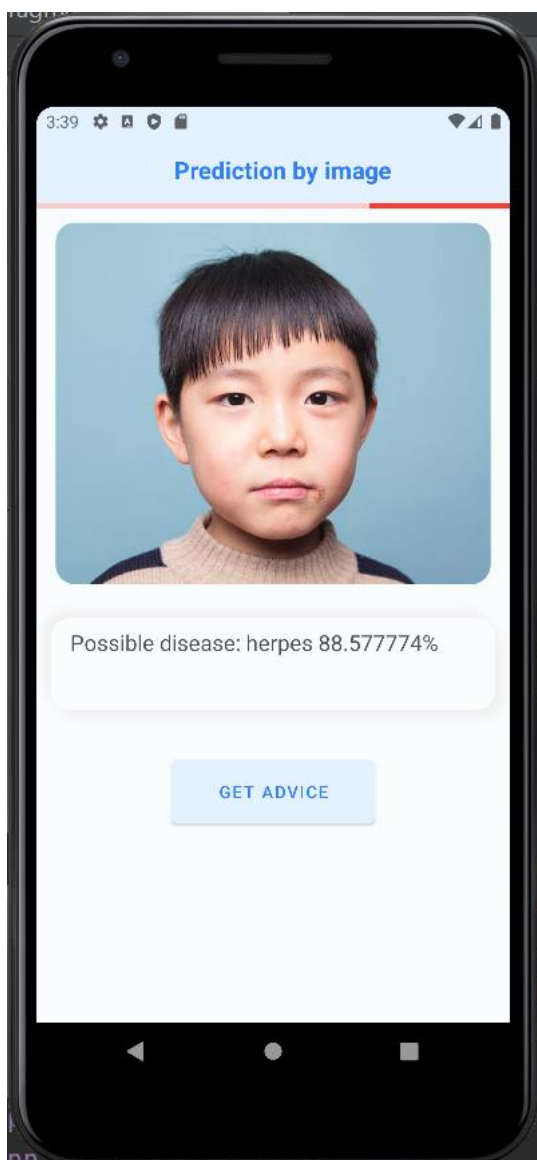


Рисунок 1.16 – Определение заболевания по изображению

1.4.3 Определение заболевания с помощью опроса

При нажатии на вкладку меню “*symptoms*” перед пользователем открывается страница с вопросами. Вопросы перед пользователем появляются поочередно. Для получения результата пользователь должен на каждый вопрос ответить “yes” или “no”. Интерфейс вкладки “*symptoms*” представлен на рисунке 1.17.

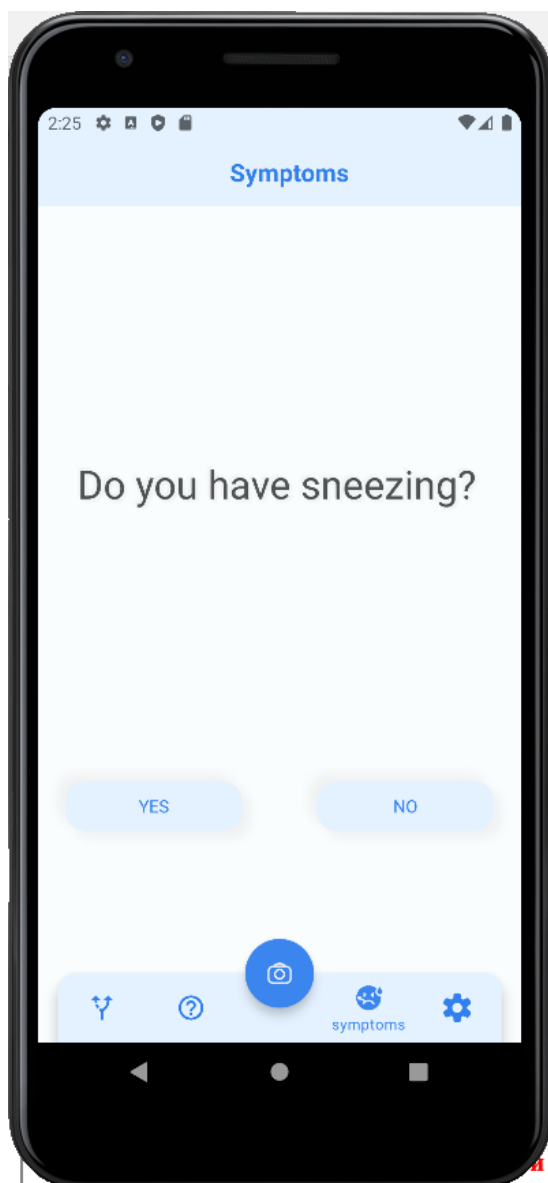


Рисунок 1.17 – Вкладка “*symptoms*”

После того, как пользователь ответит на все вопросы, он получит результат в виде названия предполагаемого заболевания и при нажатии на

кнопку “*Get advice*” получит подробное описание заболевания и информацию по оказанию первой медицинской помощи. Подробное описание рекомендаций в пункте 1.4.4 Рекомендации по оказанию медицинской помощи. Результат первичной диагностики представлен на рисунке 1.18.

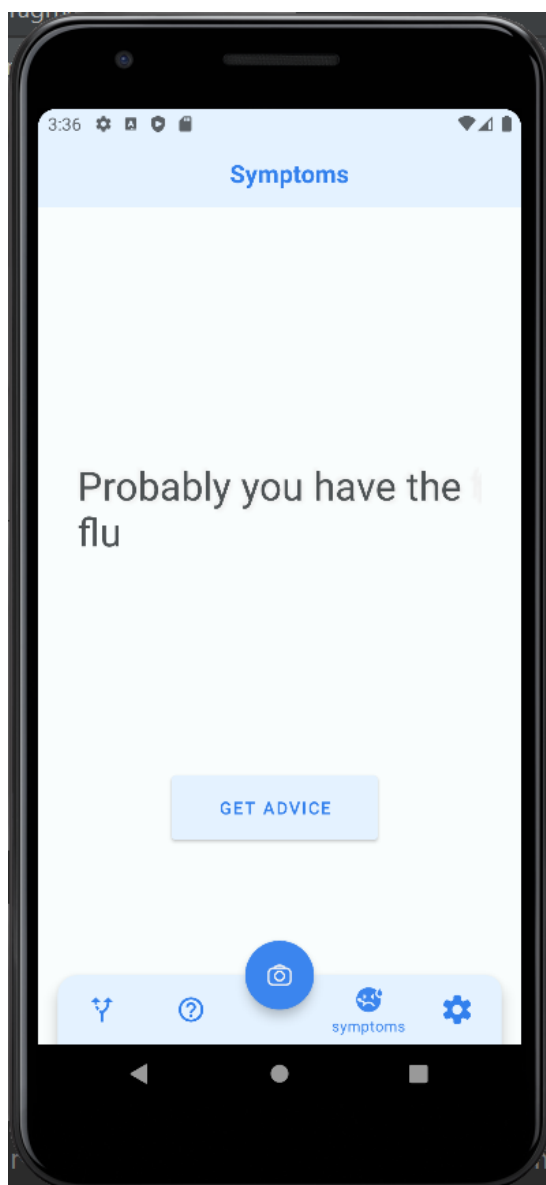


Рисунок 1.18 – Результат первичной диагностики с помощью опроса

Для проведения первичной диагностики с помощью опроса была составленная собственная экспертная система [5]. Для этого был разработан список вопросов по симптомам, которые присущи различным заболеваниям. Каждый положительный ответ равноценен одному баллу. После прохождения

опроса программа считает какая болезнь имеет наибольшее количество баллов. Болезнь с самой большой суммой баллов является наиболее вероятным заболеванием пользователя. Фрагмент кода для экспертной системы представлен ниже.

```
private void computeDisease() {

    Question.Disease disease = null;
    int max = -1;
    for (Question.Disease key : rating.keySet()) {
        if (rating.get(key) > max) {
            disease = key;
            max = rating.get(key);
        }
    }

    if (disease==null || disease.equals(""))
    {
        binding.question.setText("You are healthy!");
    }

    else

    {
        binding.question.setText("Probably you have " +
disease.name);
    }
}
```

1.4.4 Рекомендации по оказанию медицинской помощи

Для того, чтобы получить советы по оказанию первой медицинской помощи пользователю необходимо нажать на кнопку “*advice*”. После нажатия кнопки пользователь видит список с заболеваниями. Благодаря тому, что был использован список, пользователю не надо долго искать информацию о

заболевании. Список проиллюстрирован картинками, которые ассоциируются с конкретным заболеванием. Картинки помогают пользователю лучше и быстрее ориентироваться по списку, особенно, если пользователь находится в стрессовой ситуации. Интерфейс вкладки представлен на рисунке 1.19.

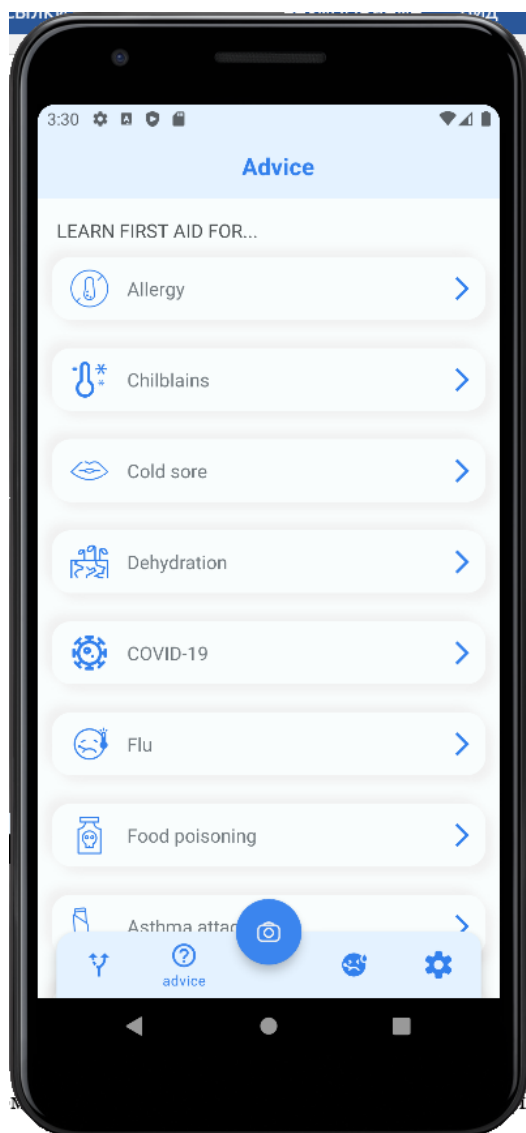


Рисунок 1.19 – Интерфейс вкладки “*advice*”

После нажатия на определенный пункт списка пользователь перейдет на новую страницу, где предоставлена подробная информация о заболевании и указания по оказанию первой медицинской помощи. Вначале страницы расположена фотография, на которой изображено состояние человека, страдающего заболеванием. После фотографии есть текст, кратко

описывающий симптоматику заболевания. Рекомендации по оказанию первой помощи разделены по пунктам. Каждый пункт пронумерован большой красной цифрой, чтобы в случае стрессовой ситуации пользователь смог быстро ориентироваться по тексту. Интерфейс совета представлен на рисунке 1.20.

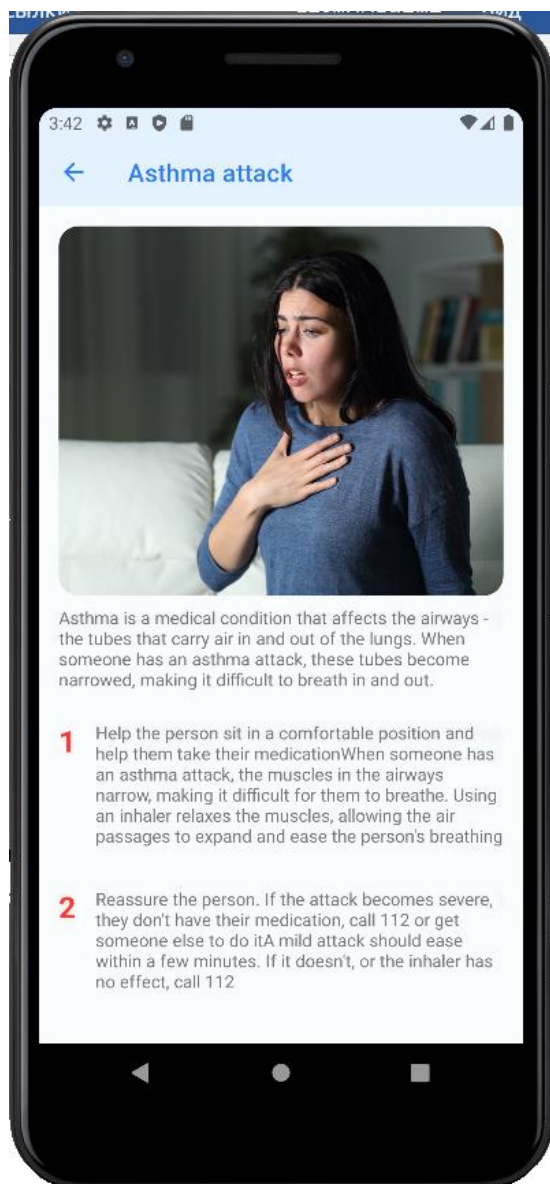


Рисунок 1.20 – Рекомендация по оказанию медицинской помощи

1.4.5 Персонализация приложения

Пользователь может менять настройки приложения. Для этого во вкладке “*settings*” пользователю необходимо перетащить ползунок вправо,

Изм.	Лист	№ докум	Подпись	Дата

если он хочет использовать какую-либо функцию интерфейс вкладки “settings” представлен на рисунке 1.21.

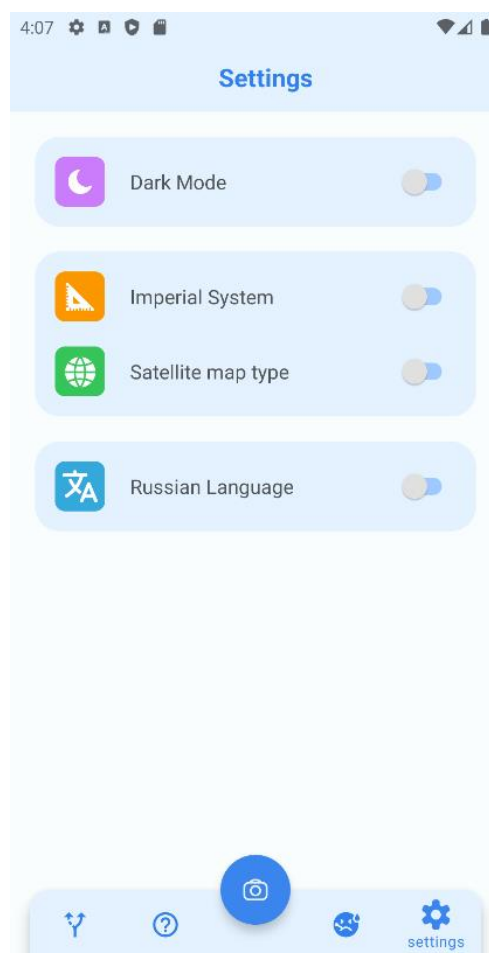


Рисунок 1.21 – Интерфейс вкладки “settings”

Можно поменять тему на темную или светлую. Для этого были созданы две цветовые темы в файле “*themes.xml*”, которые используются в зависимости от настроек пользователя.

Для того, чтобы изменить язык приложения создано два файла со строками, которые используются в приложении. Которые меняются между собой при изменении выбора пользователем. Пользователь может использовать приложение на русском или на английском языках.

Для изменения настроек карт были использованы стандартные функции *MapKit*.

Приложение после включения персональных настроек представлено на рисунке 1.22.

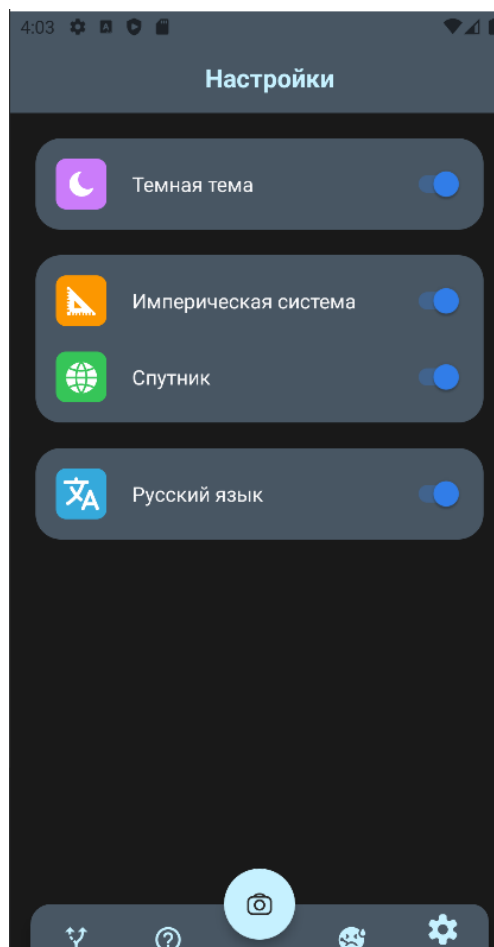


Рисунок 1.22 – Пример применения пользовательских настроек

1.4.6 Разработка сервера

Для использования нейронных сетей было необходимо разработать сервер. Разработка сервера велась с использованием фреймворка *Flask*. Данный фреймворк позволяет быстро и эффективно создавать серверные приложения. На сервер от клиента приходит сообщение об изображении в формате закодированной строки. Данные сохраняются в файл, с помощью нейронных сетей анализируются, после чего сервер отправляет ответ клиенту. Фрагмент кода сервера представлен ниже.

```
from detection import check_herpes
app = Flask(__name__)
```

Изм.	Лист	№ докум	Подпись	Дата

```

@app.route('/image', methods=['POST'])
def upload():
    print("Get photo")
    img = request.json
    data = base64.b64decode(img)
    with open('static/uploads/img.jpg', 'wb') as f:
        f.write(data)
        f.close()
    result = check_herpes(f.name)
    out = jsonify(result)
    print(result)
    return out

if __name__ == '__main__':
    # run app in debug mode on port 5000
    app.run(debug=True, port=5000)

```

Для взаимодействия сервера и клиента была использована библиотека *Retrofit*. Код программы представлен ниже.

```

package com.example.firstatravel.fragments;
import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.Headers;
import retrofit2.http.POST;

public interface DiseaseAPI {
    @Headers("Content-Type: application/json")
    @POST("image")
    Call<Prediction> listRepos(@Body String image);
}

```

2 Безопасность жизнедеятельности

2.1 Характеристика опасных факторов при работе с ПЭВМ

В процессе создания приложения, программы или любого другого проекта необходимо постараться учесть негативное влияние различных факторов, которые могут усложнить разработку. При работе за компьютером нужно создать комфортную обстановку и оградить разработчика от негативного влияния окружающих условий. Для начала нужно понять, какие факторы могут сказаться на работоспособности. Классификация опасных факторов изображена на рисунке 2.1.

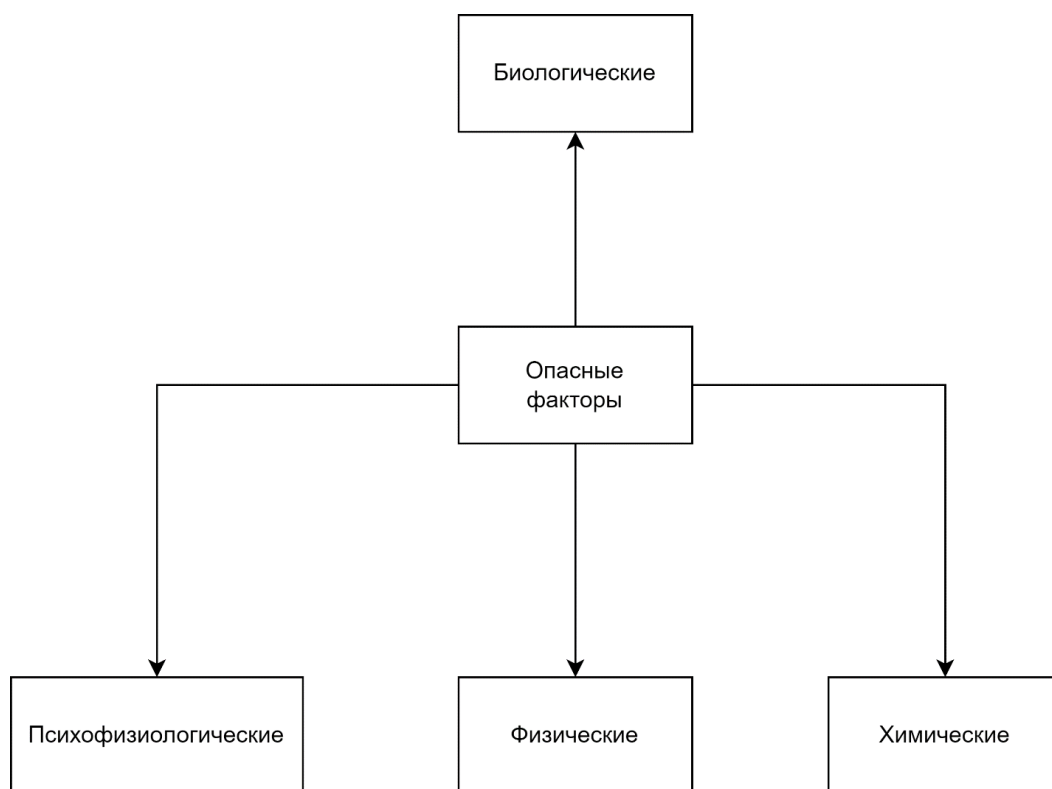


Рисунок 2.1 – Классификация опасных факторов

Рассмотрим подробнее классификацию. Первый тип опасных факторов – психофизиологический. Сюда относятся следующие опасные факторы:

- умственное и физическое перенапряжение;
- снижение работоспособности при долгой работе;

- усталость глаз;
- головные боли;
- боли в спине, шее.

При разработке программы человек может долгое время находиться за компьютером. Это может привести к умственному и физическому перенапряжению и усталости. Также могут возникать головные боли. К концу рабочего дня у человека может снижаться работоспособность, появляется сухость глаз, слезоточивость, возникают боли спины, шеи, могут затекать руки и ноги, рассеивается внимание. Для того, чтобы избежать этих негативных последствий необходимо делать перерывы в процессе работы для того, чтобы не перенапрягать глаза. Также хорошим решением будет чередование физической и умственной работы, поэтому периодически можно делать зарядку.

Второй тип – физический. К нему относятся следующие факторы:

- уровень рентгеновского излучения, отличный от нормы;
- уровень инфракрасного излучения, отличный от нормы;
- уровень статического электричества, отличный от нормы;
- уровень ультрафиолетового излучения, отличный от нормы;
- слишком низкая или высокая температура воздуха или области, в которой находится компьютер;
- большое количество пыли в воздухе рабочей области, на самом компьютере или на приборах, которые к нему подключены;
- неправильные настройки яркости и контрастности, выставленные для монитора;
- повышенное количество аэроионов в воздухе;
- повышенная блескость;
- чрезмерное количество солнечных лучей, попадающих на монитор;
- низкая освещенность рабочего места.

Химические факторы – содержание в воздухе токсичных веществ. Такие вещества могут содержаться в краске и лаках, которые могут использоваться для восстановления внешнего вида элементов интерьера в помещении. Также одним из опасных факторов является повышенное содержание углекислого газа в воздухе. Чтобы не допустить этого, нужно регулярно делать проветривание помещения, в котором происходит работа. Список всех факторов представлен ниже:

- повышенное содержание углекислого газа;
- нахождение на поверхностях элементов интерьера красок и лака;
- повышенное содержание в воздухе озона, формальдегида и других токсичных веществ;
- наличие в помещении некачественных видов пластика, которые при повышенной температуре могут выделять токсичные вещества в воздух.

Последний тип факторов – биологический. К этому типу относится повышенное содержание микроорганизмов в воздухе. В связи с этим, необходимо чаще проветривать помещение, в котором происходит рабочая деятельность.

2.2 Классификация и категорирование помещения

Рабочая деятельность программиста происходит в офисе. Для того, чтобы работа происходила в комфортных условиях и не было происшествий, необходимо позаботиться о пожарной безопасности. Для начала, нужно определить, к какой категории по пожарной безопасности относится офис, в котором программист осуществляет свою рабочую деятельность. Было выявлено, что помещение относится к категории В1-В4, т.к. в нем находятся различные твердые горючие вещества и материалы, куда можно отнести бумагу.

Для того, чтобы избежать различных происшествий в рабочей зоне, работодатель обязан обеспечивать комфортные и безопасные условия. Одной

из причин возникновения пожаров может стать повышенная температура воздуха в рабочей зоне. Температура по стандартам должна находиться в пределах от 22 до 24 градусов.

Для того, чтобы поддерживать ее в допустимой норме, в летнее время необходимо применять кондиционирование воздуха.

Кондиционер – устройство, которое обеспечивает регулирование температуры и поддержание необходимых климатических условий в помещении.

В зимнее время температуру необходимо поддерживать с помощью обогревателей и радиаторов для отопления помещения.

2.3 Санитарно-гигиеническая характеристика

Работу можно разделить на категории. Категория I подразумевает легкую физическую деятельность, в ходе которой энергозатраты не превышают 150 кКал/ч. К категории IA относятся работы, сопровождающиеся незначительным физическим напряжением мышц. Категория IB связана с небольшими нагрузками, ходьбой.

Работа программиста относится к категории IA и включает в себя следующие требования и нормы, определенные СанПиН:

- поддерживать температуру воздуха в пределах от 22 до 24 градусов;
- верхний предел относительной влажности воздуха не превышает отметку в 75%;
- скорость движения воздуха не больше 0,1 м/с.

В ходе разработки проекта человек в основном смотрит в монитор и подвергает свои глаза сильному напряжению. Уже в конце рабочего дня программист может страдать от повышенной слезоточивости глаз, помутнения зрения и сильной усталости. Также, когда человек смотрит в монитор, он может концентрироваться и забывать моргать. Все эти факторы в совокупности могут привести к синдрому сухого глаза – состоянию, когда глаз

не получают необходимого количество слезной жидкости для увлажнения. Синдром сухого глаза может привести к серьезным проблемам со зрением, травмам роговицы, а также к конъюнктивиту. Чтобы избежать проблем со зрением, можно регулярно делать перерывы в процессе работы, делать глазную гимнастику и смотреть на объекты, которые находятся далеко, т.к. глаз человека привыкает к фиксированному расстоянию, на котором находится монитор. Кроме зарядки, можно применять капли, которые помогут обеспечивать увлажнение глаза.

Одной из причин усталости и напряжения глаз является плохое освещение рабочего места. Правила, которые необходимо соблюдать при работе за компьютером:

- не допускать работу в темном помещении без использования искусственных источников света;
- не допускать чрезмерного попадания солнечных лучей на монитор, т.к. это может затруднить чтение текста на нем, а также может привести к выцветанию цветов и искаженному отображению картинки;
- сочетать искусственное освещение и естественное.

Также в рабочих помещениях не должны нарушаться пределы допустимого уровня шума и вибраций.

2.4 Электробезопасность

При разработке программ в помещении должны соблюдаться правила электробезопасности, которые помогут избежать ситуаций, в которые появляется опасность поражения электрическим током. В результате воздействия электрического тока возникают поражения кожи и внутренних органов, ожоги различной степени тяжести. В основном при прохождении тока через тело человека, поражение можно визуально заметить, однако это происходит не всегда. В таких случаях следуют срочно оказать помощь пострадавшему.

В ходе разработки проекта есть вероятность поражения людей электрическим током. Чтобы этого избежать, необходимо следовать различным правилам, которые помогут предотвратить и избежать опасных ситуаций на рабочем месте. Условия для обеспечения электробезопасности:

- содержать оборудование в исправном состоянии;
- относительная влажность воздуха не должна превышать отметку в 75%;
- рабочая поверхность не должна содержать большое количество пыли;
- поддерживать температурные характеристики в пределах установленных норм;
- проводить беседы с сотрудниками с целью их информирования о правилах электробезопасности.

Кроме вышеперечисленных условий, работодатель обязан выполнять ряд требований:

- обеспечение безопасности в ходе рабочей деятельности;
- обеспечение сотрудников всеми необходимыми инструментами и оборудованием для работы;
- соблюдать правила пожарной безопасности и распространять данную информацию среди сотрудников;
- соблюдать правила электробезопасности.

В основном поражения током могут происходить, когда человек взаимодействует с поврежденным оборудованием, проводами, при попадании жидкостей на открытые поверхности оборудования (плата, клавиатура).

Чтобы избежать травм и угроз для жизни, нужно соблюдать правила электробезопасности. К ним относятся следующие:

- не допускать попадания жидкостей на оборудование;
- не допускать пользование приборами с явными внешними повреждениями;

- не допускать взаимодействие с проводами и кабелями с открытой изоляцией или с проводами, которые имеют внешние повреждения;
- не нарушать температурные нормы и условия микроклимата в помещении;
- проводить с сотрудниками беседы с целью информирования и обсуждения мер, направленных на поддержание электробезопасности.

2.5 Пожарная безопасность

При работе за компьютером необходимо проявлять осторожность и следовать правилам, которые помогут избежать возникновения опасных ситуаций и угроз. Пожарная безопасность – комплекс мер, которые направлены на предотвращение угроз возникновения пожара.

Выделяют следующие правила, которые необходимо соблюдать:

- не пользоваться приборами и проводами, которые имеют поврежденный вид;
- не курить в помещениях, а использовать для этого специально отведенные места;
- не устанавливать приборы рядом с нагревателями и другими источниками тепла;
- обучать сотрудников правилам пожарной безопасности.

В случае возникновения пожара необходимо предпринять следующие действия:

- по возможности отключить оборудование от сети;
- по возможности эвакуировать материально-ценные вещи;
- сообщить о произошедшем соответствующим лицам (руководителю, пожарным).

Для тушения пожара необходимо иметь огнетушители, пожарные краны и пожарные щиты, которые необходимо использовать для предотвращения и ликвидации пожара. При использовании огнетушителя необходимо, чтобы его

тип соответствовал типу веществу или материалу, которое подверглось возгоранию.

2.6 Расчет количества светильников в помещении с ПЭВМ

Как было описано выше, в ходе длительной работы за компьютером, у человека может возникнуть усталость и слезоточивость глаз, а в дальнейшем все это может привести к синдрому сухого глаза. Одной из причин является недостаточное освещение рабочего места. Поэтому в ходе работы следует позаботиться об освещении помещения.

Для того, чтобы найти необходимое количество светильников, можно воспользоваться следующей формулой:

$$N = E_n * S * K_z * Z / (n * \Phi),$$

где N – число светильников в помещении;

E_n – минимальная освещенность помещения;

S – площадь помещения;

K_z – коэффициент запаса;

Z – коэффициент неравномерности освещения;

n – коэффициент использования светового потока;

Φ – световой поток.

По всем нормам, минимальная освещенность помещения E_n должна быть равна примерно 300 лк. Значение коэффициента запаса для сухих помещений принимается равной 1.4. Для освещения помещения будут применяться люминесцентные лампы, для которые коэффициент неравномерности будет равен 1.1.

Размеры помещения – 3 на 3 метра. По этим данным, можно сразу вычислить площадь, она будет равна:

$$S = 3 \text{ м} * 3 \text{ м} = 9 \text{ м}^2.$$

После этого можно приступить к расчету индекса помещения, который находится по формуле:

$$i = S / (hp * (W+L)),$$

где, i – индекс помещения;

hp – высота светильников над поверхностью стола;

W – ширина помещения;

L – длина помещения.

Размеры комнаты 3 на 3 метра, высота светильников над столом равна 1.8 метров.

$$i = 3*3 / (1.8 * (3+3)) = 0,8.$$

Коэффициента отражения потолка $R_{\text{п}}$ – 70%, потому что потолок является белой поверхностью. Коэффициент отражения стен $R_{\text{с}}$ равен 50%, так как представляет собой светлую поверхность. По таблице был определен коэффициент использования светового потока n , он равен 0,27.

Световой поток для ламп ЛБ40 равен 2800 лм.

Подсчитаем количество светильников типа ЛПО с двумя лампами:

$$N = 300 * 9 * 1.4 * 1.1 / (0,27 * 2800 * 2) = 3.$$

Для освещения комнаты необходимо 3 светильника типа ЛПО.

Заключение

В результате выполнения выпускной квалификационной работы было разработано мобильное приложение для поиска медицинских учреждений. Приложение открывает пользователям широкий доступ возможностей. Был разработан поиск по карте с помощью строки, составление маршрута, определение текущей геопозиции, отображение всех доступных медицинских учреждений на карте, изменение системы измерения, изменение внешнего вида карты. Для проведения первичной диагностики была использована нейронная сеть, сформированы наборы данных для обучения и тестирования нейронной сети, разработан сервер. После прохождения первичной диагностики пользователь получает сообщение с названием предполагаемого заболевания и его вероятность. При диагностике с помощью экспертной системы пользователь получает сообщение о наиболее вероятном заболевании. Было создано руководство по оказанию первой медицинской помощи для большого количества заболеваний. Реализовано две цветовые схемы для удобства использования приложения. Приложение доступно на английском и русском языках.

Список использованных источников

1. Адриан Форти; [пер. с англ. И. Форонова]– Дизайн и общество с 1750 года. – Издательство Студии Артемия Лебедева, 2021. – 456 с.
2. Документация *MapKit*. [Электронный ресурс]: Режим доступа: <https://yandex.ru/dev/maps/mapkit/doc/intro/concepts/about.html?from=mapkit> (дата обращения 02.05.2022).
3. Ян Пойнтер – Програмируем с *PyTorch*: Создание приложений глубокого обучения. – СПб.: Питер, 2019. – 256 с.
4. Эндрю Траск – Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.
5. Мозговой М.В. – C++ мастер-класс. 85 нетривиальных проектов, решений и задач. – СПб.: Наука и техника, 2007. – 272 с.

Приложение А

(Обязательное)

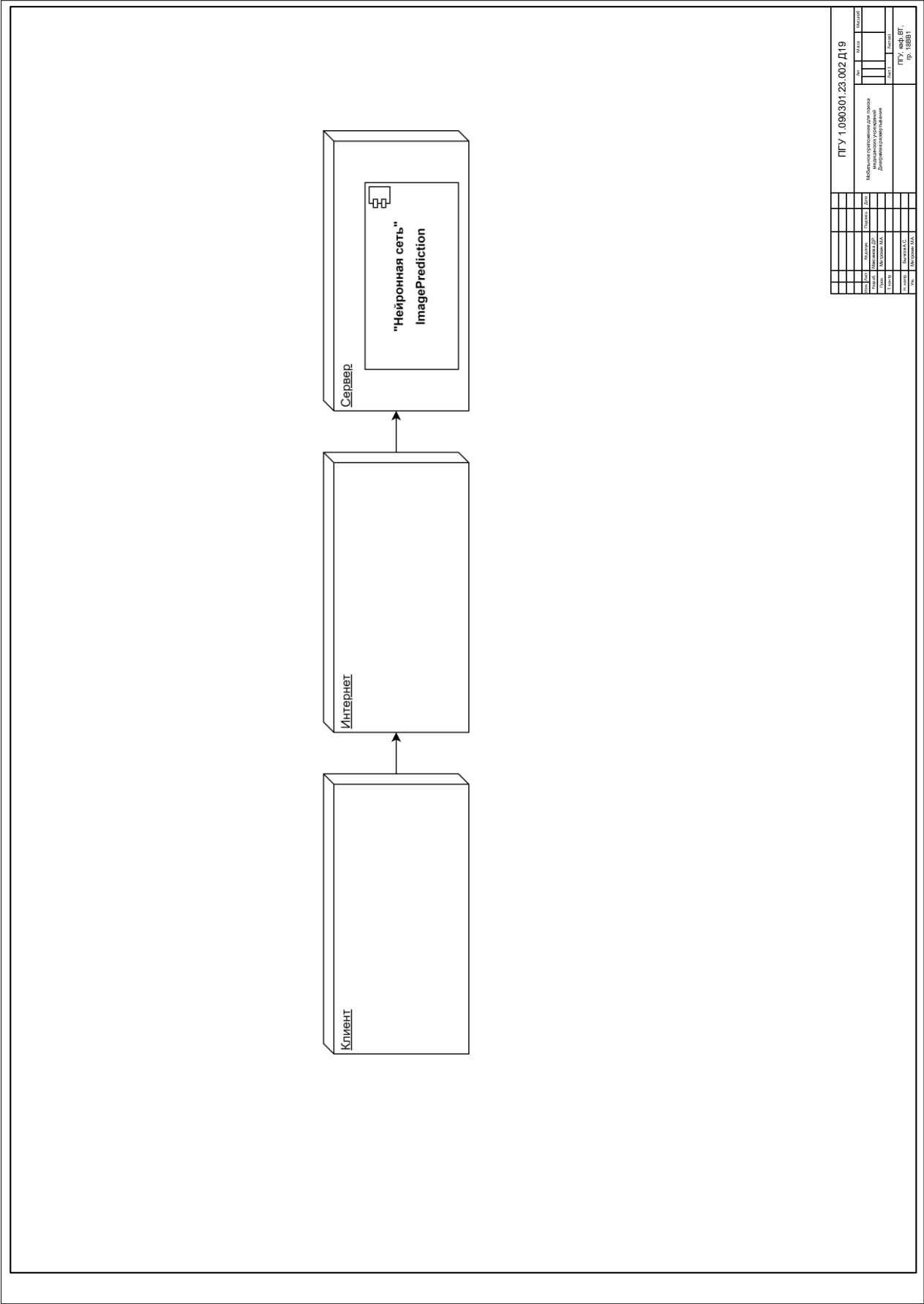
Мобильное приложение для поиска медицинских учреждений

UML-диаграммы

					<i>ПГУ 1.090301.23.001 ПЗ</i>	Лист
						54
Изм.	Лист	№ докум	Подпись	Дата		

Приложение А.1 - Диаграмма развертывания

					<i>ПГУ 1.090301.23.001 ПЗ</i>	Лист
						55
Изм.	Лист	№ докум	Подпись	Дата		

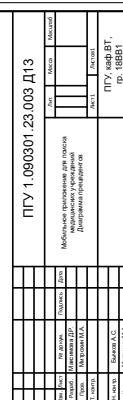


ПГУ 1.090301.23.002 Д19									
Изм.	Лист	№ докум	Подпись	Дата	Исполн.	Провер.	Утверд.	Введ.	Внес.
Модель и структура для задачи нейронной сети									
Детальная информация									
ПГУ 002 ВГ, р. 188В1									

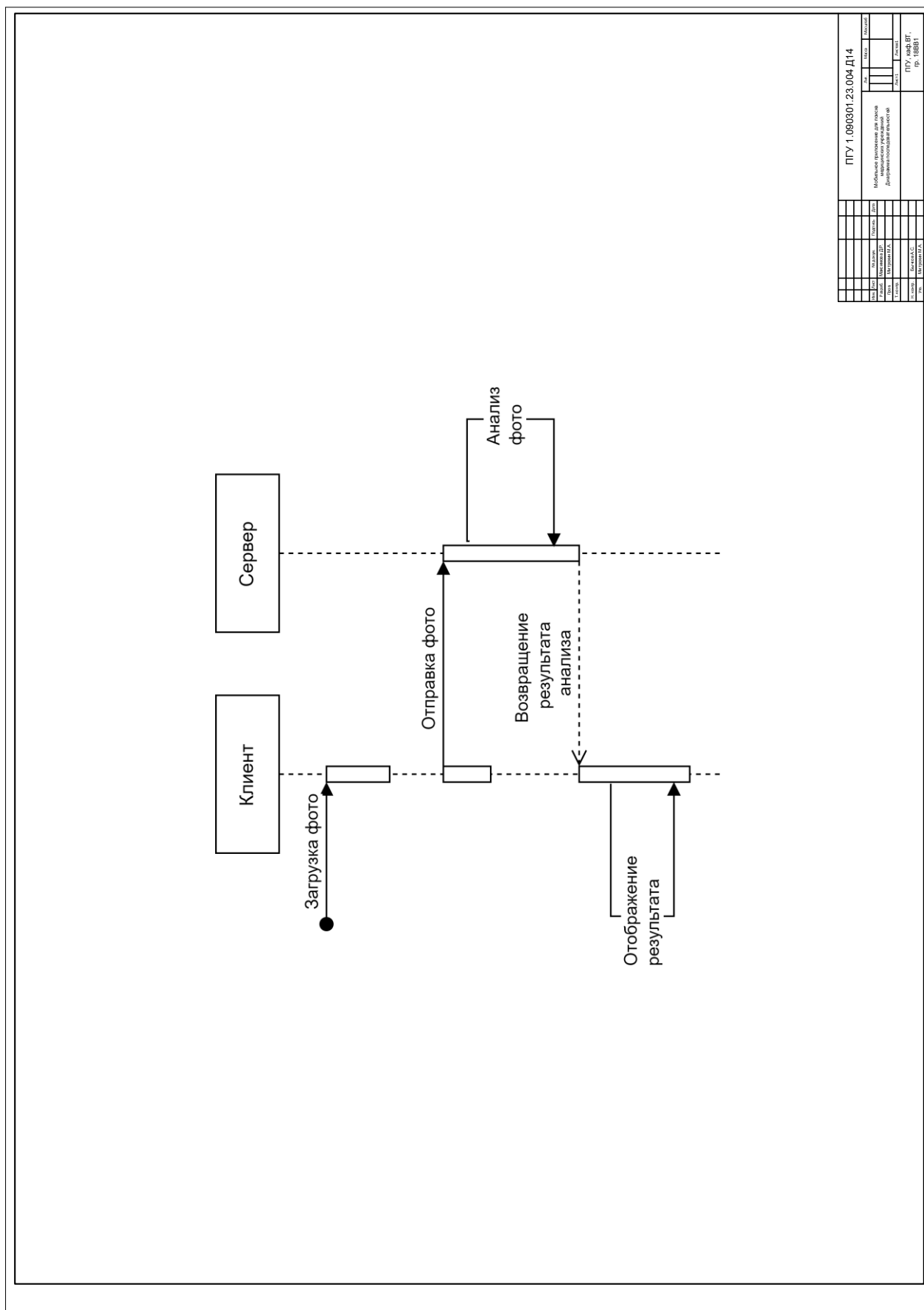
Изм.	Лист	№ докум	Подпись	Дата

Приложение А.2 - Диаграмма прецедентов

					ПГУ 1.090301.23.001 ПЗ	Лист
						57
Изм.	Лист	№ докум	Подпись	Дата		



Приложение А.3 - Диаграмма последовательностей



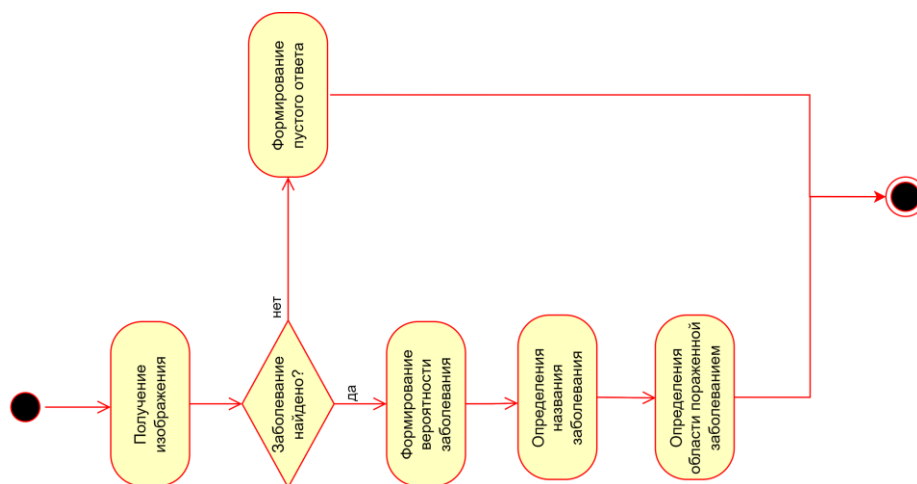
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

59

Приложение А.4 - Диаграмма деятельности сервера



ПГУ 1.090301.23.005 Д17									
Модуль встраиваемого для контроля и диагностики оборудования									
Изм.	Лист	№ докум	Подпись	Дата	Изм.	Лист	№ докум	Подпись	Дата
1	1	1			1	1	1		
2	2	2			2	2	2		
3	3	3			3	3	3		
4	4	4			4	4	4		
5	5	5			5	5	5		
6	6	6			6	6	6		
7	7	7			7	7	7		
8	8	8			8	8	8		
9	9	9			9	9	9		
10	10	10			10	10	10		
11	11	11			11	11	11		
12	12	12			12	12	12		
13	13	13			13	13	13		
14	14	14			14	14	14		
15	15	15			15	15	15		
16	16	16			16	16	16		
17	17	17			17	17	17		
18	18	18			18	18	18		
19	19	19			19	19	19		
20	20	20			20	20	20		
21	21	21			21	21	21		
22	22	22			22	22	22		
23	23	23			23	23	23		
24	24	24			24	24	24		
25	25	25			25	25	25		
26	26	26			26	26	26		
27	27	27			27	27	27		
28	28	28			28	28	28		
29	29	29			29	29	29		
30	30	30			30	30	30		
31	31	31			31	31	31		
32	32	32			32	32	32		
33	33	33			33	33	33		
34	34	34			34	34	34		
35	35	35			35	35	35		
36	36	36			36	36	36		
37	37	37			37	37	37		
38	38	38			38	38	38		
39	39	39			39	39	39		
40	40	40			40	40	40		
41	41	41			41	41	41		
42	42	42			42	42	42		
43	43	43			43	43	43		
44	44	44			44	44	44		
45	45	45			45	45	45		
46	46	46			46	46	46		
47	47	47			47	47	47		
48	48	48			48	48	48		
49	49	49			49	49	49		
50	50	50			50	50	50		
51	51	51			51	51	51		
52	52	52			52	52	52		
53	53	53			53	53	53		
54	54	54			54	54	54		
55	55	55			55	55	55		
56	56	56			56	56	56		
57	57	57			57	57	57		
58	58	58			58	58	58		
59	59	59			59	59	59		
60	60	60			60	60	60		
61	61	61			61	61	61		
62	62	62			62	62	62		
63	63	63			63	63	63		
64	64	64			64	64	64		
65	65	65			65	65	65		
66	66	66			66	66	66		
67	67	67			67	67	67		
68	68	68			68	68	68		
69	69	69			69	69	69		
70	70	70			70	70	70		
71	71	71			71	71	71		
72	72	72			72	72	72		
73	73	73			73	73	73		
74	74	74			74	74	74		
75	75	75			75	75	75		
76	76	76			76	76	76		
77	77	77			77	77	77		
78	78	78			78	78	78		
79	79	79			79	79	79		
80	80	80			80	80	80		
81	81	81			81	81	81		
82	82	82			82	82	82		
83	83	83			83	83	83		
84	84	84			84	84	84		
85	85	85			85	85	85		
86	86	86			86	86	86		
87	87	87			87	87	87		
88	88	88			88	88	88		
89	89	89			89	89	89		
90	90	90			90	90	90		
91	91	91			91	91	91		
92	92	92			92	92	92		
93	93	93			93	93	93		
94	94	94			94	94	94		
95	95	95			95	95	95		
96	96	96			96	96	96		
97	97	97			97	97	97		
98	98	98			98	98	98		
99	99	99			99	99	99		
100	100	100			100	100	100		

ПГУ, код БГ,
г. 18881

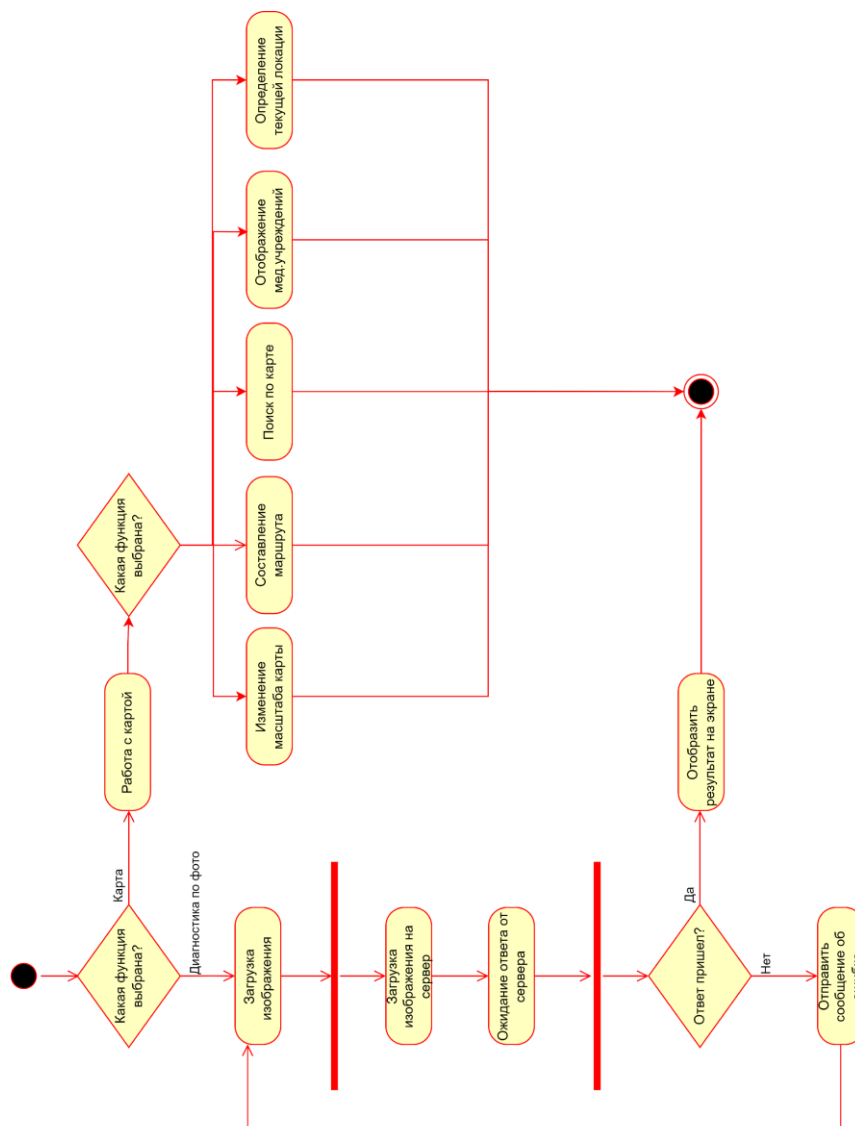
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

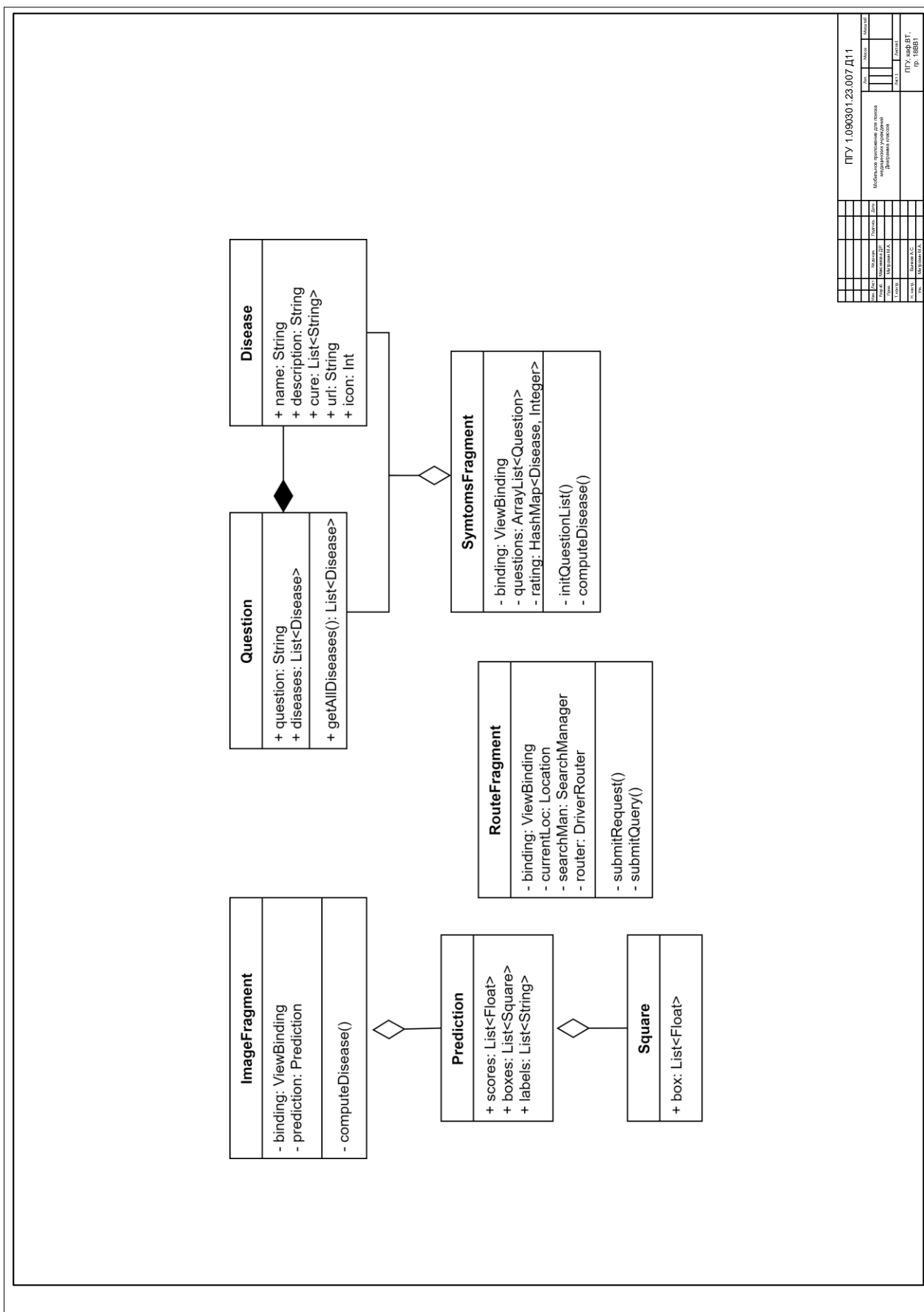
Лист

60

Приложение А.5 - Диаграмма деятельности клиента

[illegible]

Приложение А.6 - Диаграмма классов



ПУУ 1.090301.23.007 Д11		Имя	Фамилия	Имя	Фамилия
Министерство образования и науки Республики Беларусь		Имя	Фамилия	Имя	Фамилия
Институт информатики		Имя	Фамилия	Имя	Фамилия
ПУУ «Информатика»		Имя	Фамилия	Имя	Фамилия
г. Минск		Имя	Фамилия	Имя	Фамилия

Изм.	Лист	№ докум	Подпись	Дата

ПУУ 1.090301.23.001 ПЗ

Лист

Приложение Б

(Обязательное)

Мобильное приложение для поиска медицинских учреждений

Листинг программы

					<i>ПГУ 1.090301.23.001 ПЗ</i>	Лист
						63
Изм.	Лист	№ докум	Подпись	Дата		

Файл RouteFragment.java

<...>

```
public class RouteFragment extends Fragment implements
DrivingSession.DrivingRouteListener, Session.SearchListener,
CameraListener {

    public static final String DEFAULT_QUERY = "hospital";

    FragmentRouteBinding binding;
    Location curLoc = new Location();
    MapKit mapKit;

    private final Point SCREEN_CENTER = new Point(
        (ROUTE_START_LOCATION.getLatitude() +
ROUTE_END_LOCATION.getLatitude()) / 2,
        (ROUTE_START_LOCATION.getLongitude() +
ROUTE_END_LOCATION.getLongitude()) / 2);
    private MapObjectCollection mapObjects;
    private DrivingRouter drivingRouter;
    private DrivingSession drivingSession;
    private SearchManager searchManager;
    private Session searchSession;

    LocationListener unsub = new LocationListener() {
        @Override
        public void onLocationUpdated(@NonNull Location
location) {
            Log.d("TagCheck", "LocationUpdated " +
location.getPosition().getLongitude());
            Log.d("TagCheck", "LocationUpdated " +
location.getPosition().getLatitude());
            curLoc = location;
            binding.mapview.getMap().move(
                new CameraPosition(location.getPosition(),
14.0f, 0.0f, 0.0f),
```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

64

```

        new Animation(Animation.Type.SMOOTH, 1),
        null);

binding.mapview.getMap().getMapObjects().addPlacemark(new
Point(location.getPosition().getLatitude(),
location.getPosition().getLongitude()));
    }

    @Override
    public void onLocationStatusUpdated(@NonNull
LocationStatus locationStatus) {

    }

};

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup
        container, @Nullable Bundle savedInstanceState) {
    binding = FragmentRouteBinding.inflate(inflater);
    return binding.getRoot();
}

@Override
public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
    MapKitFactory.initialize(requireContext());
    DirectionsFactory.initialize(requireContext());
    SearchFactory.initialize(requireContext());
    mapKit = MapKitFactory.getInstance();
    super.onViewCreated(view, savedInstanceState);

    drivingRouter =

```

```

DirectionsFactory.getInstance().createDrivingRouter();

        mapObjects =
binding.mapview.getMap().getMapObjects().addCollection();

        //submitRequest();

        //binding.mapview.getMap().move(new
CameraPosition(SCREEN_CENTER, 5, 0, 0));

        searchManager =
SearchFactory.getInstance().createSearchManager(SearchManagerType.COMBINED);

        //binding.mapview.getMap().addCameraListener(this);
        //submitQuery(DEFAULT_QUERY);


mapKit.createLocationManager().subscribeForLocationUpdates(0, 0,
0, true, FilteringMode.ON, unsub);

        binding.currentLocation.setOnClickListener(v ->
binding.mapview.getMap().move(new
CameraPosition(curLoc.getPosition(), 14.0f, 0.0f, 0.0f),
                new Animation(Animation.Type.SMOOTH, 1),
                null));

        binding.plus.setOnClickListener(v ->
binding.mapview.getMap().move(new
CameraPosition(binding.mapview.getMap().getCameraPosition().getT
arget(),

binding.mapview.getMap().getCameraPosition().getZoom() + 1,
0.0f, 0.0f),

                new Animation(Animation.Type.SMOOTH, 1),
                null));

        binding.minus.setOnClickListener(v ->
binding.mapview.getMap().move(new

```

					<p style="text-align: center;"><i>ПГУ 1.090301.23.001 ПЗ</i></p>	Лист
Изм.	Лист	№ докум	Подпись	Дата		66

```

CameraPosition(binding.mapview.getMap().getCameraPosition().getTarget(),

binding.mapview.getMap().getCameraPosition().getZoom() - 1,
0.0f, 0.0f),

        new Animation(Animation.Type.SMOOTH, 1),
        null));

    }

    private void submitRequest() {
        DrivingOptions drivingOptions = new DrivingOptions();
        VehicleOptions vehicleOptions = new VehicleOptions();
        ArrayList<RequestPoint> requestPoints = new
ArrayList<>();
        requestPoints.add(new RequestPoint(
            ROUTE_START_LOCATION,
            RequestPointType.WAYPOINT,
            null));
        requestPoints.add(new RequestPoint(
            ROUTE_END_LOCATION,
            RequestPointType.WAYPOINT,
            null));
        drivingSession =
drivingRouter.requestRoutes(requestPoints, drivingOptions,
vehicleOptions, this);
    }

    private void submitQuery(String query) {
        searchSession = searchManager.submit(
            query,

VisibleRegionUtils.toPolygon(binding.mapview.getMap().getVisible
Region()),

        new SearchOptions(),
        this);
    }

```

```

    }

    @Override
    public void onDrivingRoutes(List<DrivingRoute> routes) {
        for (DrivingRoute route : routes) {
            mapObjects.addPolyline(route.getGeometry());
            mapObjects.addPlacemark(
                ROUTE_END_LOCATION,
                ImageProvider.fromResource(requireContext(),
R.drawable.loc));
            mapObjects.addPlacemark(
                ROUTE_START_LOCATION,
                ImageProvider.fromResource(requireContext(),
R.drawable.loc));
        }
    }

    @Override
    public void onDrivingRoutesError(Error error) {
        Toast.makeText(requireContext(), "MapKit error!",
Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onStop() {
        binding.mapview.onStop();
        MapKitFactory.getInstance().onStop();
        super.onStop();
        mapKit.createLocationManager().unsubscribe(unsub);
    }

    @Override
    public void onStart() {

```

```

        super.onStart();
        MapKitFactory.getInstance().onStart();
        binding.mapview.onStart();
    }

    @Override
    public void onCameraPositionChanged(@NonNull Map map,
        @NonNull CameraPosition cameraPosition, @NonNull
        CameraUpdateReason cameraUpdateReason, boolean finished) {
        if (finished) {
            submitQuery(DEFAULT_QUERY);
        }
    }

    @Override
    public void onSearchResponse(@NonNull Response response) {
        MapObjectCollection mapObjects =
        binding.mapview.getMap().getMapObjects();
        mapObjects.clear();

        for (GeoObjectCollection.Item searchResult :
        response.getCollection().getChildren()) {
            Point resultLocation =
            searchResult.getObj().getGeometry().get(0).getPoint();
            if (resultLocation != null) {
                mapObjects.addPlacemark(
                    resultLocation,

                    ImageProvider.fromResource(requireContext(),
                    R.drawable.search_result));
            }
        }
    }

    @Override

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

69

```

        public void onSearchError(@NonNull Error error) {
            Toast.makeText(requireContext(), "Search error!",
Toast.LENGTH_SHORT).show();
        }
    }
}

```

Файл ImageActivity.java

<...>

```

public class ImageActivity extends AppCompatActivity {
    ActivityImageBinding binding;

    Uri imageUri;

    public static String EXTRA_URI = "EXTRA_URI";

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        binding =
ActivityImageBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        String uri = getIntent().getStringExtra(EXTRA_URI);
        imageUri = Uri.parse(uri);
        binding.returnedImage.setImageURI(imageUri);

        Thread thread = new Thread() {
            @Override
            public void run() {
                Log.d("TAG", "run: ggg");
            }
        };
    }
}

```

```

        DiseaseAPI service =
provideRetrofit().create(DiseaseAPI.class);

        InputStream is = null;

        try {

            int nRead;

            byte[] data = new byte[16384];

            is =
getContentResolver().openInputStream(imageUri);

            ByteArrayOutputStream buffer = new
ByteArrayOutputStream();

            while ((nRead = is.read(data, 0,
data.length)) != -1) {

                buffer.write(data, 0, nRead);

            }

            String imgBase64 =
Base64.encodeToString(buffer.toByteArray(), Base64.DEFAULT);

            Call<Prediction> response =
service.listRepos(imgBase64);

            Response<Prediction> pred =
response.execute();

            Prediction predictionValue = pred.body();

            Log.d("TAG", "PREDICT: " + predictionValue);

            runOnUiThread(new Runnable() {

                @Override

                public void run() {

                    // change UI elements here

                    String diseases = "";

                    for(int i=0;
i<predictionValue.getLabels().size(); i++) {

```

```

                                diseases +=
predictionValue.getLabels().get(i)

                                + " " +
(predictionValue.getScores().get(i) * 100) + "%\n";

                                }

                                binding.diagnosis.setText("Possible
disease: " + diseases);

//binding.returnedImage.drawRectangles(predictionValue.getBoxes(
));

                                }

                                });

                                } catch (FileNotFoundException e) {

                                e.printStackTrace();

                                } catch (IOException e) {

                                e.printStackTrace();

                                }

                                }

                                };

                                thread.start();

                                }

private Retrofit provideRetrofit() {

                                Gson gson = new GsonBuilder().setLenient().create();

                                OkHttpClient.Builder httpClient = new
OkHttpClient.Builder().retryOnConnectionFailure(true);

                                httpClient.connectTimeout(60, TimeUnit.SECONDS);

                                httpClient.readTimeout(60, TimeUnit.SECONDS);

```

```

        httpClient.addInterceptor(new Interceptor() {

            @Override

            public okhttp3.Response intercept(Chain chain)
throws IOException {

                Request original = chain.request();

                Request request =

original.newBuilder().header("Connection", "close")

                                .header("Accept", "image/png")

                                .header("Content-Type",

"image/jpg")

                                .header("Content-Type",

"application/json")

                                .header("Accept-Encoding",

"identity")

                                .method(original.method(),

original.body()).build();

                return chain.proceed(request);

            }

        });

        return new

Retrofit.Builder().baseUrl("http://10.0.2.2:5000/")

.addConverterFactory(GsonConverterFactory.create(gson))

.addConverterFactory(ScalarsConverterFactory.create()).client(ht

tpClient.build()).build();

    }

}

```

Файл SymptomsFragment.java

<...>

```

public class SymptomsFragment extends Fragment {

    FragmentSymptomsBinding binding;

    ArrayList<Question> questions = new ArrayList<Question>();

    int index = 0;

    HashMap<Question.Disease, Integer> rating = new
    HashMap<Question.Disease, Integer>();

    @Nullable

    @Override

    public View onCreateView(@NonNull LayoutInflater inflater,
    @Nullable ViewGroup container, @Nullable Bundle
    savedInstanceState) {

        binding = FragmentSymptomsBinding.inflate(inflater);

        initQuestionList();

        binding.question.setText(questions.get(index).question);

        binding.yes.setOnClickListener(v -> {

            if (index < questions.size() && index > -1) {

                Question q = questions.get(index);

                if (rating.containsKey(q.question)) {

                    for (Question.Disease disease : q.diseases)

                        rating.put(disease,
rating.get(q.question) + 1);

                }
            }
        })
    }
}

```

```

        } else {

            for (Question.Disease disease : q.diseases)
            {

                rating.put(disease, 1);

            }

        }

        index++;

        if (index >= questions.size()) {

            computeDisease();

            index = -1;

            rating = new HashMap<Question.Disease,
Integer>();

        }

        else {

binding.question.setText(questions.get(index).question);

        }

    });

    binding.no.setOnClickListener(v -> {

        index++;

        if (index >= questions.size()) {

            computeDisease();

            index = -1;

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

75

```

        rating = new HashMap<Question.Disease,
Integer>();

        } else {

binding.question.setText(questions.get(index).question);

        }

    });

    return binding.getRoot();

}

private void computeDisease() {

    Question.Disease disease = null;

    int max = -1;

    for (Question.Disease key : rating.keySet()) {

        if (rating.get(key) > max) {

            disease = key;

            max = rating.get(key);

        }

    }

    if (disease==null || disease.equals("")) {

        binding.question.setText("You are healthy!");

    } else {

        binding.question.setText("Probably you have " +
disease.name);

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

76

```

    }

}

private void initQuestionList() {

    questions.add(new Question("Do you have sneezing?",
Arrays.asList(Question.Allergy, Question.Covid, Question.Flu)));

    questions.add(new Question("Do you have a runny or
blocked nose?", Arrays.asList(Question.Allergy, Question.Flu,
Question.Covid)));

    questions.add(new Question("Do you have red, itchy,
watery eyes?", Arrays.asList(Question.Allergy)));

    questions.add(new Question("Do you have a red, itchy
rash?", Arrays.asList(Question.Allergy)));

    questions.add(new Question("Do you have worsening of
asthma or eczema symptoms?", Arrays.asList(Question.Allergy)));

    questions.add(new Question("Do you have a cough,
shortness of breath or difficulty breathing?",
Arrays.asList(Question.Allergy, Question.Covid, Question.Flu)));

    questions.add(new Question("Did your skin swell and turn
red or dark blue?", Arrays.asList(Question.Chilblains)));

    questions.add(new Question("Did your skin break and sore
or blisters form?", Arrays.asList(Question.Chilblains,
Question.Allergy)));

    questions.add(new Question("Do you have a burning and
itching sensation on your skin(especially when you're in a warm
room)?", Arrays.asList(Question.Chilblains)));

```

```
questions.add(new Question("Do you feel yourself thirsty  
and lightheaded?", Arrays.asList(Question.Dehydration,  
Question.Flu, Question.Covid)));
```

```
questions.add(new Question("Do you have a dry mouth?",  
Arrays.asList(Question.Dehydration)));
```

```
questions.add(new Question("Do you have dark coloured,  
strong-smelling urine?", Arrays.asList(Question.Dehydration)));
```

```
questions.add(new Question("Do you pass urine less often  
than usual?", Arrays.asList(Question.Dehydration)));
```

```
questions.add(new Question("Do you have a tingling,  
itching or burning sensation around your mouth?",  
Arrays.asList(Question.ColdSore)));
```

```
questions.add(new Question("Have you lost your sense of  
smell and taste?", Arrays.asList(Question.Covid)));
```

```
questions.add(new Question("Do you have muscle or body  
aches?", Arrays.asList(Question.Covid, Question.Flu)));
```

```
questions.add(new Question("Do you have a fever or  
chills?", Arrays.asList(Question.Covid, Question.Flu,  
Question.FoodPoisoning)));
```

```
questions.add(new Question("Do you have a vomiting or  
diarrhea?", Arrays.asList(Question.Covid,  
Question.FoodPoisoning)));
```

```
questions.add(new Question("Do you have a high  
temperature (fever) of 38C (100.4F) or above?",  
Arrays.asList(Question.Flu, Question.Covid,  
Question.FoodPoisoning)));
```

```
questions.add(new Question("Do you have stomach  
cramps?", Arrays.asList(Question.FoodPoisoning)));
```

```

    }

}

}

```

Файл MainActivity.java

<...>

```

public class MainActivity extends AppCompatActivity {

    ActivityMainBinding binding;

    private FirebaseAnalytics mFirebaseAnalytics;

    public String uriImg;

    @Override

    protected void onCreate(@Nullable Bundle savedInstanceState)
    {

        //          mFirebaseAnalytics =
        FirebaseAnalytics.getInstance(this);

        super.onCreate(savedInstanceState);

        setContentView(R.layout.action_item_layout);

        binding =
        ActivityMainBinding.inflate(getLayoutInflater());

        binding.bottomAppBar.setBackground(null);

        if (ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED ||

```

					ПГУ 1.090301.23.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		79

```

        ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this,

            new
String[]{Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION},

            112);

    }

    setContentView(binding.getRoot());

    NavController navController = ((NavHostFragment)
(getSupportFragmentManager()).

findFragmentById(R.id.fragmentContainerView)).getNavController(
);

    NavigationUI.setupWithNavController(binding.bottomNavigationView
, navController);

    binding.bottomNavigationView.setOnNavigationItemSelectedListener
(item ->

        {

            if (item.getItemId() !=
binding.bottomNavigationView.getSelectedItemId())

                NavigationUI.onNavDestinationSelected(item,
navController);

            return true;
        }
    )
}

```

```

    });

    ActivityResultLauncher<String> mGetContent =
registerForActivityResult(new
ActivityResultContracts.GetContent(),

    uri -> {

        // Handle the returned Uri

        Intent intent = new Intent(this,
ImageActivity.class);

        uriImg = uri.toString();

        intent.putExtra(ImageActivity.EXTRA_URI,
uriImg);

        startActivity(intent);

    });

    binding.fab.setOnClickListener(new
View.OnClickListener() {

        @Override

        public void onClick(View view) {

            // Pass in the mime type you'd like to allow the
user to select

            // as the input

            mGetContent.launch("image/*");

        }

    });

}

}

```

Приложение В

(Обязательное)

Мобильное приложение для поиска медицинских учреждений

Презентация

					<i>ПГУ 1.090301.23.001 ПЗ</i>	Лист
						82
Изм.	Лист	№ докум	Подпись	Дата		

Мобильное приложение для поиска медицинских учреждений

Выполнила: Максимова Д.Р., гр. 18ВВ1

Руководитель: Митрохин М.А.



Проблема

Нет удобного поиска по медицинским учреждениям

Невозможность **первичной диагностики** заболевания самостоятельно

Затруднительный доступ к информации о медицинской помощи



Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.23.001 ПЗ

Лист

83

Q Средства разработки

- Java
- Android Studio
- XML
- MapKit
- Google Colab
- Python
- PyCharm
- PyTorch
- Retrofit



Q Диаграммы

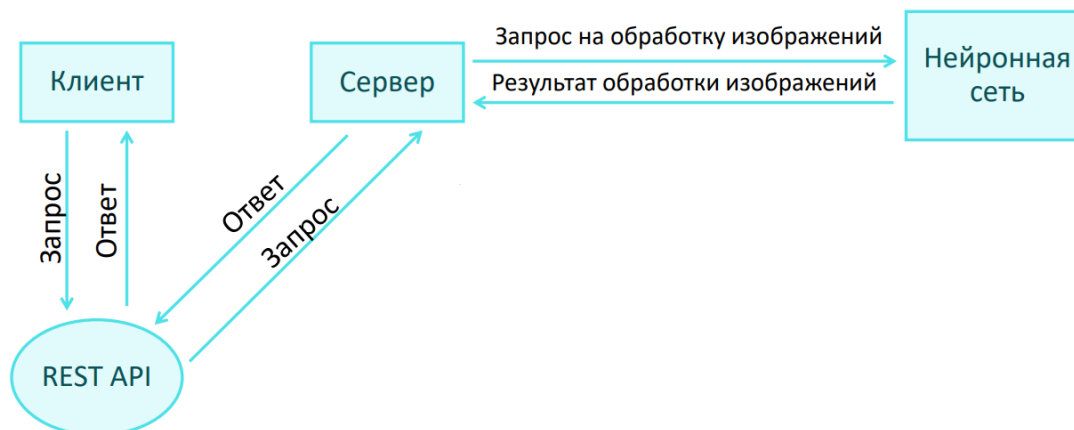
Структурная ×

Вариантов
использования ×

Классов ×

Структура приложения

×



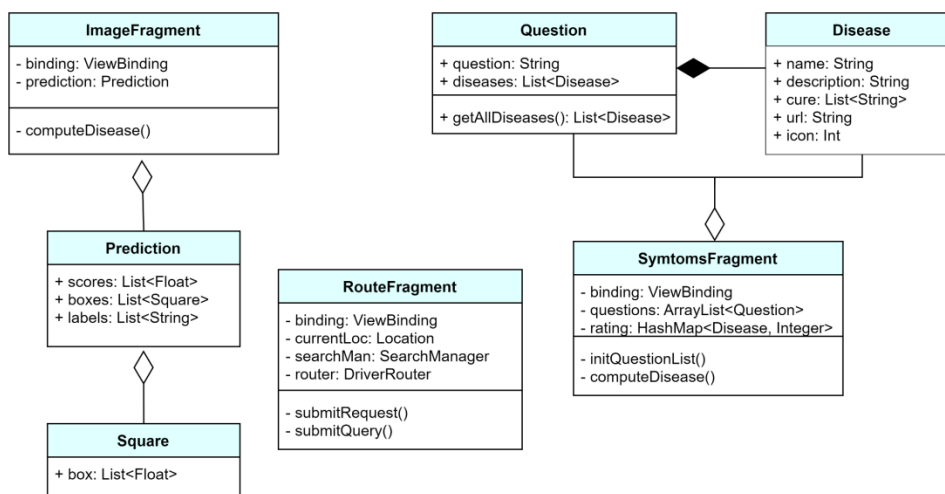
Вариантов использования

×



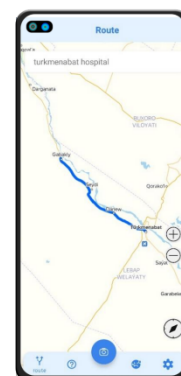
Диаграмма классов

×



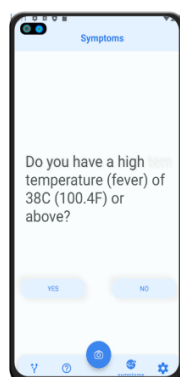
Карты

- Составление маршрута
- Отображение всех медучреждений на карте
- Поиск по медучреждениям
- Определение текущей геопозиции
- Изменение масштаба

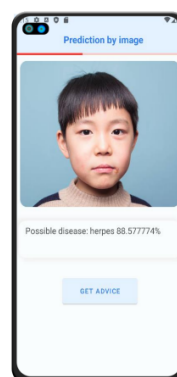


Диагностика

Опрос



Изображение

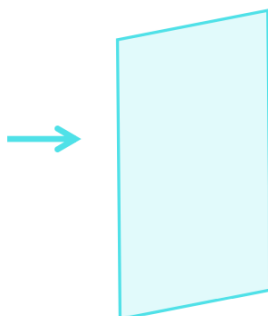


Нейронная сеть

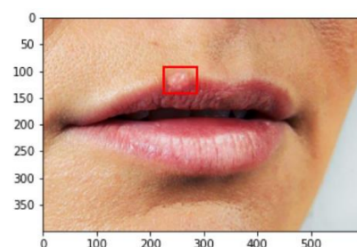
Фото



Faster R-CNN

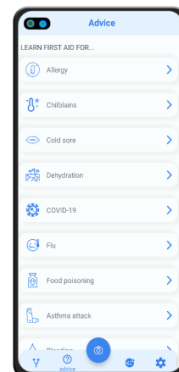


Результат



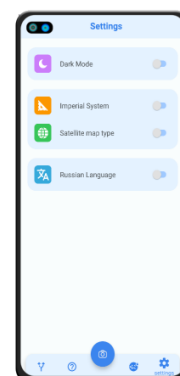
Рекомендации

Пошаговое руководство по оказанию первой медицинской помощи



Персонализация

Пользователь может менять внешний вид приложения и карт в зависимости от предпочтений



Изм.	Лист	№ докум	Подпись	Дата

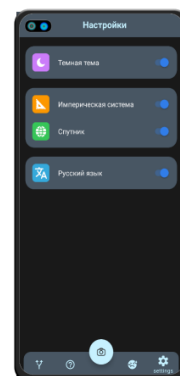
ПГУ 1.090301.23.001 ПЗ

Лист

88

🔍 Персонализация

Пользователь может менять внешний вид приложения и карт в зависимости от предпочтений



🔍 Заключение

Развитие проекта

Усовершенствование нейронной сети, увеличение количества изображений и заболеваний

Мобильный **ассистент** для медработников

Приложение помогающее в **обучении** студентам мединститут

Изм.	Лист	№ докум	Подпись	Дата