|  |  |  |
| --- | --- | --- |
| |  | | --- | | Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* | | Кирьянов Сергей Вячеславович  **Разработка системы диагностики остеопороза по КТ-снимкам позвоночника**  *Курсовая работа*  студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель  Преподаватель кафедры информационных технологий в бизнесе НИУ ВШЭ-Пермь    М. Д. Чистогов | |

**Пермь, 2024**

**Содержание**

[**Введение** 3](#_Toc161851967)

[**1** **Анализ предметной области** 5](#_Toc161851968)

[**1.1** **Данные о патологии** 5](#_Toc161851969)

[**1.2** **Данные для обучения системы** 6](#_Toc161851970)

[**1.3** **Анализ DICOM-формата** 7](#_Toc161851971)

[**1.4** **Анализ средств для разработки и обучения модели** 11](#_Toc161851972)

[**1.5** **Модели искусственного интеллекта, применяемые в медицине** 12](#_Toc161851973)

[**2** **Обработка наборов данных** 14](#_Toc161851974)

[**2.1** **Отбор и разметка данных для обучения системы** 14](#_Toc161851975)

[**2.2** **Предобработка данных** 17](#_Toc161851976)

[**2.3** **Подготовка данных к использованию в обучении, аугментация** 22](#_Toc161851977)

[**3** **Разработка и обучение модели** 25](#_Toc161851978)

[**3.1** **Выбор модели нейронной сети** 25](#_Toc161851979)

[**3.2** **Создание и обучение модели** 27](#_Toc161851980)

[**3.3** **Тестирование модели** 32](#_Toc161851981)

[**3.4** **Проверка совместимости системы с записями для селф-тестирования** 36](#_Toc161851982)

[**4** **Заключение** 39](#_Toc161851983)

[**5** **Список используемой литературы** 40](#_Toc161851984)

[ПРИЛОЖЕНИЕ А **Техническое задание** 42](#_Toc161851985)

[ПРИЛОЖЕНИЕ Б **Руководство пользователя** 47](#_Toc161851986)

[ПРИЛОЖЕНИЕ В **Код модели Animal** 60](#_Toc161851987)

[ПРИЛОЖЕНИЕ С **Остальной код приложения** 62](#_Toc161851988)

**Введение**

В настоящее время искусственный интеллект (ИИ) проникает во все сферы человеческой деятельности, и медицина не является исключением. Системы машинного обучения все чаще используются в диагностике заболеваний, помогая врачам делать более точные и быстрые вывода. Использование ИИ в медицине открывает новые возможности для улучшения качества медицинских услуг и уменьшения трудозатрат медицинских работник, что, в свою очередь, может способствовать удешевлению некоторых медицинских услуг.

Искусственный интеллект уже сейчас используется в различных областях медицины, в том числе, на территории Российской Федерации. Чаще всего системы диагностики заболеваний на основе ИИ используются в таких областях медицины, как радиология, патология, дерматология, офтальмология. Информационные системы на основе машинного обучения позволяют создавать алгоритмы, которые способны анализировать большие объемы медицинских данных, выявляя закономерности и расширяя возможности диагностики заболеваний. Эти системы могут помогать врачам в выявлении рака на ранних стадиях, диагностике пневмонии и COVID-19, обнаружении патологий опорно-двигательного аппарата и многих других заболеваниях.

Тем не менее, стоит понимать, что искусственный интеллект (по крайней мере, на данный момент) не исключает работу медицинских работников, и, в частности, подтверждение ими наличия того или иного заболевания. Такие системы используются исключительно для поддержки диагностирования, то есть, в первую очередь, для увеличения точности постановки диагноза специалистом.

Целью данной работы является разработка системы диагностики остеопороза по КТ-снимкам позвоночника.

Таким образом, предметом исследования является реализация системы диагностики остеопороза по КТ-снимкам позвоночника, а объектом исследования программные реализации систем с использованием искусственного интеллекта, предназначенных для использования в сфере здравоохранения, в частности, для анализа медицинских изображений.

Для достижения поставленной цели были сформулированы следующие задачи:

1. выполнить анализ предметной области, в том числе проанализировать клиническую задачу, решаемую ИИ-сервисом и признаки патологии (остеопороз), проанализировать требования к формату ответу ИИ-сервиса и форме его предоставления, осуществить поиск и анализ данных, подходящих для обучения системы, произвести анализ DICOM-формата, осуществить обзор и сравнение существующих аналогов ИИ-сервиса, а также рассмотреть существующие модели искусственного интеллекта (нейронных сетей), применяемые в медицине;
2. выполнить отбор данных для обучения системы, подготовить разметку данных, реализовать предобработку медицинских изображений для обучения системы, выбрать структуру и модель нейронной сети (обосновав выбор);
3. разработать модель и затем обучить её, произвести тестирование на размеченном наборе данных, с учетом различных метрик, проверить совместимость системы с записями из набора данных, предназначенного для селф-тестирования от ГБУЗ «НПКЦ ДиТ ДЗМ, реализовать предоставление ответа ИИ-сервиса для серии снимков в виде вероятности наличия патологии (в процентах);
4. осуществить разработку веб-приложения, проектирование пользовательского интерфейса.

Выбранные цели выполнялись последовательно, что способствовало поэтапной разработке. Также это помогло следовать корректному жизненному циклу программного обеспечения.

1. **Анализ предметной области**
   1. **Данные о патологии**

Данная система разрабатывается для диагностики остеопороза. Остеопороз - хронически прогрессирующее системное метаболическое (обменное) заболевание скелета или клинический синдром, проявляющийся при других заболеваниях и характеризующийся снижением плотности костей, нарушением их микроархитектоники и усилением хрупкости по причине нарушения метаболизма костной ткани с преобладанием катаболизма над процессами костеобразования, снижением прочности кости и повышением риска переломов. [ССЫЛКА](https://ru.wikipedia.org/wiki/Остеопороз)

Согласно данным ВОЗ, остеопороз поражает примерно 6,3% мужчин старше 50 лет и 21,2% женщин старше того же возраста во всем мире [ССЫЛКА](https://www.sciencedirect.com/science/article/abs/pii/S875632820700823X?via%3Dihub). Исходя из численности мирового населения, в соответствии с соотношением по полам, примерно 500 миллионов мужчин и женщин во всем мире могут быть пострадать от этого заболевания.

По оценкам, в 2019 году в Европе остеопорозом страдали 32 миллиона человек в возрасте от 50 лет и старше, что эквивалентно 5,6% от общей численности населения Европы в возрасте старше 50 лет. Из них, приблизительно 25,5 миллиона женщин (22,1% женского населения Европы в возрасте старше 50 лет) и 6,5 миллиона мужчин (6,6% от общей численности мужского населения Европы в возрасте старше 50 лет) [ССЫЛКА](https://pubmed.ncbi.nlm.nih.gov/34080059/).

Основные признаки, по которым обычно определяется наличие патологии (согласно данным от mosmed.ai) [ССЫЛКА](file:///C:\Users\apka2\Downloads\Базовые_диагностические_требования_14.08.2023.pdf):

1. наличие на нативных изображениях позвонков, имеющих компрессионную деформацию тел ≥ 25 %, по полуколичественной шкале Genant, 2–3 степень.
2. снижение минеральной плотности костной ткани в телах позвонков в интервале Th11 – L3 (оптимально L1– L2) согласно критериям ACR 2018, позиции ISCD 2019 на нативных изображениях.

Из-за малого объема данных, основным признаком, по которому будет определяться наличие патологии, это снижение минеральной плотности костной ткани в телах позвонков, по их аксиальным снимкам.

Таким образом, остеопороз – опасное и труднодиагностируемое заболевание, особенно, на ранних стадиях развития, которое может затронуть миллионы человеческих жизней.

* 1. **Данные для обучения системы**

Для обучения системы с сайта mosmed.ai (сайт центра диагностики и телемедицины) были взяты 3 набора данных КТ-снимков с признаками остеопороза позвоночника. Поставщиком данным в данном случае является ГБУЗ «НПКЦ ДиТ ДЗМ». Далее под записями будут подразумеваться наборы серий КТ-снимков.

Первый набор имеет 6 записей [ССЫЛКА](https://mosmed.ai/datasets/mosmeddata-kt-s-priznakami-osteoporoza-pozvonochnika-tip-iii-versiya-2/), а второй 10 записей [ССЫЛКА](https://mosmed.ai/datasets/mosmeddata-kt-s-priznakami-osteoporoza-pozvonochnika-tip-iii/). При этом, 2 записи из второго набора имеют технический дефект, о чем указано в описании. Таким образом, получается всего 14 записей, с бинарной разметкой (без признаков патологии; с признаками патологии). При этом, соотношение классов равно 5:9, то есть, записей без признаков патологии почти вдвое меньше, чем записей с признаками патологии, а по количеству самих снимков эта разница еще больше: снимков без признаков патологии почти в 3 раза меньше, чем снимков с признаками патологии.

Третий же набор предназначен для селф-тестирования по множеству патологий и имеет 7 подходящих записей [ССЫЛКА](https://mosmed.ai/datasets/aie20selftest/). Тем не менее, из-за низкого количества снимков без признаков патологии, были взяты данные и из данного набора (всего 2 записи из 7). Это позволило существенно увеличить качество системы.

Информация по всем наборам также представлены ниже в таблице 1.

Таблица 1 – Информация по используемым наборам данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник данных** | **Тип данных** | **Кол-во данных** | **Тип разметки данных** | **Соответствование классам** | **Уточнение** |
| ГБУЗ «НПКЦ ДиТ ДЗМ», mosmed.ai | DICOM, КТ-снимки, 3D | 10 | Категориальная. Классы: С патологией, Без патологии, Технический Дефект | 4;4;2 | MosMedData КТ с признаками остеопороза позвоночника тип III. |
| ГБУЗ «НПКЦ ДиТ ДЗМ», mosmed.ai | DICOM, КТ-снимки, 3D | 6 | Бинарная. Классы: Без патологии, с патологией | 1;5 | MosMedData КТ с признаками остеопороза позвоночника тип III, версия 2. |
| ГБУЗ «НПКЦ ДиТ ДЗМ», [mosmed.ai](http://mosmed.ai/) | DICOM, КТ-снимки, 3D | 7 (всего 41, остальные не соответствуют исследуемой патологии) | Бинарная. Классы: Без патологии, с патологией | - | Набор данных КТ, ММГ, РГ/ФЛГ с целью селф-тестирования ИИ-серсивов для поиска признаков приоритетных патологий |

Таким образом, данные, предназначенные для обучения системы, представлены в достаточно ограниченном объеме. Размета данных бинарная, что подразумевает также и бинарную классификацию (наличие или отсутствие патологии).

* 1. **Анализ DICOM-формата**

DICOM (Digital Imaging and Communications in Medicine) — медицинский отраслевой стандарт создания, хранения, передачи и визуализации цифровых медицинских изображений и документов обследованных пациентов [ССЫЛКА](https://ru.wikipedia.org/wiki/DICOM). Файлы DICOM имею расширение «.dcm».

DICOM-формат содержит много метаданных о пациенте, так называемых тэгов. При этом, часто многие такие метаданные удаляются из датасетов для соблюдения полной анонимности.

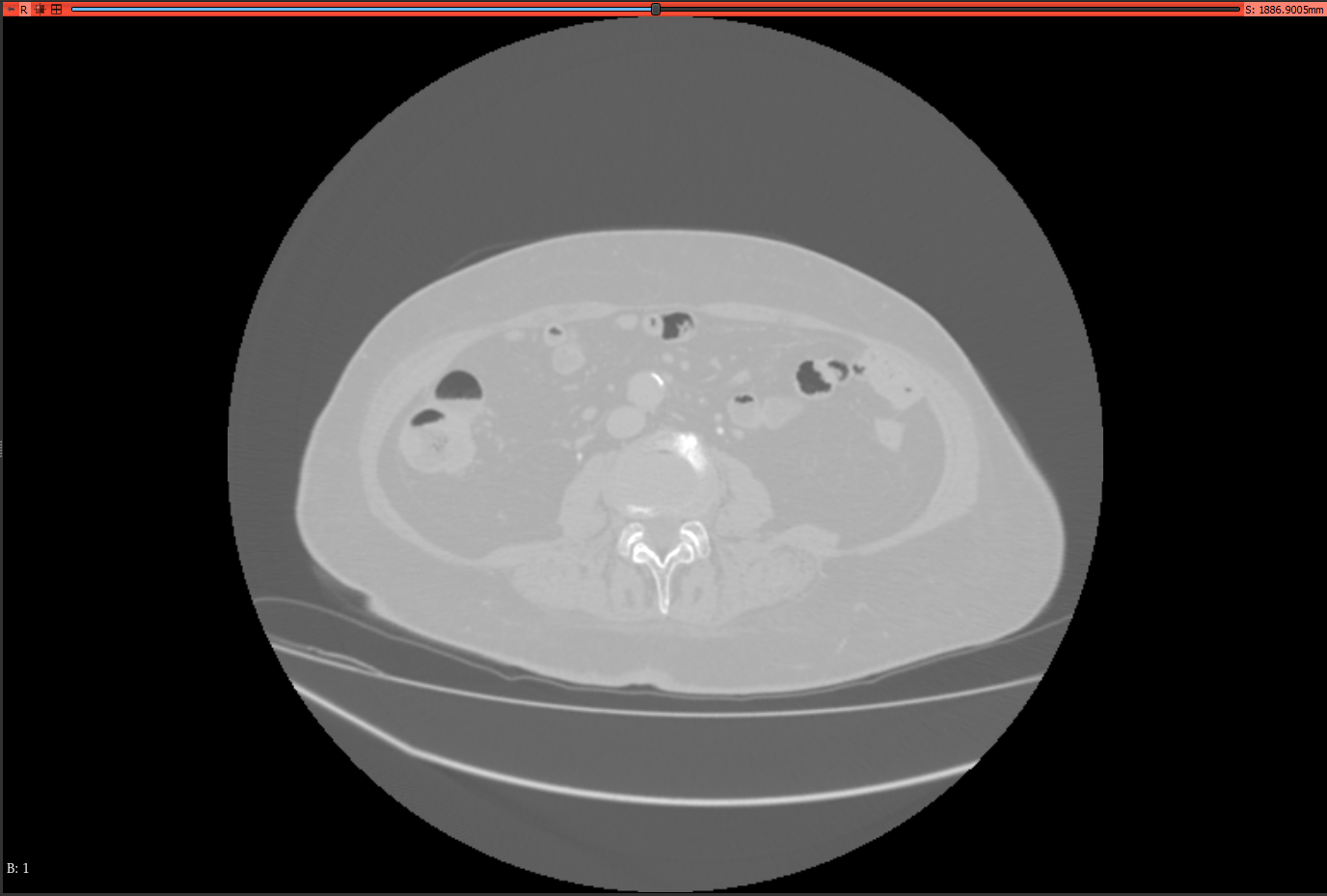
Примеры метаданных: положение пациента, вид проекции (снимка) – аксиальная, сагитальная или корональная, номер среза (под срезом подразумевается 1 снимок из серии), возраст пациента, пол пациента, данные о медицинской технике аппарате компьютерной томографии в данном случае), выдержка, фокусное расстояние, пиковое напряжение (для того, чтобы выделять те или иные ткани), изображение в пикселях (двумерный массив пикселей) и т.д. [ССЫЛКА](https://healtis.ru/articles/dicom-tags/).

Для обработки DICOM-формата на языке Python существует несколько библиотек, самой оптимальной (и часто используемой) является библиотека «Pydicom» [ССЫЛКА](https://pydicom.github.io/). Но, для корректной обработки тэга «pixel\_array», необходимо также наличие модуля «gdcmswig».

В настоящее время, при работе с DICOM-форматом в медицинских учреждениях (больницах, клиниках и т.д.), используются специализированные программы, позволяющие просматривать не только метаданные, но и непосредственно снимки в формате изображений, при чем, как в двумерном формате (в разных проекциях), так иногда и в трехмерном пространстве.

Примером такой программы является «3D Slicer» - бесплатное приложение с открытым исходным кодом для визуализации, обработки, сегментации, регистрации и анализа медицинских, биомедицинских и других 3D-изображений. [ССЫЛКА](https://slicer.readthedocs.io/en/latest/)

Примеры визуализации DICOM-изображений (из датасетов) в программе «3D Slicer» представлены ниже. Пример визуализации аксиальной проекции приведен на рисунке 1.



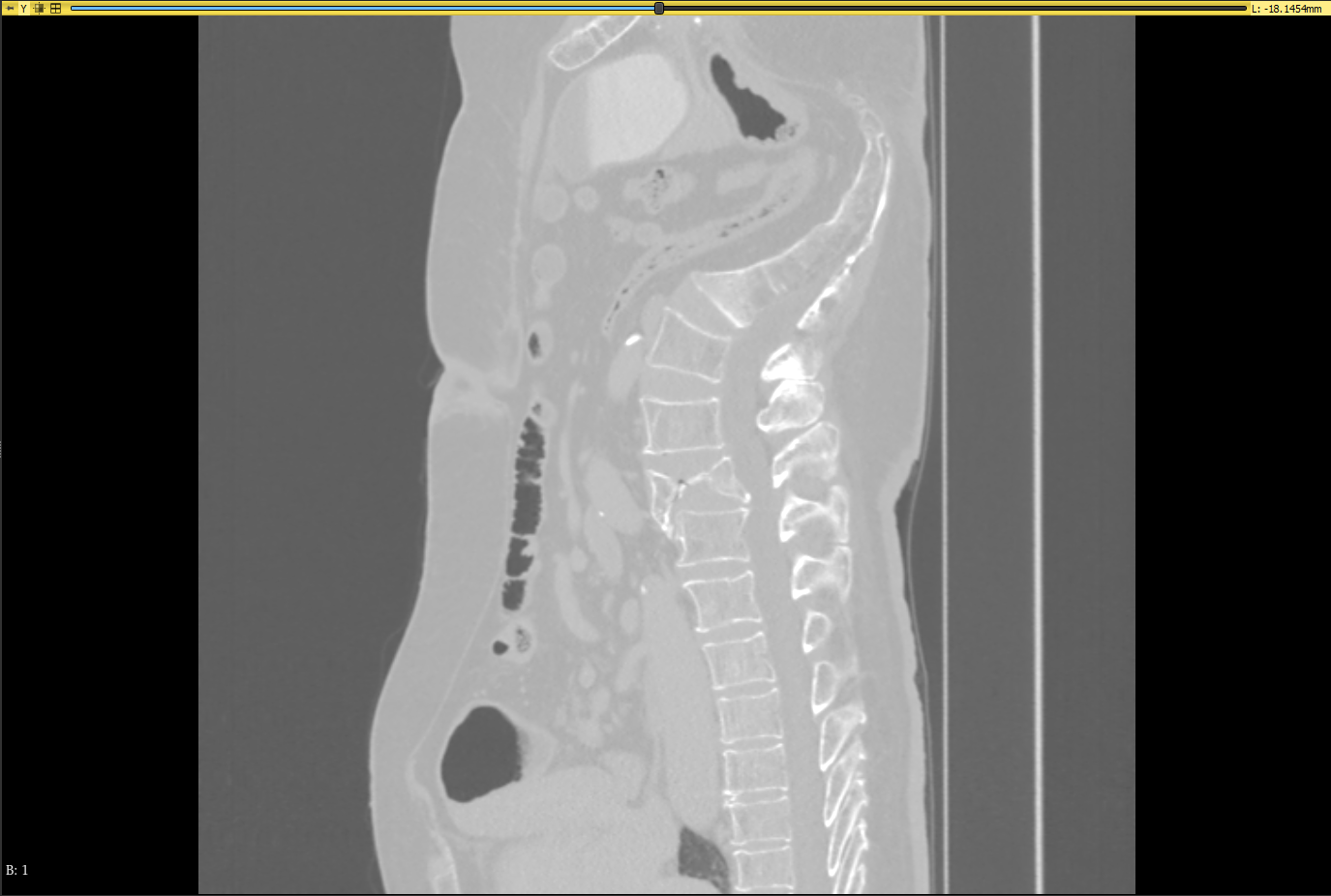
***Рисунок 1 – Пример визуализации аксиальной проекции в «3D Slicer»***

Пример визуализации корональной проекции приведен ниже, на рисунке 2.



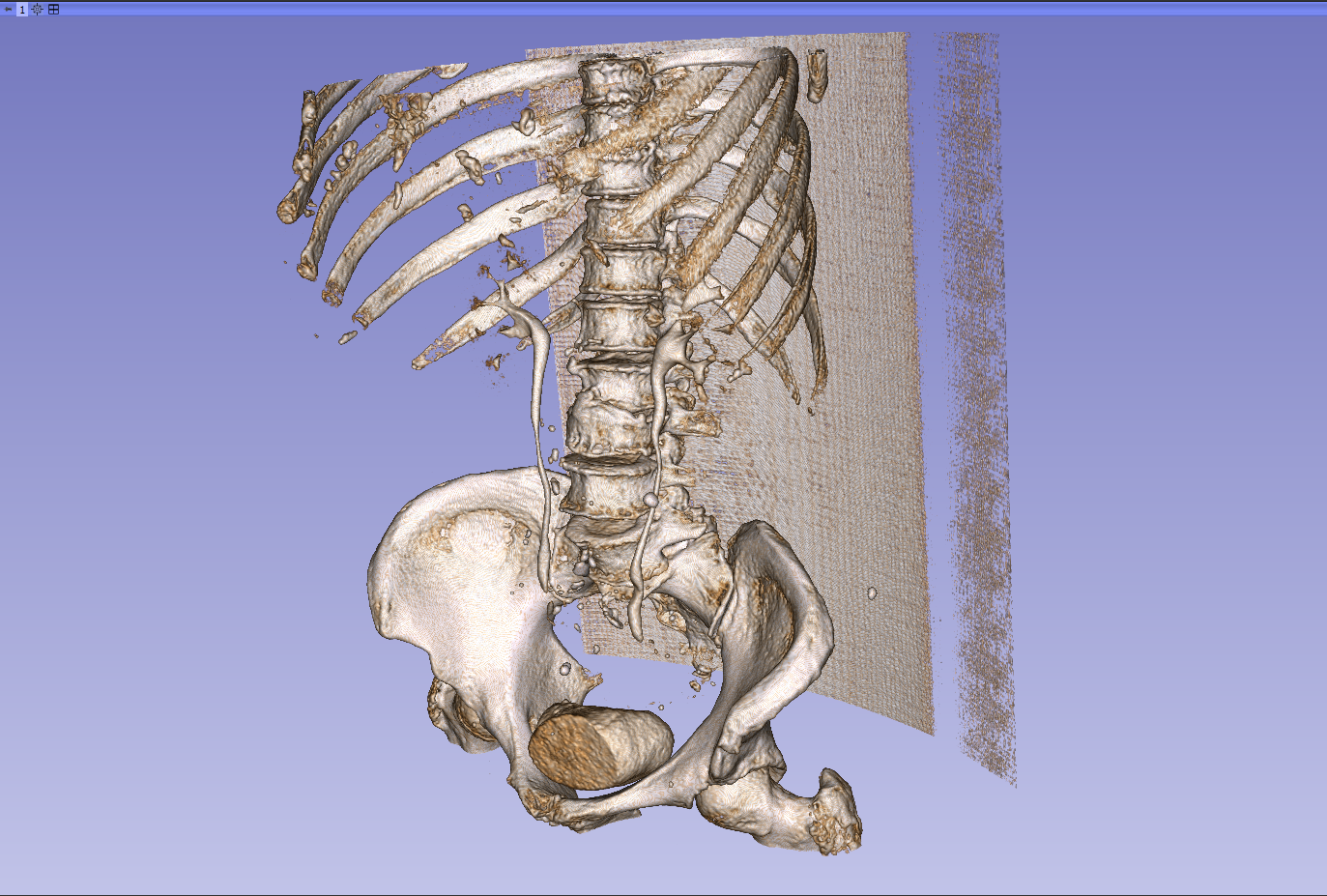
***Рисунок 2 – Пример визуализации корональной проекции в «3D Slicer»***

Визуализация сагитальной проекции изображена ниже, на рисунке 3.



***Рисунок 3 – Пример визуализации сагитальной проекции в «3D Slicer»***

Также, как уже было сказано ранее, «3D Slicer» позволяет создавать трехмерные проекции. Пример визуализации трехмерной проекции представлен ниже, на рисунке 4.



***Рисунок 4 – Пример визуализации трехмерной проекции в «3D Slicer»***

Существенным недостатком программы «3D Slicer» является то, что она достаточно тяжела в освоении, для использования продвинутого функционала требуется соответствующее обучение. Также, для корректной визуализации проекций (особенно трехмерной), необходима правильная сортировка файлов. По умолчанию, «3D Slicer» сортирует DICOM-файлы из указанного каталога (серию снимков) просто по их названиям, из-за чего порядок снимков становится неправильным (такая проблема наблюдается у всех серий КТ снимков из наборов данных от mosmed.ai). Для того, чтобы с этим бороться, можно перезаписать все файлы из серии, отсортировав их по тэгу «SliceLocation» (номер снимка/среза в списке срезов) или по тэгу «ImagePositionPatient» (номер снимка/среза в одной из проекций). Для визуализации примеров, представленных выше, такая сортировка и перезапись файлов была выполнена.

Пример визуализации различных проекций на ЯП Python, при помощи библиотек «Pydicom» и «Matplotlib», представлен ниже, на рисунке 5.



***Рисунок 5 – Пример визуализации различных проекций в Python***

При этом видно, что на данных проекциях костные ткани выделены ярче, нежели другие, мягкие ткани. Это было достигнуто при помощи специальных функций, встроенных в библиотеку «Pydicom». Речь о них пойдет далее, на этапе проектирования системы и предобработки данных.

Таким образом, DICOM-формат содержит в себе не только изображения КТ снимков, но и множество дополнительной информация, которая, в свою очередь, может быть использована как для преобразования самих изображений, построения разных проекций, так и в качестве дополнительной информации, обрабатываемой моделью.

* 1. **Анализ средств для разработки и обучения модели**

Разрабатывать и обучать модель можно было не только локально, используя ресурсы компьютера или ноутбука (использовался компьютер, технические характеристики будут описаны далее), но и используя среду разработку «Google Colab», используя, при этом, еще и облачные вычислительные мощности, предоставляемые компанией Google (аппаратные ускорители на базе ГПУ, а не ЦПУ, «TPU» и «T4 GPU»). Такие облачные вычислительные мощности позволяют существенно ускорить обучение модели, однако, имеют ограничение по вычислительным единицам. Это значит, что после определенного количества обучения модели, выбранный аппаратный ускоритель станет временно недоступным. При этом, может произойти внезапная остановка обучения модели, или же среды выполнения целиком, что доставляет существенные неудобства при многократном экспериментальном обучении модели, при подборе структуры модели и её гиперпараметров.

Также, при работе с «Google Colab» немного усложняется работа с данными, т.к. медицинские изображения в DICOM-формате, имеют очень большой вес (наборы данных весят десятки гигабайт), из-за чего загружать их в облако достаточно долго.

Как уже было сказано ранее, обучение модели происходит в разы быстрее при использовании мощностей ГПУ, т.к. архитектура графического процессора позволяет значительно лучше распараллеливать вычисления. [ССЫЛКА](https://bytexd.com/hardware/why-are-gpus-well-suited-for-machine-learning/).

Рассматривая локальные ресурсы, было выяснено, что их может вполне хватить для разработки и дальнейшего обучения модели, т.к. комплектующие компьютеры оказались достаточно мощными. Обучение модели производилось на ГПУ «RTX 3070» (5888 CUDA-ядер). Также, использовался 8-ядерный процессор «Intel Xeon e5-2689», с тактовой частотой 3.3 ГГц. Медицинские изображения хранились локально, на твердотельном накопителе.

При дальнейшем сравнении скорости обучения на видеокарте и на процессоре, было выяснено, что обучение на видеокарте происходит примерно в 9 раз быстрее, чем на процессоре (в конкретно моем случае).

Таким образом, было решено разрабатывать и обучать модель в среде разработки «Visual Studio Code», с использованием интерактивных блокнотов «Jupyter Notebooks», используя при этом локальные ресурсы персонального компьютера (в первую очередь, вычислительные мощности графического процессора).

* 1. **Модели искусственного интеллекта, применяемые в медицине**

Как уже было сказано ранее, ИИ уже применяется в медицине для ускорения работы врачей. И если ранее модели ИИ чаще всего применялись для выявления групп риска тех или иных заболеваний (например, группа риска сердечно-сосудистых заболеваний на основе данных о потреблении людьми алкоголя, занятии спортом, возрасте и т.д.), на основе различных текстовых или числовых данных, то теперь же искусственный интеллект (особенно, сверточные нейронные сети) все чаще и чаще применяют для анализа различных изображений, от простых фотографий (например, фотографии кожи с подозрением на дерматологические заболевания) до снимков, сделанных специальной аппаратурой (МРТ, КТ, Рентген и другие), а также, трехмерных проекций. Последние чаще всего используются именно при работе с КТ снимками, но, несмотря на то, что трехмерные проекции серий снимков можно было бы построить, таких данных бы не хватило для обучения трехмерной сверточной нейронной сети.

Таким образом, для диагностики остеопороза по КТ снимкам позвоночника, было решено использовать именно сверточную нейронную сеть, при чем, именно предобученную модель.

1. **Обработка наборов данных**
   1. **Отбор и разметка данных для обучения системы**

Из-за того, что стандартные, размеченные наборы данных имели очень малое количество записей, было решено также использовать для обучения и записи из набора данных для проведения селф-тестирования. Были найдены 2 записи, судя по проекциям которых, патология отсутствовала. Эти записи были размечены самостоятельно.

Также, не были использованы записи из второго набора данных от ГБУЗ «НПКЦ ДиТ ДЗМ», у которых было отмечено наличие технических дефектов.

Другие наборы данных, не от ГБУЗ «НПКЦ ДиТ ДЗМ», содержащие КТ-снимки позвоночника с признаками остеопороза, не были найдены. Существуют наборы данных на «Kaggle» на несколько сотен гигабайт с КТ-снимками позвонков шейного с признаками переломов, но, эти наборы данных не являются подходящими (т.к. стандартно требуется работа с позвонками брюшной полости, и, наличие переломов позвонков не гарантирует наличие остеопороза. Также, в таком наборе данных тяжело получить информацию о плотности костной ткани).

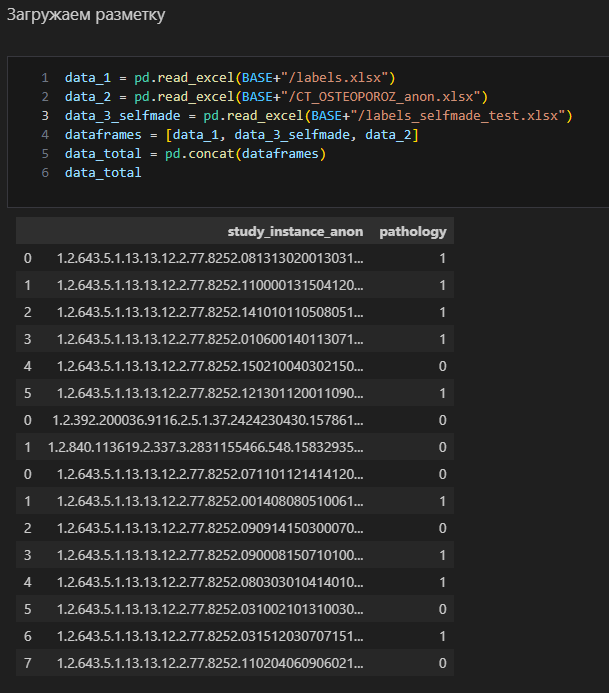
Разметка данных, как уже было сказано ранее, бинарная. Ключевым признаком является наличие патологии.

Также, из-за того, что записи в наборах данных включают в себя, в том числе, и схожие между собой серии КТ снимков (серии КТ снимков одного и того же пациента), было принято решение использовать для обучения модели снимки из первой серии каждой записи, чтобы уменьшить количество схожих изображений. В итоге, мы имеем около 2500 аксиальных изображений без признаков патологии, а с признаками патологии более около 4000. Часть из них будет использована для тестирования системы (не будет использоваться для обучения).

Стандартные размеры изображений (массивов) во всех сериях – 512 на 512 пикселей.

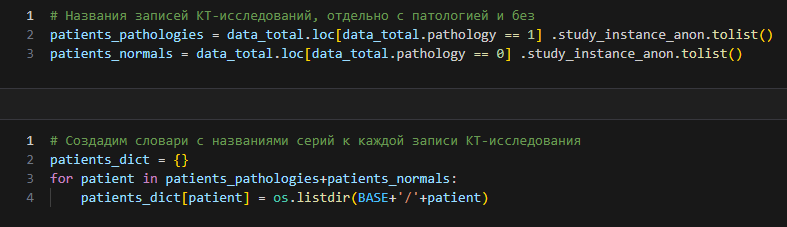
Сама разметка подгружается из нескольких Excel-таблиц, в которых название записи КТ исследования (столбец «study\_instance\_anon») и данные о наличии патологии (столбец «pathology»), 0 – признаки патологии не выявлены, 1 – признаки патологии выявлены. При этом, название записи КТ исследования соответствует названию каталога, в котором хранятся серии КТ снимков этой записи.

Пример загрузки разметки представлен на рисунке 6.



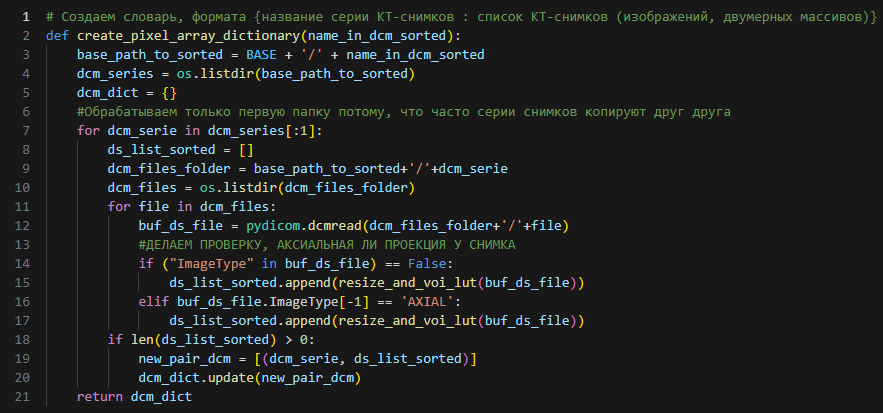
***Рисунок 6 – Пример загрузки разметки***

После этого, названия записей КТ исследований передаются в списки, отдельно для записей с признаками патологии и без признаков патологии. Далее, создаются словари с названиями серий к каждой записи КТ-исследования, при помощи которых далее данные и будут извлечены. Код, используемый для этой подготовки данных, изображен ниже, на рисунке 7.



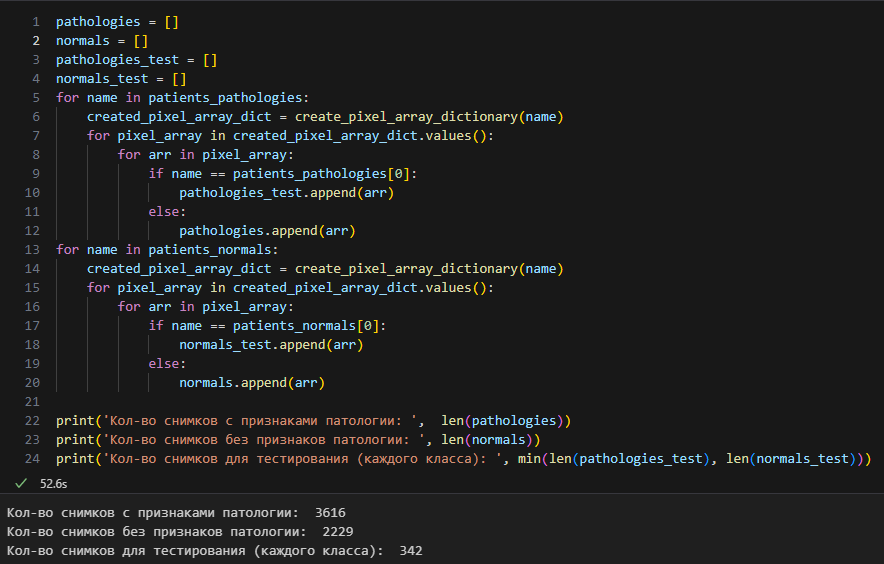
***Рисунок 7 – Код получения названий серий к каждой записи КТ исследования***

Далее, для работы непосредственно с самими изображения КТ снимков из каждой серии, при помощи функции «create\_pixel\_array\_dictionary», создается словарь, в котором название серии КТ снимков соотносится со списком изображений из этой серии. Код функции «create\_pixel\_array\_dictionary» представлен на рисунке 8. При этом можно заметить, что к изображениям применяется функция «resize\_and\_voi\_lut». Об этой функции будет рассказано далее.



***Рисунок 8 – Код функции «create\_pixel\_array\_dictionary»***

Затем, происходит извлечение снимков и их добавление в соответствующие общие списки (с признаками патологии и без признаков патологии), для дальнейшей обработки и использования в обучении модели. При этом, можно заметить, что снимки первой записи КТ исследования попадают в отдельные списки, предназначенные для финального тестирования модели, и не используются для её обучения. Код извлечения всех снимков представлен ниже, на рисунке 9.



***Рисунок 9 – Код извлечения всех изображений КТ снимков***

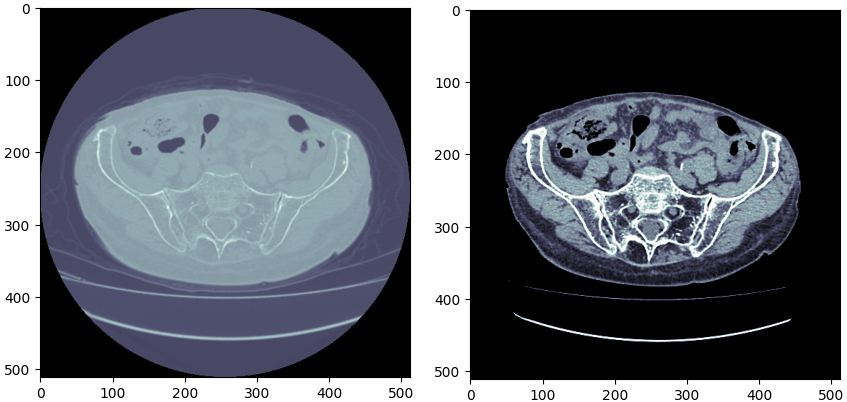
Таким образом, для обучения модели будет использовано около 2200, а для тестирования модели около 340 изображений каждого класса (с признаками и без признаков наличия остеопороза позвоночника).

* 1. **Предобработка данных**

В случае работы с медицинскими изображениями, важна и возможна не только базовая предобработка данных, такая как изменение размеров изображений, их нормализация и аугментация, для ускорения и улучшения качества обучения модели. Также, при помощи использования метаданных DICOM-формата и функций библиотеки «Pydicom», появляется возможность выделения тканей, рассматриваемых при постановке анализа. Это достигается при помощи использования функции «apply\_voi\_lut». Она преобразовывает исходное изображение в соответствии с данными из таблицы «VOI LUT», скрытой в метаданных. Благодаря этому, значительно ярче и контрастнее на изображении становятся костные ткани, необходимые для диагностики остеопороза. (Тип выделяемых тканей зависит от таблицы «VOI LUT». В зависимости от типа исследуемой патологии, могут выделять не только костные ткани, но и мягкие ткани, сосуды, внутренние органы и так далее.

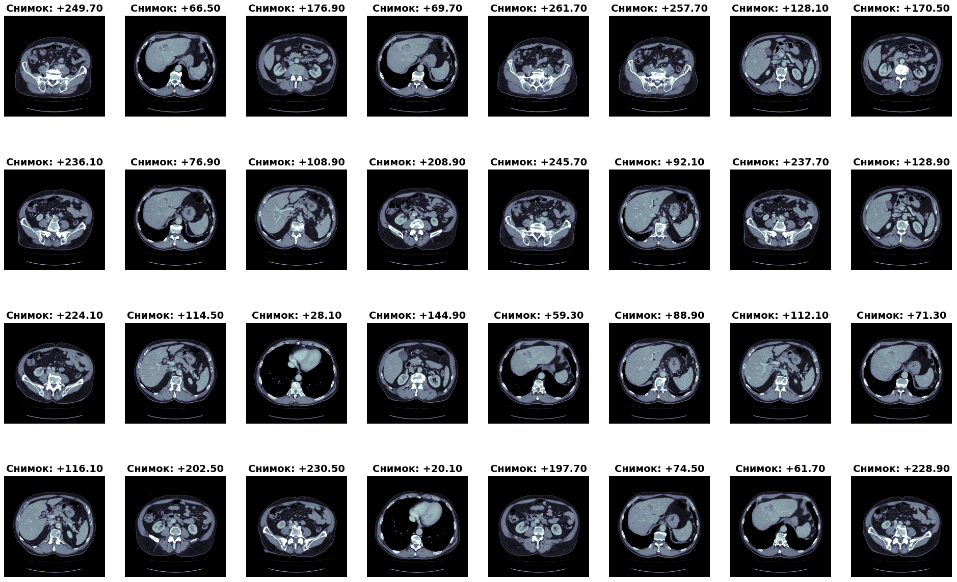
Получение двумерных массивов – изображений из DICOM файлов происходит через использование библиотеки «Pydicom», а вывод изображений и их частичная обработка, при помощи библиотек «matplotlib.pyplot» и «cv2» соответственно.

Пример работы функции «apply\_voi\_lut» представлен на рисунке 10. Слева изображение до обработки, справа – после. На изображении справа видно, что костные ткани стали значительно ярче и контрастнее.



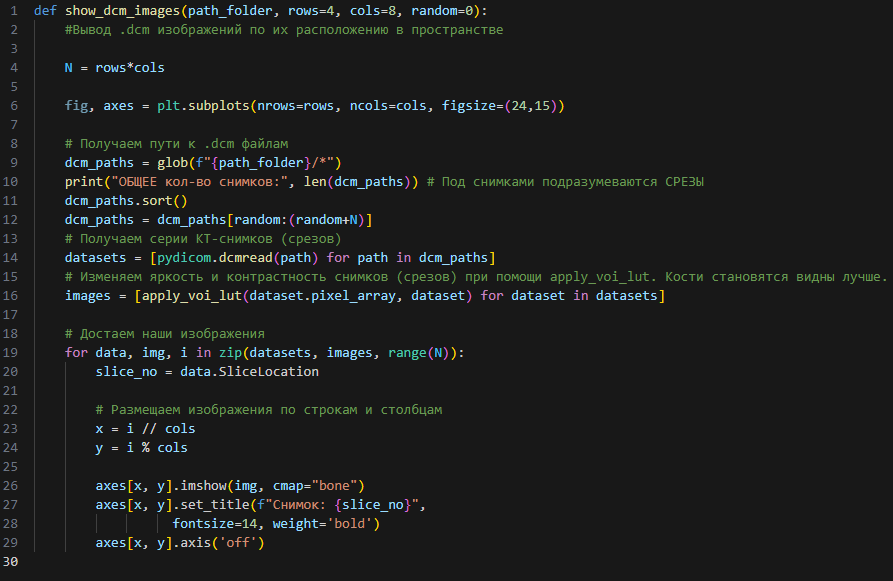
***Рисунок 10 – Пример работы функции «apply\_voi\_lut».***

Также, для дополнительной информативности, при разработке модели использовался вывод множества случайных аксиальных снимков, с указанием их положения (внутри серии). Пример такого вывода представлен ниже, на рисунке 11.



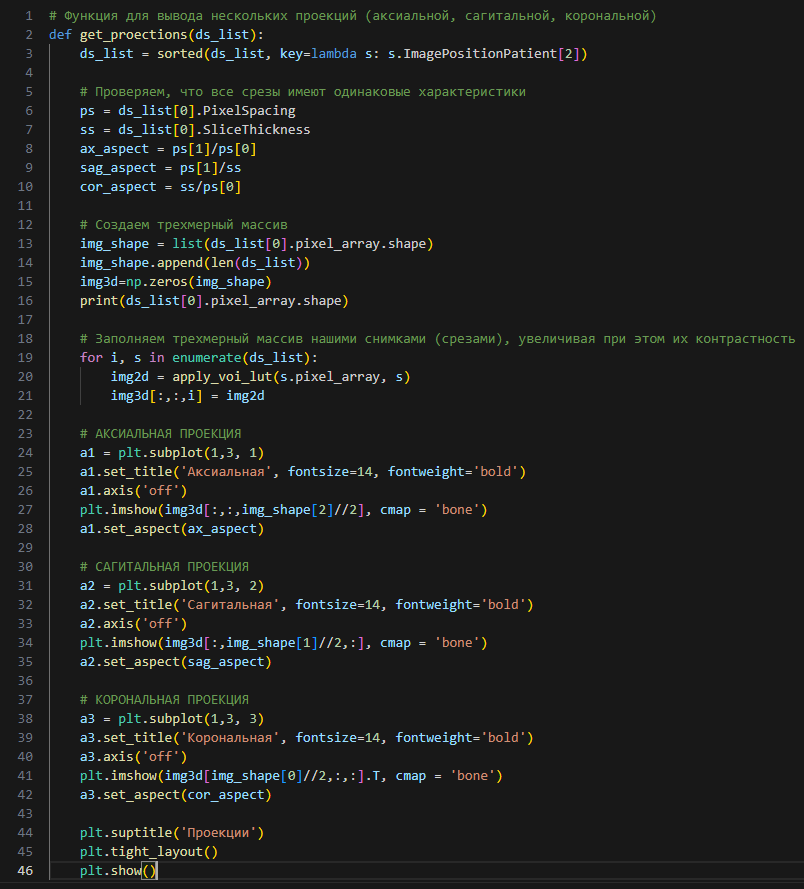
***Рисунок 11 – Пример вывода множества случайных аксиальных снимков***

Реализует такой вывод функция «show\_dcm\_images». Она получает на вход путь к серии КТ-снимков на диске и размеры сетки для снимков. Снимки из серии загружаются, к ним применяется функция «apply\_voi\_lut», после чего они размещаются по сетке и выводятся на экран. Код функции «show\_dcm\_images», реализующей такой вывод, представлен ниже, на рисунке 12.



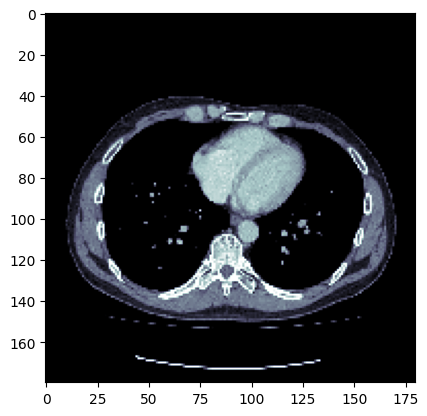
***Рисунок 12 – Код функции «show\_dcm\_images»***

Для вывода различных проекций серии КТ снимков, о которых было упомянуто ранее, на этапе анализа, была разработана функция «get\_proections», которая получает на вход список КТ-снимков в DICOM-формате, после чего из них выстраивается трехмерный массив пикселей, который уже после и преобразовывается в аксиальную, сагитальную и корональную проекции. Стоит отметить, что в данном случае очень важно отсортировать снимки по тэгу «SliceLocation» или же по «ImagePositionPatient[2]», иначе снимки будут перемешаны и проекции будут выстраиваться некорректно. В данной функции используется сортировка по второму тэгу. Код функции «get\_proections» представлен ниже, на рисунке 13.



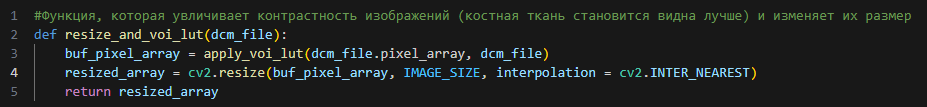
***Рисунок 13 – Код функции «get\_proections»***

Также, так как функция «apply\_voi\_lut» возвращает простой двумерный массив, его размер можно сразу же изменить, уменьшив в пределах 100-250 пикселей, для того, чтобы ускорять обучение нейронной сети и не вызывать переполнения оперативной памяти и видеопамяти ГПУ. Размер в 180 на 180 пикселей был выбран оптимальным размером изображений для обучения. Для одновременного применения функции «apply\_voi\_lut» и изменения размера изображений, была создана функция «resize\_and\_voi\_lut». Изображение, обработанное этой функцией, изображено ниже, на рисунке 14.



***Рисунок 14 – Изображение, обработанное функцией «resize\_and\_voi\_lut»***

Данная функция принимает на вход DICOM файл и возвращает преобразованное изображение в виде двумерного массива. Код функции «resize\_and\_voi\_lut» представлен на рисунке 15.

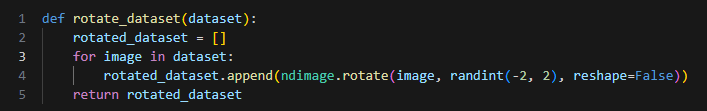


***Рисунок 15 – Код функции «resize\_and\_voi\_lut»***

Таким образом, использование особенностей DICOM-формата позволяет сделать специфическую обработку изображений из серии, а также, преобразовать их в различные проекции.

* 1. **Подготовка данных к использованию в обучении, аугментация**

Для борьбы с переобучением, была создана функция «rotate\_dataset», предназначенная для небольшого поворота изображений, в пределах 2 градусов. Несмотря на столь небольшой поворот изображений, такая аугментация данных всё же улучшает общее качество модели. Поворот изображений производится при помощи функции «rotate» из библиотеки «ndimage». Код функции «rotate\_dataset» изображен ниже, на рисунке 16.



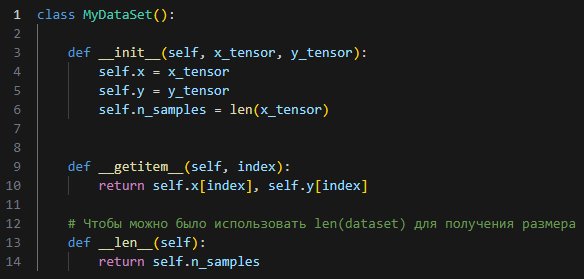
***Рисунок 16 – Код функции «rotate\_dataset»***

Сначала количество изображений каждого класса уравнивается, после чего все изображения поворачиваются случайным образом при помощи функции, описанной ранее. После этого, данные нормализуются, перемешиваются и преобразовываются в тензоры. Код, реализующий создание обработанных тензоров, изображен на рисунке 17.



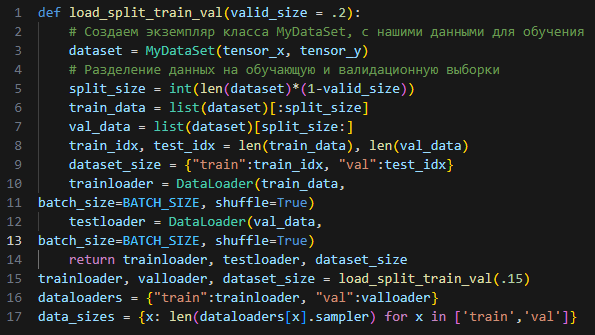
***Рисунок 17 – Код создания обработанных тензоров***

Далее, для того, чтобы использовать функционал класса «DataLoader» из библиотеки «PyTorch», был создан класс «MyDataSet», получающий на вход тензоры с изображениями и затем реализующий доступ к данным из тензора. Код класса «MyDataSet» представлен ниже, на рисунке 18.



***Рисунок 18 – Код класса «MyDataSet»***

Также, для удобного разделения данных на обучающую и валидационную выборки, была разработана функция «load\_split\_train\_val», которая делит данные по указанному соотношению и создает загрузчики данных класса «DataLoader». Данные в этих загрузчиках перемешиваются и собираются в батчи (оптимальный размер батча, который был подобран – 32). Из-за малого набора данных, для валидации использовалось только 15% данных. Код функции «load\_split\_train\_val» изображен на рисунке 19.



***Рисунок 19 – Код функции «load\_split\_train\_val»***

Таким образом, данные для обучения были нормализованы, аугментированы, преобразованы в тензоры и разделены на обучающую и валидационную выборки, с использованием батчей, размером 16.

1. **Разработка и обучение модели**
   1. **Выбор модели нейронной сети**

Чаще всего, по данным исследований [ССЫЛКА](https://www.researchgate.net/publication/369872580_Pathological_Images_Classification_based_on_the_Pretrained_Convolutional_Neural_Network), для классификации медицинских изображений используются такие предобученные сверточные нейронные сети, как ResNet50, ResNet152, DenseNet121, DenseNet161, VGG16, VGG19 и другие. Все они показывают очень высокую точность уже после десятка эпох обучения (в районе 90%), особенно, при использовании заранее подготовленных весов.

Фреймворк «PyTorch» позволяет использовать множество готовых моделей сверточных нейронных сетей вместе с готовыми весами, включая все те, которые были упомянуты выше. При этом, не получилось бы эффективно использовать слишком тяжелые модели, с слишком большим количеством параметров и требующие очень высокопроизводительные системы для обучения (исходя из данных о GFLOPS (кол-во операций с плавающей точкой в секунду)), потому что тогда эксперименты с моделью в ходе подбора гиперпараметров, вида нормализации и аугментации данных, попытки использования данных из сторонних наборов данных требовали бы слишком много времени (даже для обучения одной эпохи), а некоторые модели и вовсе не смогли бы обучаться локально, из-за недостатка вычислительных мощностей. [ССЫЛКА](https://pytorch.org/vision/stable/models.html)

Данные о предобученных нейронных сетях в фреймворке «PyTorch» представлены ниже, в таблице 2.

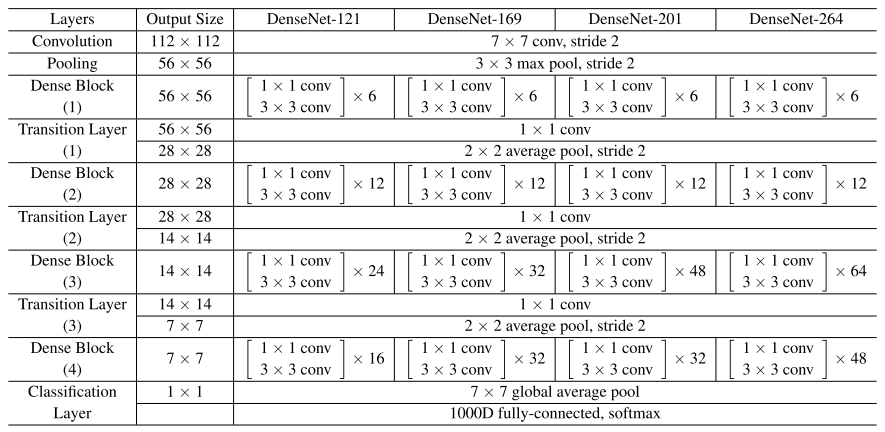
Таблица 2 – Информация о предобученных моделях фреймворка «PyTorch»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название нейронной сети** | **Кол-во параметров, миллионы параметров** | **Требование к производительности, GFLOPS** | **Точность после 5 эпох на наборе данных «ImageNet-1K»** |
| AlexNet | 61.1 | 0.71 | 79.066 |
| VGG16 | 138.4 | 15.47 | 90.382 |
| VGG19 | 143.7 | 19.63 | 90.876 |
| ResNet50 | 25.6 | 4.09 | 92.862 |
| ResNet152 | 60.2 | 11.51 | 94.046 |
| DenseNet121 | 8.0 | 2.83 | 91.972 |
| DenseNet151 | 28.7 | 7.73 | 93.56 |

Одно из самых лучших соотношений количества параметров с требованиями и производительности к точности наблюдается у нейронных сетей архитектуры DenseNet, особенно у DenseNet121.

Модели архитектуры DenseNet были созданы в 2016 году и представляют из себя сверточные нейронные сети с измененной схемой соединения между слоями: были созданы компактно соединенные (dense) блоки, которые соединяют другие слои между собой.

Более подробная архитектура сверточных нейронных сетей DenseNet представлена ниже, на рисунке 20.

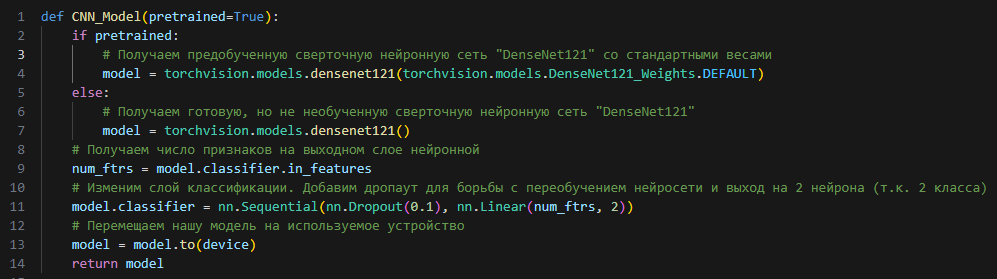


***Рисунок 20 – Архитектура нейронных сетей DenseNet***

Таким образом, в качестве самой оптимальной модели, по соотношению сложности модели – точность, была выбрана сверточная нейронная сеть DenseNet121.

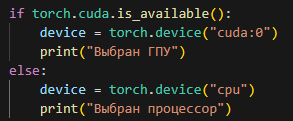
* 1. **Создание и обучение модели**

Для создания модели нейронной сети была создана функция «CNN\_Model», возвращающая модель DenseNet121 (с весами или без), с измененным блоком классификации, в который был добавлен слой Dropout (для борьбы с переобучением) и выходной слой на 2 нейрона, в соответствии с двумя классами (с признаками патологии и без признаков патологии). Затем, модель переносится на используемое устройство для обучения. Код функции «CNN\_Model» изображен на рисунке 21.



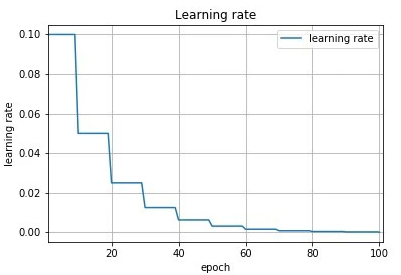
***Рисунок 21 – Код функции «CNN\_Model»***

Устройство, которое будет использовано при обучении, определяется кодом, который представлен на рисунке 22. В случае, если доступно обучение на CUDA-ядрах видеокарты, то вычисления модели будут осуществляться именно на ней. Иначе, в качестве доступного устройства будет задействован процессор.



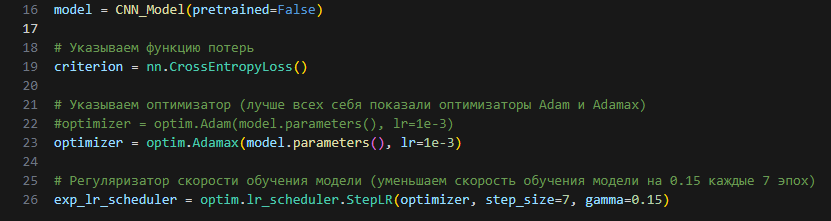
***Рисунок 22 – Код выбора доступного устройства для работы модели***

После создания модели, необходимо определить функцию потерь, оптимизатор и регуляризатор скорости обучения модели. В качестве функции потерь используется метрика Cross-entropy loss, подходящая для задач классификации. Среди оптимизаторов лучше всех себя показали «Adam» и «Adamax». В итоге всё же было отдано предпочтение оптимизатору «Adamax», т.к. с ним модель обучалась быстрее и имела более высокую точность. Регуляризатор скорости обучения модели – «StepLR», который уменьшает скорость обучения модели на 15% каждые 7 эпох (эти параметры также были подобраны экспериментально). Изменение скорости обучения происходит «ступенчато», пример работы такого регуляризатора показан ниже, на рисунке 23.



***Рисунок 23 – Визуализация работы регуляризатора StepLR***

Код, выполнение которого создает модель, функцию потерь, оптимизатор и регуляризатор скорости обучения, представлен ниже, на рисунке 24.



***Рисунок 24 – Код создания модели, функции потерь, оптимизатора и регуляризатора***

В созданной модели содержится 6.955.906 обучаемых параметров. Их веса будут меняться в ходе обучения модели.

Далее можно начинать обучение модели. Для этого была создана функция «train\_model» и списки, в которых будут сохраняться значения потерь и точности на каждой эпохе, для их дальнейшего анализа. Как уже было сказано ранее, в ходе обучения модели будет использоваться две фазы, обучающая, при которой веса модели будут изменяться, и валидационная, которая служит для оценки качества модели на каждой эпохе (в фазе валидации значения весов не корректируются).

Так как модель DenseNet стандартно работает с изображениями, которые используют цветовую модель RGB (то есть, изображения представляют из себя не один двумерный массив, а сразу 3, каждый из которых хранит информацию в соответствии с цветом), а изображения, извлекаемые из DICOM-формата, представлены в формате обычных двумерных массивов, их приходится преобразовывать, при помощи конкатинации тензоров, приводя к размеру (3, 180, 180).

Модель пробует сделать предсказания по данным, после чего эти предсказания сравниваются с правильными лейблами, высчитывается функция потерь. После этого, в фазе обучения, выполняется обратное распространение ошибки и инициализируется функция потерь. Веса модели корректируются.

Затем, обрабатываются данные о потерях и точности, происходит их вывод и сохранение в соответствующих списков, чтобы потом вывести графики.

Если на фазе валидации потери обновили свой минимум, происходит сохранение весов. Модель с именно этими весами и будет возвращена в конце обучения.

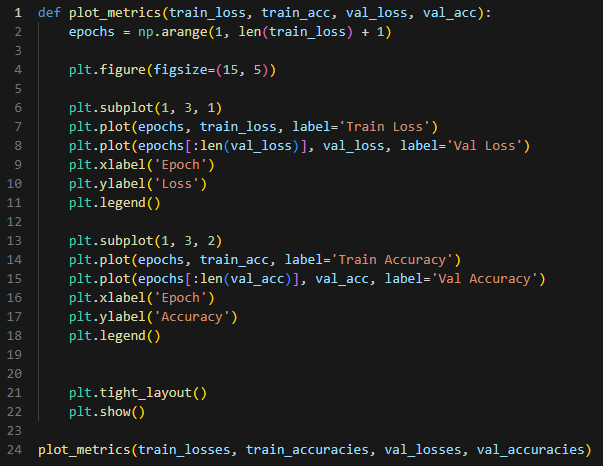
В конце будет выведена информация о времени, потребовавшемся на обучение и лучший Loss (самое низкое значение потерь на фазе валидации).

Основной код функции «train\_model» представлен ниже, на рисунке 25. Полный код функции находится в приложении. **УКАЗАТЬ В КАКОМ**



***Рисунок 25 – Код создания модели, функции потерь, оптимизатора и регуляризатора***

Вывод метрик осуществляется при помощи функции «plot\_metrics». Эта функция создает графики по данным точности и потерь в процессе обучения модели. Код функции «plot\_metrics» изображен ниже, на рисунке 26.



***Рисунок 26 – Код функции «plot\_metrcis»***

Обучение модели производилось более ста раз, с перебором параметров, от размера изображений и размера батчей, до смены оптимизаторов и функций потерь. Одним из самых удачным набором параметров стал: размер батчей 16 (небольшой размер батчей также способствовал уменьшению переобучения), оптимизатор «Adamax», размер изображений 180 на 180, размер Dropout’а 0.1, шаг оптимизатора 7, гамма оптимизатора 0.15, стандартные веса предобученной модели.

На этих параметрах модель была обучена 201 эпоху. Обучение длилось несколько часов. По результатам обучения были выведены графики, представленные ниже, на рисунке 27.

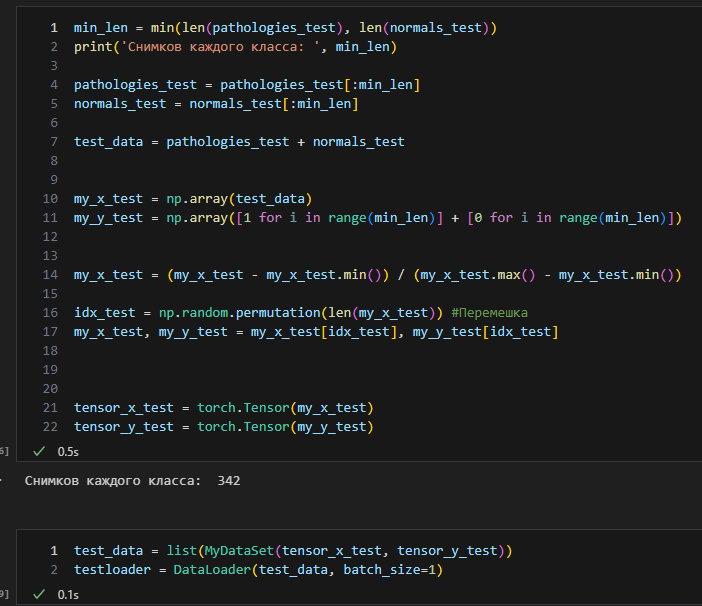


***Рисунок 27 – Графики по результатам обучения модели на 201 эпохе***

Таким образом, после множества экспериментов с моделью, были подобраны оптимальные параметры, на основе которых произвелось обучение на 201 эпохе. Данная модель была сохранена. Тестирование модели будет описано далее.

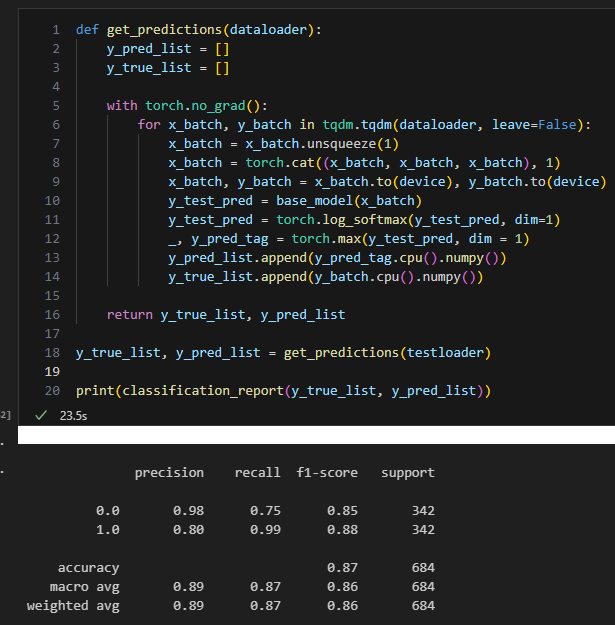
* 1. **Тестирование модели**

Тестирование проводилось по первым двум записям, которые не использовались при обучении и их изображения были сохранены в списках «pathologies\_test» и «normals\_test». Эти данные были обработаны, нормализованы и преобразованы в тензоры. Затем к ним был создан загрузчик данных, который использовался при тестировании. Код обработки тестовых КТ снимков и создание загрузчика данных к ним представлен ниже, на рисунке 28.



***Рисунок 28 – Обработка тестовых КТ снимков и создание загрузчика данных к ним***

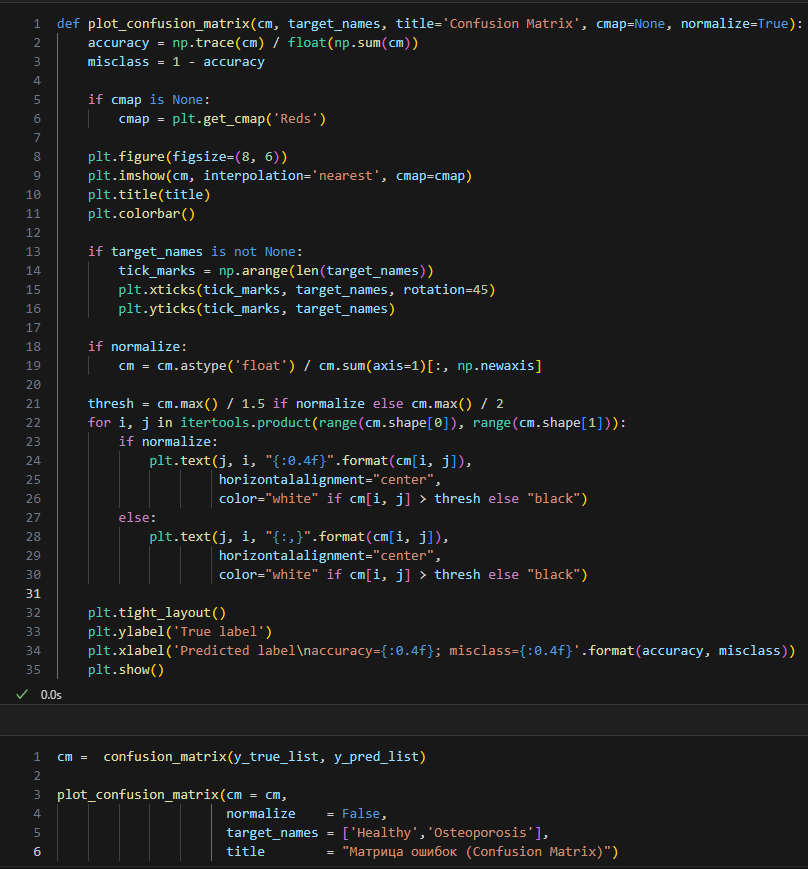
Тестовые изображения были переданы модели для обработки в функцию «get\_predictions», которая возвращает список лейблов, предсказанных моделью список истинных лейблов. После чего результаты, представленные моделью, были сравнены с истинными лейблами, с выводом соответствующих метрик при помощи функции «classification\_report» из библиотеки «sklearn». Код тестирования с выводом метрик и, непосредственно, сами значения метрик, изображены ниже, на рисунке 29.



***Рисунок 29 – Код функции «get\_predictions» и результаты тестирования с выводом метрик***

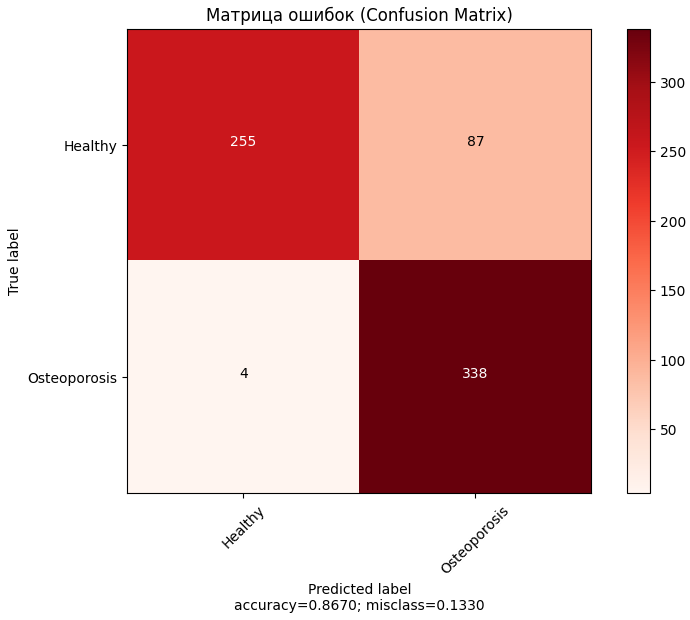
Исходя из метрик, модель имеет не только достаточно высокую общую точность (87%), но, также, большой показатель Recall, что особенно важно в задачах диагностики заболевания. Несмотря на то, что модель иногда ошибается, обозначая снимки без патологии как снимки с патологией, намного важнее то, что она редко делает обратные ошибки (отмечает снимки с патологией, как снимки без патологии).

Также, для тестирования была рассмотрена матрица ошибок. Для получения данных о матрице ошибок, была использована функция «confusion\_matrix» из библиотеки «sklearn». Также, была создана функция «plot\_confusion\_matrix», для построения графика матрицы ошибок. Код функции «plot\_confusion\_matrix» и код создания матрицы ошибок представлены ниже, на рисунке 30.



***Рисунок 30 – Код функции «plot\_confusion\_matrix» и код создания матрицы ошибок***

Матрица ошибок, построенная по результатам тестирования, изображена ниже, на рисунке 31.



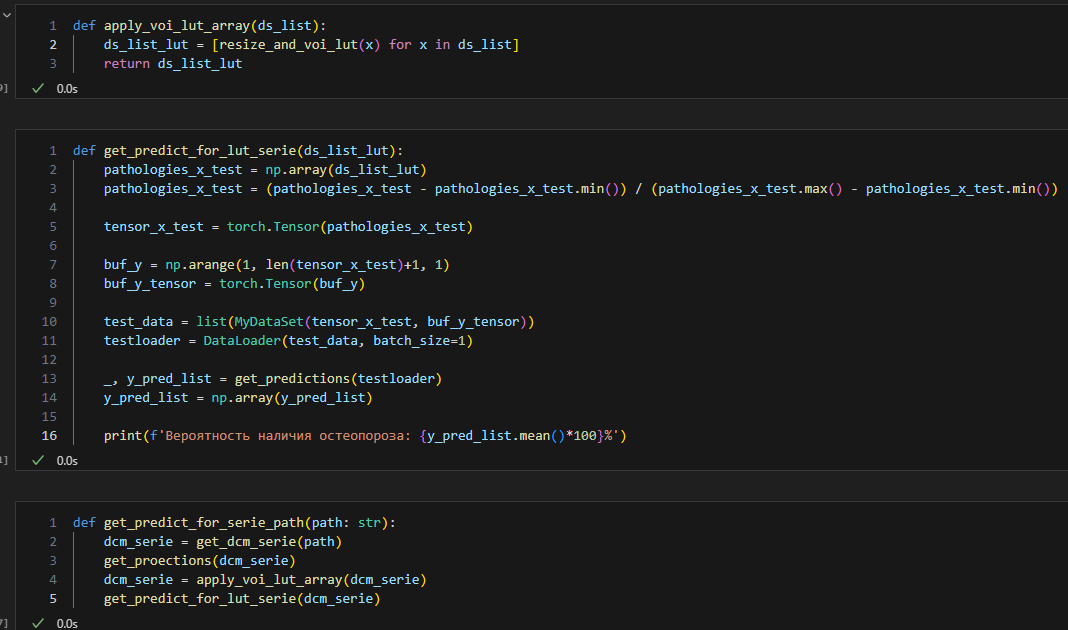
***Рисунок 31 – Матрица ошибок по тестовым данным***

По данным из графика можно еще раз убедиться в том, что модель достаточно точно определяет наличие остеопороза, но, периодически совершает ошибки, отмечая наличие патологии на снимках, где признаки патологии отсутствуют.

Таким образом, была получена достаточно точная модель для диагностики остеопороза по КТ снимкам позвоночника, которая была протестирована по различным метрикам.

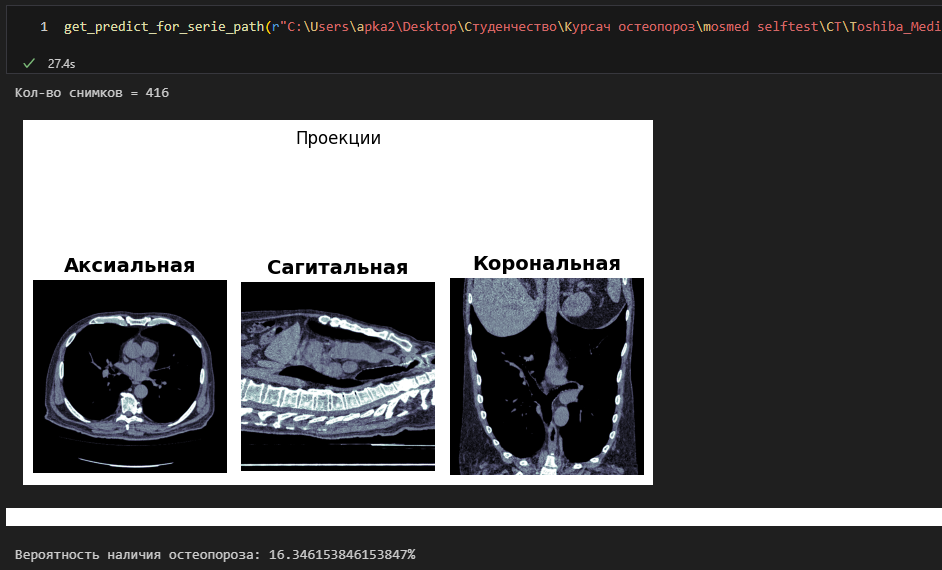
* 1. **Проверка совместимости системы с записями для селф-тестирования**

Систему необходимо также протестировать на совместимость с записями из набора данных, предназначенного для селф-тестирования от ГБУЗ «НПКЦ ДиТ ДЗМ». Для этого были созданы функции для получения данных по их адресу, их предобработке, созданию тензора и загрузчика данных: функции «get\_dcm\_serie», «apply\_voi\_lut\_array» и «get\_predict\_for\_lut\_serie» соответственно. Затем все они были объединены в функции «get\_predict\_for\_serie\_path», которая получает данные по адресу, строит по ним проекции, выполняет предобработку изображений и выводит вероятность наличия остеопороза. Код этих функций представлен ниже, на рисунке 32.

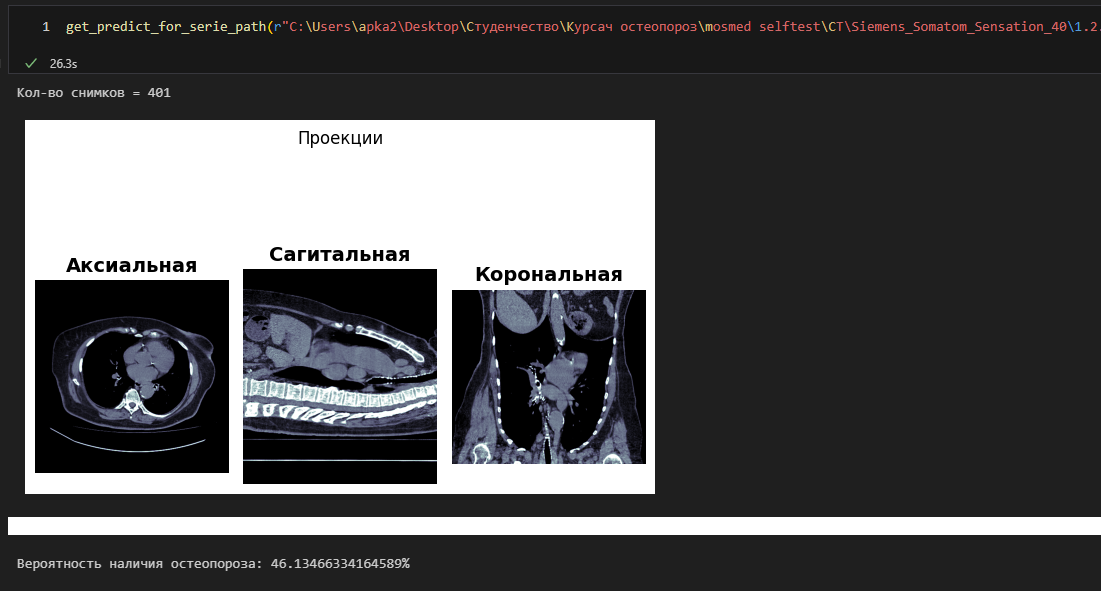


***Рисунок 32 – Код функций «apply\_voi\_lut\_array», «get\_predict\_for\_lut\_serie» и «get\_predict\_for\_serie\_path»***

Функционирование системы было проверено на записях из набора данных, предназначенного для проведения селф-тестирования. Система успешно смогла обработать данные, вывести проекции снимков и вероятность наличия остеопороза в процентах. Примеры работы системы с этими записями приведены ниже, на рисунках 33 и 34.



***Рисунок 33 – Первый пример работы системы с записью из набора данных для селф-тестирования***



***Рисунок 34 – Второй пример работы системы с записью из набора данных для селф-тестирования***

Таким образом, модель является совместимой с записями из набор данных для селф-тестирования от ГБУЗ «НПКЦ ДиТ ДЗМ», и потенциально, может использоваться в их информационной среде.

1. **Заключение**

В процессе выполнения учебной практики была поставлена цель разработать веб-приложение, предназначенное для автоматизации деятельности станции скорой помощи для животных. Для достижения этой цели были решены следующие задачи:

1. выполнен анализ предметной области, в том числе проанализированы статистические данные по безнадзорным животным в России, описана деятельность станции скорой помощи для животных, осуществлен обзор и сравнение существующих аналогов, а также описаны роли предметной области и сценарии использования информационной системы;
2. выполнено проектирование базы данных, в том числе разобраны объекты и атрибуты предметной области, база данных приведена к третьей нормальной форме, составлена ER-диаграмма нормализованной базы данных, выбрана СУБД, разработана физическая модель данных и перенесена в технологию Entity Framework Core;
3. осуществлена разработка веб-приложения на платформе ASP.NET CORE MVC, составлены диаграммы активностей, спроектирован пользовательский интерфейс, обеспечен высокоуровневый доступ к данным и выполнены итоговое тестирования и отладка приложения.

Таким образом, цель была успешно достигнута. Итоговое веб-приложение полноценно функционирует и вполне пригодно для использования в реальной станции скорой помощи для животных, существенно облегчая процесс учета животных, взаимодействия с работниками и волонтерами, а также делая процесс благотворительных сборов проще. Тем не менее, остается простор для совершенствования приложения, от переноса базы данных с локального сервера на удаленный, до добавления совершенно нового функционала (работа с медикаментами, составление рабочих планов и т.д.).

Во время выполнения учебной практики были развиты навыки проектирования, реализации, тестирования и отладки программных систем, а также приобретен опыт работы с СУБД и закреплены навыки разработки документации и оформления отчетов о проделанной работе.

1. **Список используемой литературы**
2. Вконтакте, ССП Zoohelp Антидогхантер Пермь. [Электронный ресурс] // URL: <https://vk.com/zoohelp_perm> Дата обращения: 10.05.2023
3. Ассоциация благополучие животных, Статистика мониторинга численности безнадзорных животных. [Электронный ресурс] // URL: <https://blagozoo.ru/statistika> Дата обращения: 10.05.2023
4. Tass, В России проживает более 4 млн бездомных животных. [Электронный ресурс] // URL: <https://tass.ru/obschestvo/12947457> Дата обращения: 10.05.2023
5. Zoom Market, Опрос и исследование россиян о защите животных от жестокого обращения. [Электронный ресурс] // URL: <https://www.mazm.ru/article/a-2064.php> Дата обращения: 10.05.2023
6. Ветеринарная клиника ЦЕНТР, Догхантеры – кто они такие?. [Электронный ресурс] // URL: [https://vetcentr.ru/publications/doghantery.html](https://vetcentr.ru/publications/doghantery.html%20) Дата обращения: 10.05.2023
7. Добро.ru, Фонд помощи животным «Станция скорой помощи». [Электронный ресурс] // URL: <https://dobro.ru/organizations/10016553/info> Дата обращения: 10.05.2023
8. ACITS, Acits. [Электронный ресурс] // URL: <https://acits.ru/> Дата обращения: 11.05.2023
9. Vc.ru, Благотворительный стартап Acits. [Электронный ресурс] // URL: <https://vc.ru/u/292812-tasha-sokolova/185101-blagotvoritelnyy-startap-acits-razrabotal-besplatnoe-prilozhenie-dlya-priyutov-i-reabilitacionnyh-centrov-dlya-zhivotnyh> Дата обращения: 13.05.2023
10. OMG.ru, OMG UML. [Электронный ресурс] // URL: <https://www.omg.org/spec/UML/2.5.1/PDF> Дата обращения: 15.05.2023
11. Overcode, Сравнение баз данных. [Электронный ресурс] // URL: <https://overcoder.net/manuals/postgresql-mariadb-i-sqlite-sravnenie-baz-dannyh> Дата обращения: 15.05.2023
12. Tproger, Сравниваем популярные реляционные СУБД. [Электронный ресурс] // URL: <https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/> Дата обращения: 15.05.2023
13. Apache Friends, XAMPP Installers. [Электронный ресурс] // URL: <https://xampp.ru/> Дата обращения: 16.05.2023
14. Microsoft Learn, Общие сведения ASP.NET Core MVC [Электронный ресурс] // URL: <https://learn.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-7.0> Дата обращения: 18.05.2023
15. Uprock, Принцип сходства в визуальном дизайне [Электронный ресурс] // URL: <https://www.uprock.ru/articles/princip-shodstva-v-vizualnom-dizayne> Дата обращения: 20.05.2023

ПРИЛОЖЕНИЕ А  
**Техническое задание**

**Пермский филиал федерального государственного автономного**

**образовательного учреждения высшего образования**

**«Национальный исследовательский университет**

**«Высшая школа экономики»**

*Факультет социально-экономических и компьютерных наук*

Кирьянов Сергей Вячеславович

**Информационная система станции скорой помощи для животных**

*Техническое задание*

Заказчик

Доцент  
кафедры информационных   
технологий в бизнесе  
НИУ ВШЭ-Пермь

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
М.А. Плаксин

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Руководитель

Доцент   
кафедры информационных   
технологий в бизнесе  
НИУ ВШЭ-Пермь

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
М.А. Плаксин

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**Пермь 2023**

**1 Цель проекта**

Целью реализации проекта является разработка информационной системы станции скорой помощи для животных, представляющей собой веб-приложение к реляционной базе данных, позволяющее просматривать, добавлять, модифицировать и удалять данные, создавать отчеты, а также экспортировать их в заранее предопределенных форматах.

**2 Информация о заказчике**

Заказчиком проектных работ является кафедра информационных технологий в бизнесе НИУ ВШЭ Пермь в лице доцента М.А. Плаксина.

**3 Сроки реализации**

Срок прохождения практики: с 10 мая 2023 г. по 23 мая 2023 г.

**4 Функциональные требования**

Разрабатываемая информационная система должна давать пользователю следующие возможности:

* регистрировать новых пользователей;
* сохранять данные о зарегистрированных пользователях;
* синхронизировать работу пользователя с базой данных;
* выводить данные из базы данных;
* добавлять новые записи в базу данных;
* редактировать записи в базе данных;
* удалить записи из базы данных;
* представлять данные в удобном формате;
* экспортировать данные в формате csv.

**5 Требования к исполнителю**

Исполнителем является студент второго курса бакалавриата направления подготовки 09.03.04 «Программная инженерия», Кирьянов Сергей Вячеславович.

**6 Этапы и график реализации проекта**

Длительность проекта определяется итерационным характером модели жизненного цикла продукта и линейным характером жизненного цикла проекта.

С точки зрения управления проектом выделяются стадии:

* инициация;
* планирование;
* исполнение;
* завершение.

С точки зрения управления жизненным циклом продукта проект состоит из трех итераций, каждая из которых состоит из пяти этапов:

1. Инициализация.
2. Проектирование.
3. Реализация.
4. Тестирование.
5. Развертывание.

Более подробно входящие в этапы проекта задачи, их продолжительность, исполнители и ожидаемые результаты описаны ниже, в таблице 1.

Таблица 1 – Этапы проекта

| **Номер ИСР** | **Описание задачи** | **Сроки выполнения** | **Результат** |
| --- | --- | --- | --- |
| 1 | **Анализ и планирование проекта** | | |
| 1.1 | Анализ предметной области | 10.05.2023 |  |
| 1.2 | Изучение теории по разработке приложений на платформе ASP.NET CORE MVC | 10.05.2023-11.05.2023 |  |
| 1.3 | Анализ существующих решений по смежным областям | 11.05.2023 |  |
| 1.4 | Анализ ролей предметной области | 11.05.2023-12.05.2023 | Описание ролей предметной области |
| 1.5 | Составление графика работ | 12.05.2023 | График работ |
| 2 | **Проектирование и создание прототипа продукта** | | |
| 2.1 | Сбор данных и требований к приложению | 13.05.2023-14.05.2023 | Техническое задание |
| 2.2 | Проектирование базы данных | 14.05.2023-15.05.2023 | Прототип базы данных с объектами и атрибутами |
| 2.3 | Нормализация базы данных | 15.05.2023 | База данных в третьей нормальной форме |
| 2.4 | Выбор СУБД | 16.05.2023 | Выбор СУБД |
| 2.5 | Разработка физической модели денных | 16.05.2023-17.05.2023 | База данных со связями |
| 2.6 | Перенос базы данных в технологию Entity Framework Core | 17.05.2023-18.05.2023 | Физ. модель базы данных в технологии ORM |
| 3 | **Реализация и тестирование** | | |
| 3.1 | Описание диаграмм активностей | 18.05.2023 | Диаграммы активностей по прецедентам |
| 3.2 | Создание пользовательского интерфейса | 18.05.2023 –  19.05.2023 | Рабочий интерфейс приложения |
| 3.3 | Разработка приложения | 19.05.2023 - 20.05.2023 | Рабочий прототип приложения |
| 3.4 | Тестирование приложения | 21.05.2023-22.05.2023 | Задокументированные результаты тестирования |
| 3.5 | Написание документации | 22.05.2023-23.05.2023 | Отчет, руководство пользователя |

Проектный цикл начинается с 1 ноября 2022 года и заканчивается 27 марта 2023 года.

**7 Критерии завершенности проекта**

Критерием завершенности проекта являются:

1. Полное выполнение всех этапов разработки согласно графику по итерациям:
   1. Планирование.
   2. Разработка.
   3. Тестирование.
   4. Внедрение.
2. Составление документации.
3. Наличие информационной системы станции скорой помощи для животных, работающей с развернутой заблаговременно базой данных.
4. Проведение демонстрации работы программы.
5. Наличие полного комплекта сопроводительной документации.

Цели проекта считаются выполненными и проект принимается заказчиком, если:

1. Система запускается и исправно работает на ПК пользователя.
2. Система запускает выполнение команды: управляет исполнительными механизмами, исправно взаимодействует с базой данных.

**8 Критерии качества продукта**

Критерии качества продукта отражены в таблице 2.

Таблица 2 – Критерии качества продукта

| **№** | **Критерий** | **Подтверждение** | **Баллы** |
| --- | --- | --- | --- |
| 1 | Наличие работоспособной информационной системы станции скорой помощи для животных | Сценарии тестирования | 3 |
| 2 | Программа развернута на оборудовании заказчика | Протокол развертывания | 3 |
| 3 | Наличие документации | Реестр требований | 2 |
| План приемо-сдаточных испытаний системы |
| План тестирования |
| График работ |
| Техническое задание |
| Руководство пользователя |
| Документ о принятии работ |
| 5 | Отчет о завершении практики | Отчет | 2 |

**9 Форма отчетности**

К защите практики предоставить следующий комплект программной документации:

* техническое задание;
* отчет по учебной практике;
* руководство пользователя;
* заявка-предложение;
* реестр требований;
* сценарии использования;
* диаграммы использования;
* отзыв руководителя.

**10 Трудоемкость проекта**

Итоговая трудоемкость проекта составляет 3 з.е.

ПРИЛОЖЕНИЕ Б  
**Руководство пользователя**

|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
| Кирьянов Сергей Вячеславович  **Информационная система станции скорой помощи для животных**  *Руководство пользователя*  студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель  Доцент кафедры информационных технологий в бизнесе НИУ ВШЭ-Пермь    М.А. Плаксин |

Пермь, 2023 год

1. **Введение**
2. **Область применения**

Приложение применяется при:

1. Предварительных комплексных испытаниях.
2. Приемочных испытаниях.
3. Свободная эксплуатация.

Основные сферы использования приложения можно охарактеризовать, как профессиональную и волонтерскую.

1. **Краткое описание возможностей**

Веб-приложение предназначено для осуществления деятельности станции скорой помощи для животных.

Приложение позволяет создавать электронные таблицы с данными, просматривать и редактировать их. Также, есть возможность экспортировать данные из других программ для работы с электронными таблицами.

Эксплуатация приложения возможна на одном языке: русском.

1. **Уровень подготовки пользователя**

Для успешной эксплуатации приложения пользователь должен ознакомиться с эксплуатационной документацией и не обязан обладать какими-либо специализированными знаниями и/или умениями.

1. **Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю**

Пользователю необходимо ознакомиться с руководством пользователя.

1. **Назначение и условия применения**

Вею-приложение предназначено для осуществления деятельности станции скорой помощи для животных.

Приложение позволяет создавать электронные таблицы с данными, просматривать и редактировать их. Также, есть возможность экспортировать и импортировать данные из других программ для работы с электронными таблицами.

Для успешного функционирования веб-приложения необходимо постоянное, стабильное интернет-соединение.

1. **Подготовка к работе**
   1. **Состав и содержание дистрибутивного носителя данных**

Для использования функций импорта и экспорта данных информационной системы станции скорой помощи для животных требуется наличие такого предустановленного программного обеспечения, как Microsoft Excel и любой браузер из перечня: Google Chrome, Microsoft Edge, Opera.

* 1. **Порядок загрузки приложения**

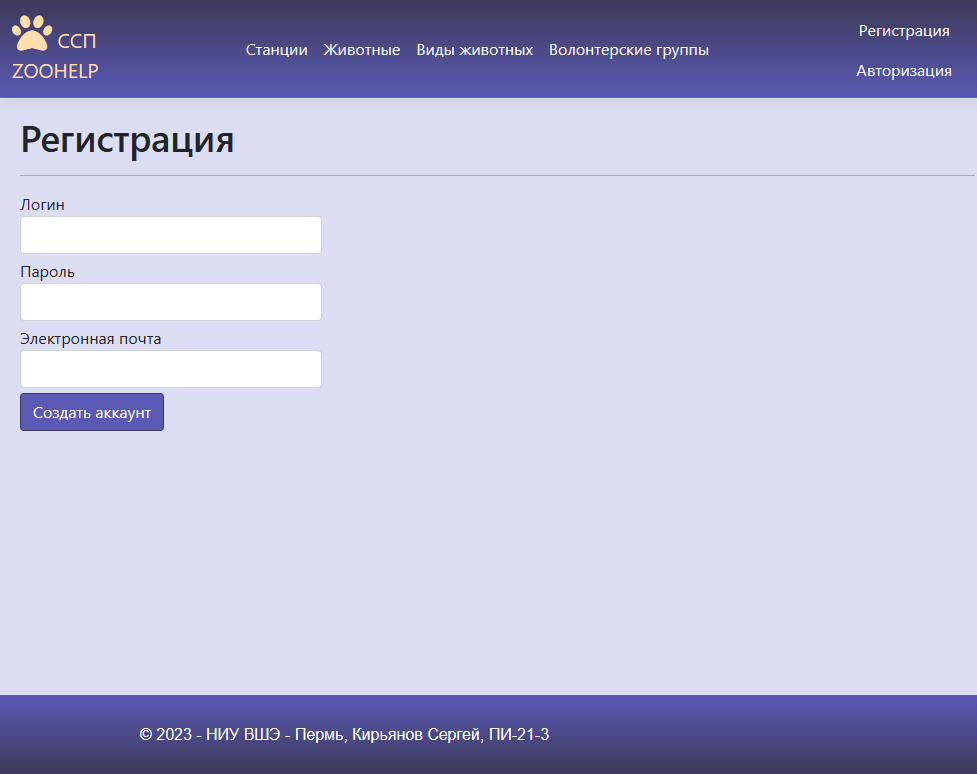
Перед началом работы с приложением на рабочем месте пользователя необходимо выполнить следующие действия:

1. Подключиться к базе данных.
2. Открыть в браузере страницу веб-приложения.
   1. **Порядок проверки работоспособности**

Для проверки работоспособности информационной системы станции скорой помощи для животных необходимо выполнить следующие действия:

1. Открыть страницу веб-приложения, перейдя по его адресу.
2. Дождаться загрузки домашней страницы веб-приложения..
3. Попытаться зарегистрироваться или авторизоваться в системе.
4. **Описание окон приложения**
   1. **Окно регистрации**

Окно регистрации изображено ниже, на рисунке 1.



***Рисунок 1 – Окно регистрации***

Поле ввода «Логин» используется для ввода логина (уникального идентификатора) пользователя.

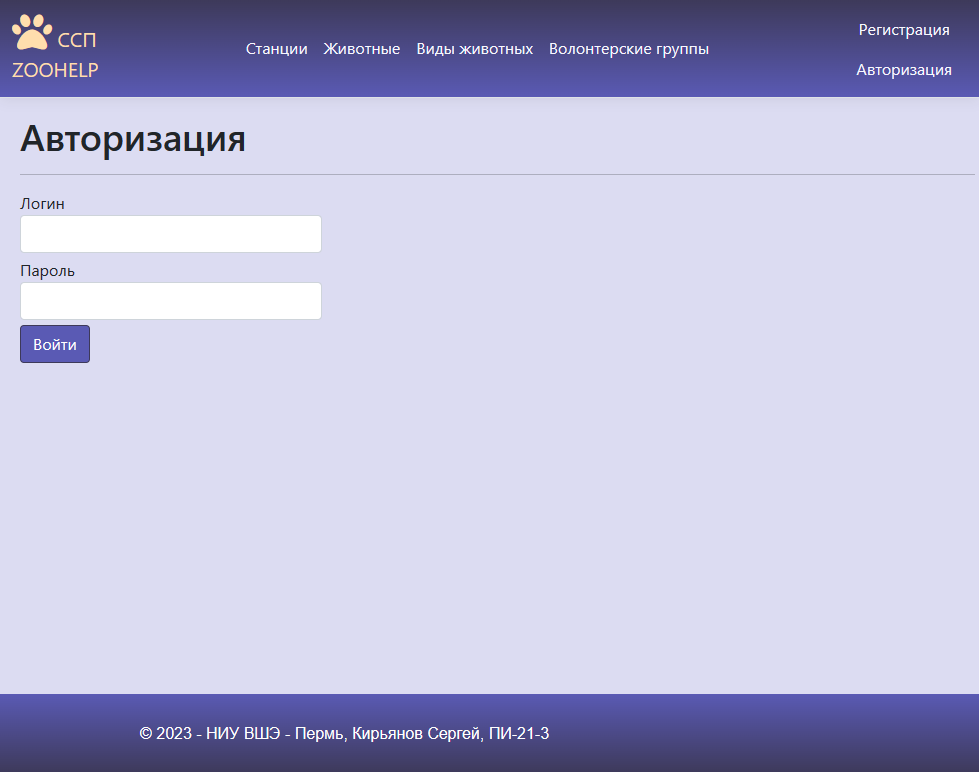
Поле ввода «Пароль» используется для ввода пароля пользователя.

Поле ввода «Электронная почта» используется для ввода адреса электронной почты пользователя.

Кнопка «Создать аккаунт» инициирует попытку регистрации пользователя. В случае возникновения ошибок, о них будет сообщено пользователю. В случае успешной регистрации, пользователю будет предложено авторизоваться в системе для ее дальнейшего использования.

* 1. **Окно авторизации**

Окно авторизации изображено ниже, на рисунке 2.



***Рисунок 2 – Окно авторизации***

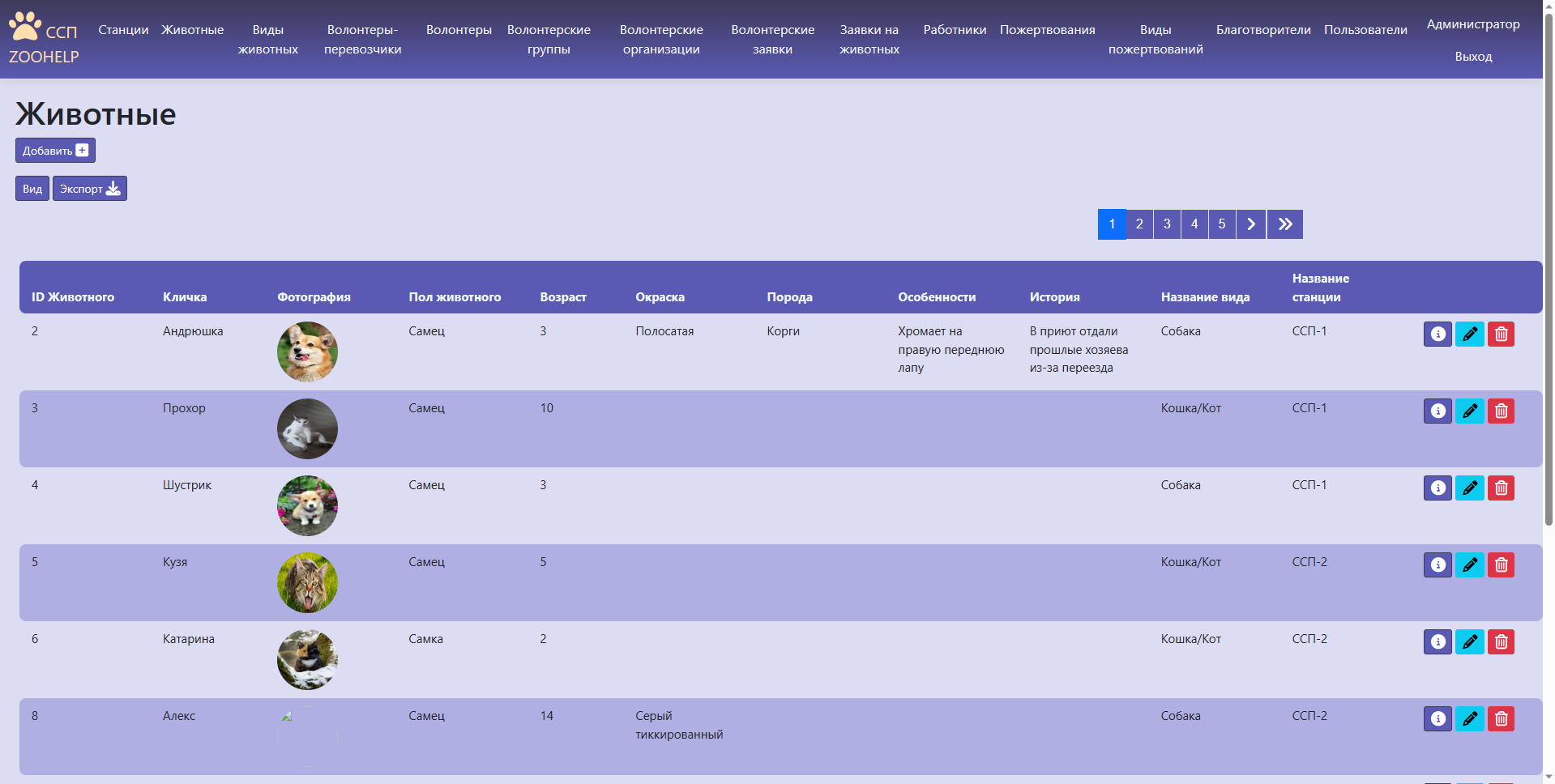
Поле ввода «Логин» используется для ввода логина (уникального идентификатора) пользователя.

Поле ввода «Пароль» используется для ввода пароля пользователя.

Кнопка «Войти» инициирует попытку входа в учетную запись пользователя. В случае возникновения ошибок, о них будет сообщено пользователю. В случае успешной авторизации, откроется домашняя страница веб-приложения.

* 1. **Окно просмотра информации**

Окно просмотра информации изображено ниже, на рисунке 3.



***Рисунок 3 – Окно просмотра информации***

В верхней части экрана (в навигационной панели) находятся ссылки на все доступные таблицы.

В центре экрана находится таблица со всеми данными.

Кнопка «Подробности» откроет окно подробностей выбранной записи.

Кнопка «Добавить» откроет окно добавления новой записи.

Кнопка «Удалить» откроет окно подтверждения удаления выбранной записи.

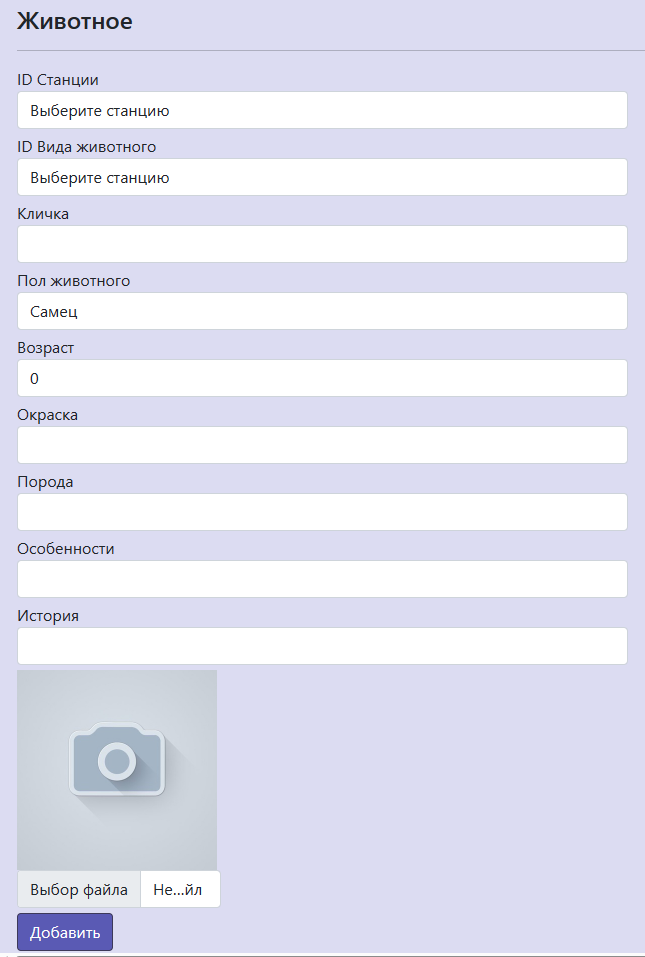
Кнопка «Изменить» откроет окно изменения выбранной записи.

Кнопка «Экспорт» экспортирует данные в формате .csv.

Кнопка «Вид» откроет окно альтернативного просмотра информации (применимо только к таблице с животными).

* 1. **Окно добавления записи**

Окно добавления новой записи изображено ниже, на рисунке 4.



***Рисунок 4 – Окно добавления записи***

В верхней части окна расположено название таблицы.

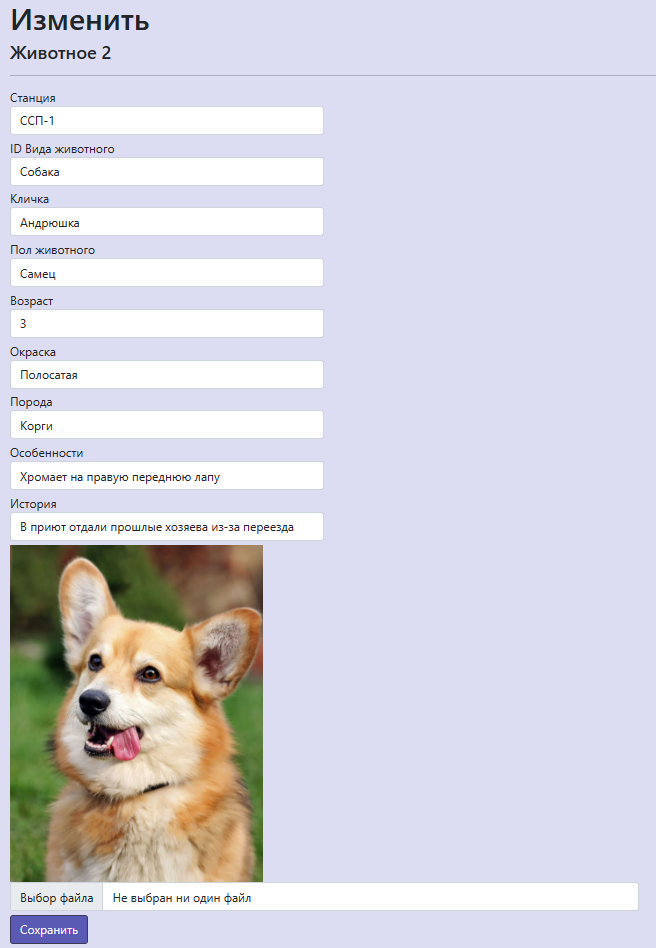
В центральной части окна расположены поля ввода (поля текстового ввода и выпадающие списки) для ввода информации о новой записи. Над ними находятся названия полей.

Кнопка «Выберите файл» появляется только в том случае, когда таблица может принять на вход изображение.

Кнопка «Добавить» инициирует попытку добавления новой записи в таблицу, согласно введенным данным. При возникновении ошибки, новая запись не будет добавлена. Пользователь получит соответствующее сообщение.

* 1. **Окно изменения записи**

Окно изменения выбранной записи изображено ниже, на рисунке 5.



***Рисунок 5 – Окно изменения записи***

В верхней части окна расположено название таблицы и название объекта с идентификационным номером.

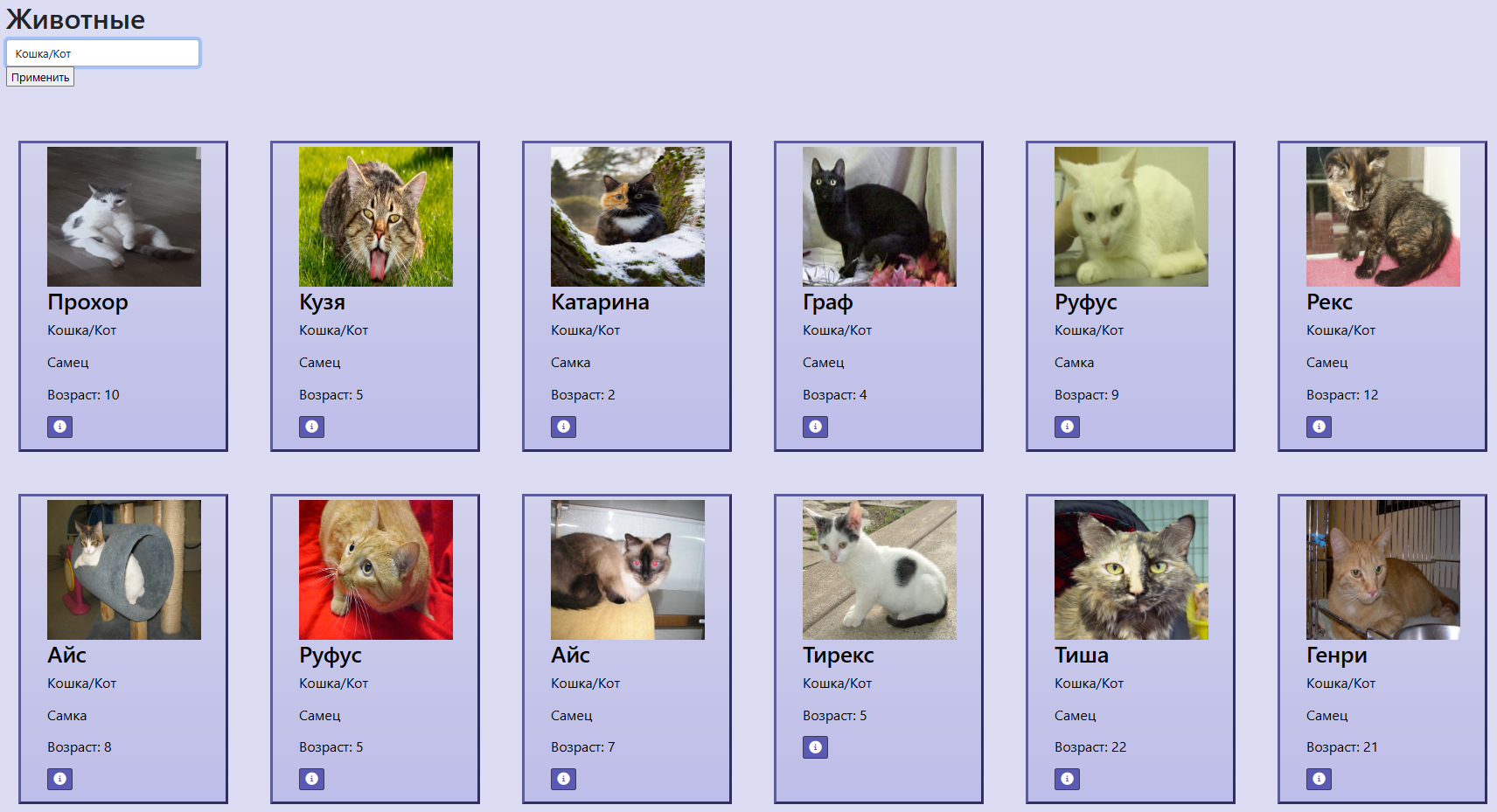
В средней части экрана расположены поля ввода (поля текстового ввода и выпадающие списки) для ввода информации о новой записи. Над ними находятся названия полей. Изначально в этих полях будут текущие данные, заданные в таблице.

Кнопка «Выберите файл» появляется только в том случае, когда таблица может принять на вход изображение.

Кнопка «Сохранить» инициирует попытку изменения данных о записи, согласно введенным данным. При возникновении ошибки, запись не будет изменена. Пользователь получит соответствующее сообщение. При успешном изменении, пользователь также получит сообщение и данное окно будет закрыто.

* 1. **Окно альтернативного просмотра**

Окно альтернативного просмотра изображено ниже, на рисунке 6.



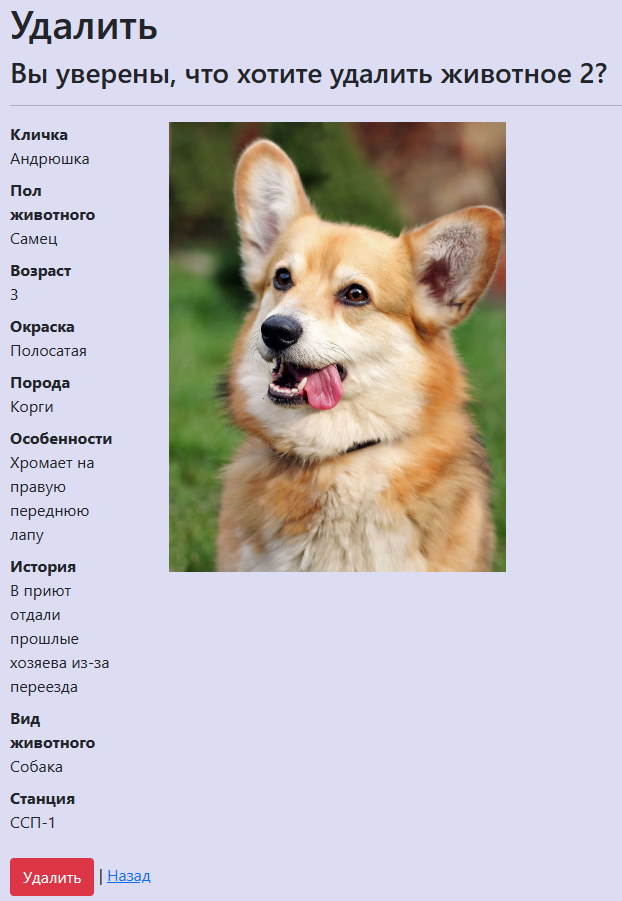
***Рисунок 6 – Окно альтернативного просмотра***

В верхней части экрана находится выпадающий список для выбора вида животных, по которому будет произведена группировка.

В центральной части экрана расположены данные о записях таблицы «Животные».

* 1. **Окно подтверждения удаления**

Окно подтверждения удаления изображено ниже, на рисунке 7.



***Рисунок 7 – Окно подтверждения удаления***

В середине окна выведена сводная информация о записи, которая была выбрана для удаления.

Кнопка «Назад» закроет это окно и запись не будет удалена.

Кнопка «Удалить» удалит выбранную запись и закроет это окно.

1. **Аварийные ситуации**

В случае возникновения ошибок при использовании приложения, необходимо перезагрузить страницу браузера.

В случае, если страница веб-приложения не закрывается, необходимо закрыть браузер.

1. **Рекомендации по освоению**

При работе с приложением рекомендуется отключить дополнительные расширения браузера.

Особых рекомендаций по освоению и эксплуатации приложения нет.

ПРИЛОЖЕНИЕ В  
**Код модели Animal**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Drawing;

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json.Serialization;

#nullable disable

namespace WebApplication2.DataObjects

{

[Table("animals")]

[Index(nameof(AnimalTypeId), Name = "ID Животного")]

[Index(nameof(StationId), Name = "ID Станции")]

[Display(Name = "Животные")]

public partial class Animal

{

public Animal()

{

AnimalApplications = new HashSet<AnimalApplication>();

}

private byte[] photoArr = null;

[Key]

[Column("animal\_id", TypeName = "int(11)")]

[Display(Name = "ID Животного")]

public int AnimalId { get; set; }

[Column("station\_id", TypeName = "int(11)")]

[Display(Name = "ID Станции")]

public int StationId { get; set; }

[Column("animal\_type\_id", TypeName = "int(11)")]

[Display(Name = "ID Вида животного")]

public int AnimalTypeId { get; set; }

[Required(ErrorMessage = "Необходимо ввести корректную кличку!")]

[Column("name")]

[Display(Name = "Кличка")]

[StringLength(64)]

public string Name { get; set; }

[Column("photo", TypeName = "mediumblob")]

[Display(Name = "Фотография")]

public byte[] Photo { get; set; }

[Required(ErrorMessage = "Необходимо добавить корректную фотографию!")]

[Column("animal\_sex")]

[Display(Name = "Пол животного")]

[StringLength(5)]

public string AnimalSex { get; set; }

[Display(Name = "Возраст")]

[Column("age", TypeName = "tinyint(4)")]

public byte Age { get; set; }

[Column("coloring")]

[Display(Name = "Окраска")]

[StringLength(80)]

public string Coloring { get; set; }

[Column("breed")]

[Display(Name = "Порода")]

[StringLength(40)]

public string Breed { get; set; }

[Column("specificity")]

[Display(Name = "Особенности")]

[StringLength(200)]

public string Specificity { get; set; }

[Column("history")]

[Display(Name = "История")]

[StringLength(500)]

public string History { get; set; }

//I FORM FILE PHOTOTOT

[Display(Name = "PhotoHelper")]

[NotMapped]

public IFormFile PhotoHelper { get; set; }

[ForeignKey(nameof(AnimalTypeId))]

[InverseProperty("Animals")]

public virtual AnimalType AnimalType { get; set; }

[ForeignKey(nameof(StationId))]

[InverseProperty("Animals")]

public virtual Station Station { get; set; }

[InverseProperty(nameof(AnimalApplication.Animal))]

public virtual ICollection<AnimalApplication> AnimalApplications { get; set; }

}

}

ПРИЛОЖЕНИЕ С  
**Остальной код приложения**

Остальной код приложения расположен в репозитории на веб-сервисе для хостинга IT-проектов GitHub. Сделано это для того, чтобы не перегружать отчет излишним количеством аналогичного кода представлений, моделей и контроллеров.

Ссылка на репозиторий: <https://github.com/Laoreet/PracticeWorkCourse2/>