

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

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Сервис хранения, поиска и визуализации наборов
открытых данных**

Студент

Будн 14.06.21
(подпись, дата)

Будников Данил Сергеевич
(ФИО полностью)

Руководитель

Балаш 14.06.21
(подпись, дата)

Балашова И.Ю.
(фамилия, инициалы)

Нормоконтролер

ТД 22.06.21
(подпись, дата)

Такташкин Д.В
(фамилия, инициалы)

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

Заведующий кафедрой

Макарычев
(подпись)

Макарычев П.П.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от 15.06.21 № 16)

Секретарь ГЭК

Григор
(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2021

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

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

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Будников Данил Сергеевич гр. 17ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Сервис хранения, поиска и визуализации наборов открытых данных

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

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) Назначение: сервис для свободного распространения наборов открытых данных с
возможностью поиска и визуализации табличных данных.

2) Среда выполнения: операционная система Ubuntu 20.04, Web-браузер

Языковые средства разработки: Python, JavaScript, UML.

Инструментальные средства: VS Code, PostgreSQL 12, DBeaver, npm, Draw.io,
Технологии разработки ООП, сервисно-ориентированный подход.

3) Основные функции приложения:

а) регистрация и авторизация;

б) просмотр списка открытых наборов данных;

в) предварительный просмотр набора данных;

г) загрузка собственного набора данных;

д) скачивание набора данных;

е) оставление оценки;

ж) администрирование пользователей;

з) визуализация табличных данных;

. Перечень подлежащих разработке вопросов:

- 1) Анализ предметной области и постановка задачи;
- 2) Проектирование;
- 3) Реализация;
- 4) Контроль качества программного обеспечения;
- 5) План разработки бюджета;
- 6) Оформление пояснительной записки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Под руковод
1	Анализ предметной области и постановка задачи	10%	14.04.2021 – 27.04.2021	
2	Проектирование	25%	27.04.2021 – 10.05.2021	
3	Реализация	35%	10.05.2021 – 30.06.2021	
4	Контроль качества программного обеспечения	10%	31.05.2021 – 02.06.2021	
5	План разработки бюджета	5%	02.06.2021 – 03.06.2021	
6	Оформление пояснительной записки	15%	03.06.2021 – 14.06.2021	

Дата выдачи задания «28» _____ декабря _____ 2020 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета

(подпись, дата)

Реферат

Пояснительная записка содержит 102 листа, 47 рисунков, 12 таблиц, 23 использованный источник и 3 приложения.

ОТКРЫТЫЕ ДАННЫЕ, ВИЗУАЛИЗАЦИЯ, УМЕНЬШЕНИЕ РАЗМЕРНОСТИ, КЛИЕНТ, СЕРВЕР ПРИЛОЖЕНИЙ, СЕРВЕР БАЗЫ ДАННЫХ, UML, REST, MVC.

Объект исследования – сервисы свободного распространения наборов данных и основные методы визуализации данных.

Цель выпускной квалификационной работы – разработать сервис с возможностью свободного распространения, поиска и визуализации наборов открытых данных.

Технологии разработки – разработка сервера приложений проводилась на языке программирования Python с использованием фреймворка Flask и ORM SQLAlchemy, клиентская часть разрабатывалась на языке JavaScript с использованием фреймворка React.js, для создания сервера БД использовался PostgreSQL 12. В качестве среды разработки использовался редактор программного кода VS Code и менеджер баз данных DBeaver.

Среда исполнения – Linux, Web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть сервиса хранения, поиска и визуализации наборов открытых данных. Все требования к разработке приложения соблюдены.

Лист				ПГУ 09.03.04– 8ВР171.07 ПЗ			
Лист	№ докум.	Подпись	Дата	Сервис хранения, поиска и визуализации наборов открытых данных. Пояснительная записка	Лит.	Лист	Листов
5.	Будников Д.С.		14.06.21			3	102
	Балашова И.Ю.		14.06.21				
втр.	Такташкин Д.В.		22.06.21		Группа 17ВП1		
	Максимов П.А.						

Содержание

Введение.....	6
1 Анализ сервисов открытых данных	7
1.1 Понятие открытых данных и стандарты их публикации.....	7
1.2 Сравнительный анализ платформ открытых данных.....	10
1.2.1 Портал открытых данных некоммерческих организаций.....	10
1.2.2 Портал открытых данных Министерства культуры РФ	12
1.2.3 Портал Data.world	13
1.2.4 Сервис для проведения соревнований по машинному обучению Kaggle	14
1.2.5 Сервис для проведения соревнований по машинному обучению Drivendata	16
1.2.6 Сервис для анализа и визуализации данных Yandex DataLens	17
1.3 Способы визуального представления данных	20
1.4 Методы понижения размерности данных	22
1.4.1 Методы нормализации данных.....	22
1.4.2 Метод главных компонент	23
1.4.3 Линейный дискриминантный анализ.....	24
1.4.4 Нейронная сеть типа автокодировщик	26
1.5 Постановка задачи	27
1.6 Анализ требований.....	28
1.6.1 Функциональные требования	29
1.6.2 Нефункциональные требования	32
2 Проектирование сервиса открытых данных.....	33
2.1 Архитектура приложения.....	33
2.2 Проектирование структур данных	34
2.2.1 Логическая модель базы данных сервиса.....	34
2.2.2 Физическая модель базы данных	36
2.3 Проектирование программных средств.....	37
2.3.1 Проектирование классов сервиса открытых данных	37
2.3.2 Проектирование алгоритма загрузки данных	41

2.3.3	Проектирование алгоритма создания табличного набора данных	43
2.3.4	Проектирования алгоритма визуализации данных	44
2.3.5	Проектирования алгоритма уменьшения размерности данных	46
2.4	Проектирование пользовательского интерфейса web-приложения	49
2.5	Проектирование клиентского приложения	59
3	Реализация программного обеспечения сервиса	61
4	Контроль качества программного обеспечения.....	67
4.1	Функциональное тестирование.....	67
4.2	Модульное тестирование	70
4.3	Метрики кода.....	72
4.4	Планирование разработки и оценка бюджета.....	76
	Заключение	79
	Список использованных источников	80
	Приложение А. Глоссарий	83
	Приложение Б. Листинг приложения.....	85
	Приложение В. Результаты тестирования	98

Введение

Развитие технологий на современном этапе связано с повсеместным внедрением методов анализа, обработки и представления данных для выработки грамотных управленческих решений. Так, например, основанием для принятия бизнес-решений сегодня служат разнообразные данные, собранные при создании и реализации товаров и услуг [1, 2, 3]. Анализ информации является неотъемлемой частью ведения бизнеса и одним из важных факторов повышения его конкурентоспособности [4]. Ученым необходимы данные для проведения различных исследований и тестирования разработанных методов, это позволяет стимулировать разработки и максимизировать их пользу для общества [5, 6, 7].

Очень часто создание собственного набора для решения конкретных задач представляет собой достаточно трудоемкую задачу, требующую множество человеко-часов работы. Сэкономить время на сборе и подготовке данных можно благодаря использованию общедоступных наборов данных, готовых к использованию для задач машинного обучения.

Открытые данные – это важнейший компонент многих общедоступных работ, связанных с искусственным интеллектом (ИИ), без них невозможно обучение алгоритмов на больших объёмах данных и реальных задачах [8]. Актуальность разработки сервиса хранения, поиска и визуализации пользовательских наборов данных обусловлена необходимостью повышения доступности и качества данных, требуемых для развития технологий искусственного интеллекта.

Целью выпускной квалификационной работы является разработка сервиса хранения, поиска и визуализации открытых данных. Разработка включает в себя анализ требований, проектирование, реализация и тестирование программного обеспечения.

1 Анализ сервисов открытых данных

1.1 Понятие открытых данных и стандарты их публикации

Понятие открытых данных на данный момент стало широко распространенным и является одним из важнейших факторов развития различных сфер, таких как обеспечение открытости государства, прозрачности корпораций, доступности результатов научных исследований для их воспроизводимости, развития технологий искусственного интеллекта.

Можно дать следующее определение открытым данным. Открытые данные – общедоступная информация, размещенная в Интернете для неоднократного, свободного и бесплатного использования в виде машиночитаемых систематизированных данных, в формате, позволяющем их автоматизированную обработку [9].

Концепция открытых данных воплощает данную идею о том, что определенные данные должны быть в свободном доступе для использования и дальнейшей повторной публикации без ограничений авторского права, патентов и других механизмов контроля.

Из определения можно выделить, что данные должны быть машиночитаемого вида, т.е. они должны быть доступны для обработки машинными методами. Можно выделить следующие машиночитаемые форматы данных:

- XLS, XLSX – форматы файлов, используемые табличными редакторами;
- CSV – наиболее распространенные формат публикации табличных данных, который условно неограничен по объему, но при этом слабо поддерживает валидацию;
- JSON – распространенный формат обмена сообщениями в web-приложениях, который еще позволяет в структурированном виде публиковать данные;

- XML один из наиболее стандартизированных форматов файлов для обмена и публикации данных, который часто используется в государственных и корпоративных системах;
- SQL форматы файлов для резервного хранения баз данных, хорошо машиночитаемые и подходящие для публикации больших наборов данных;
- ZIP, 7z, RAR – архивный формат файлов, которые используются как контейнеры для хранения данных большого объема.

На данный момент в мире уже сложились основные принципы и стандарты публикации открытых данных:

- принципы Хартии открытых данных;
- стандарт публикации открытых данных Frictionless Data;
- открытые стандарты для данных от Университета Джона Хопкинса;
- открытые стандарты публикации данных в Project Open Data;
- открытые стандарты для данных от The ODI.

Хартией открытых данных выделено шесть основных принципов, которые предоставляют согласованный на глобальном уровне набор норм для публикации данных [10].

Выделены следующие принципы:

- данные являются открытыми по умолчанию это значит, что любые опубликованные данные являются открытыми, если только правительство или автор не обоснуют необходимость закрытия, например, из соображений безопасности;
- открытые данные должны быть актуальными;
- данные должны быть доступными и удобными для использования, это значит, что данные должны быть машиночитаемыми и их можно было бы легко найти, допустим через поисковый механизм;
- данные должны быть сопоставимы и совместимы, если это возможно;

- открытые данные необходимы для лучшего управления и вовлечение граждан в дела государства;
- открытые данные можно использовать в целях развития и инновации в разных сферах общества.

Также необходимым атрибутом любых открытых данных является использование лицензии, которая обеспечит правовую основу открытости данных. Лицензия на открытые данные должна быть свободной или открытой. Открытая лицензия в отличие от свободной дает максимально широкие права, такие как право на бесплатное повторное использование, переработку и распространение. При наложении запрета на коммерческое использование лицензия не считается открытой, но при этом она может считаться свободной.

Ограничения, которые могут накладывать данные лицензии – это требование при повторной публикации указать ссылку на автора или на источник. Данный вид свободных лицензий называется пермиссивным.

Другой вид лицензии называется вирусным. Он накладывает еще одно ограничение: все производные от произведения должны распространяться точно по такой же лицензии, что и оригинал.

Самым распространенным семейством лицензий на данные являются лицензии семейства Creative Commons. Данный вид лицензии часто используют для непрограммного контента, таких как тексты, фотографии, музыка, видео и т.д. К определению непрограммного контента попадают и любые наборы данных. Выбранную лицензию необходимо указать в паспорте набора открытых данных, проставив ссылку на текст лицензии, иначе данный набор не будет считаться открытым.

Также открытые данные можно распространять под лицензиями GNU General Public License (GPL) и Open Database License (ODbL).

1.2 Сравнительный анализ платформ открытых данных

Основная задача подсистемы сервиса для хранения, поиска и анализа пользовательских наборов данных состоит в том, чтобы предоставить основным ее пользователям сервис для свободного распространения наборов данных для машинного обучения, имеющий удобный инструментарий поиска необходимых наборов данных и их визуализации.

Рассмотрим наиболее популярные сервисы, которые предоставляют открытый доступ к наборам данных, а также инструментарий, позволяющий визуализировать данные.

1.2.1 Портал открытых данных некоммерческих организаций

Целью данной платформы являлось раскрытие данных некоммерческими организациями. Благодаря данной платформе данные организации могут свободно распространять свои информационные продукты [11]. Интерфейс, предоставляющий информацию о наборе данных представлен на рисунке 1.

The screenshot displays the interface of the 'Open Data Portal of Non-Profit Organizations'. On the left, there is a sidebar with the logo of 'АНО Информационная культура' (ANO Information Culture) and a description of the organization. The main content area shows the details of a dataset named 'Корпус новостей с официального сайта Правительства РФ с разметкой по простоте языка' (Corpus of news from the official website of the Government of the Russian Federation with markup by simplicity of language). The dataset is described as a collection of news from the website <http://government.ru/news> with text extracted and marked with simplicity metrics. It includes statistics: 9823 news texts and 55852 sentences over 9 years (starting from 2012). Below the description, there are three download options: 'fedgovnews_final_shorter.jsonl' (JSONL format), 'fedgovnews_final_shorter.csv' (CSV format), and 'fedgovnews_final.zip' (ZIP format). Each option has a green 'Исследуй' (Study) button. At the bottom, there is a table with additional information about the dataset.

Поле	Величина
Источник	http://government.ru/news
Автор	Иван Бегтин
Администратор	Иван Бегтин
Версия	1.0
Последнее обновление	27 марта 2021 г., 10:31 (UTC+03:00)
Создано	27 марта 2021 г., 10:23 (UTC+03:00)
Дата создания документа	2020-03-27

Рисунок 1 – Открытый набор данные на портале НКО

На данном ресурсе можно публиковать различные материалы, такие как:

- статистические данные и наборы иных данных;
- различные тематические списки и реестры;
- библиографии;
- резервные копии баз данных;
- методические рекомендации и памятки;
- презентации;
- инфографика.

Преимущества:

- возможность поиска наборов данных с помощью строки поиска и по различным критериям, таких как организация, загрузившая материалы, темы, теги, форматы данных и лицензии;
- наличие подробной метаинформации о наборе данных, такой как описание набора, лицензия использования, информация о владельце, формат набора данных, данные о изменении и создании, тэги и т.д.;
- наличие возможности изменения загруженных наборов данных.
- большое количество поддерживаемых типов файлов, такие как PDF, DOCX для отчетов о данных и CSV, XLSX, JSON, XML, BSON, SQL, ZIP для самих данных;
- наличие подробных практических инструкций.

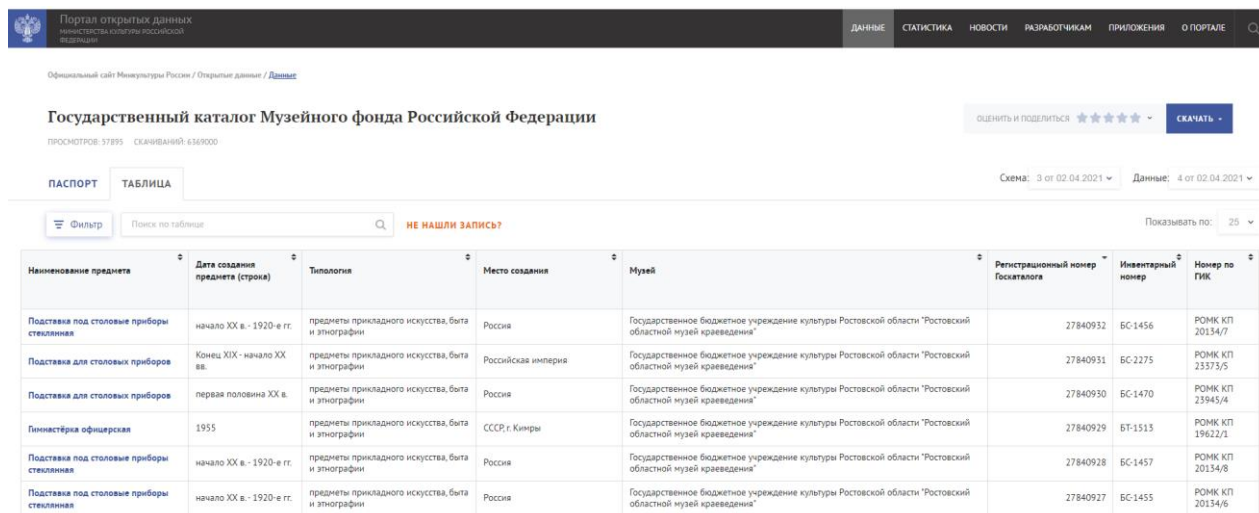
Недостатки:

- отсутствие возможности предварительного просмотра;
- слабая наполненность данными;
- отсутствие инструментов визуализации;
- возможность загрузки набора данных только от лица некоммерческой организации.

Портал данных НКО предоставляет хороший инструментарий для распространения открытых наборов данных в России для некоммерческих организаций. Но по причине общего непонимания принципов работы с данными данных организаций данный портал обладает малым количеством открытых наборов данных.

1.2.2 Портал открытых данных Министерства культуры РФ

Министерство культуры РФ разработало специальный портал публикации открытых данных. На данный момент, этот портал является самым эксплуатируемым органами власти в нашей стране [12]. На рисунке 2 представлен предварительный просмотр данных в табличном виде.



Портал открытых данных
МИНИСТЕРСТВА КУЛЬТУРЫ РОССИЙСКОЙ ФЕДЕРАЦИИ

Официальный сайт Министерства культуры России / Открытые данные / Данные

Государственный каталог Музейного фонда Российской Федерации

ПРОСМОТРОВ: 37895 СКАЧИВАНИЙ: 636900

ОЦЕНИТЬ И ПОДЕЛИТЬСЯ ★★★★★ СКАЧАТЬ

ПАСПОРТ ТАБЛИЦА

Схема: 3 от 02.04.2021 Данные: 4 от 02.04.2021

Фильтр Поиск по таблице НЕ НАШЛИ ЗАПИСЬ? Показывать по: 25

Наименование предмета	Дата создания предмета (строка)	Типология	Место создания	Музей	Регистрационный номер Госкаталога	Инвентарный номер	Номер по ГИК
Подставка под столовые приборы стелляная	начало XX в. - 1920-е гг.	предметы прикладного искусства, быта и этнографии	Россия	Государственное бюджетное учреждение культуры Ростовской области "Ростовский областной музей краеведения"	27840932	БС-1456	РОМК КП 20134/7
Подставка для столовых приборов	Конец XIX - начало XX вв.	предметы прикладного искусства, быта и этнографии	Российская империя	Государственное бюджетное учреждение культуры Ростовской области "Ростовский областной музей краеведения"	27840931	БС-2275	РОМК КП 23373/5
Подставка для столовых приборов	первая половина XX в.	предметы прикладного искусства, быта и этнографии	Россия	Государственное бюджетное учреждение культуры Ростовской области "Ростовский областной музей краеведения"	27840930	БС-1470	РОМК КП 23945/4
Пиннакель офицерская	1955	предметы прикладного искусства, быта и этнографии	СССР, г. Кинзы	Государственное бюджетное учреждение культуры Ростовской области "Ростовский областной музей краеведения"	27840929	БТ-1513	РОМК КП 19622/1
Подставка под столовые приборы стелляная	начало XX в. - 1920-е гг.	предметы прикладного искусства, быта и этнографии	Россия	Государственное бюджетное учреждение культуры Ростовской области "Ростовский областной музей краеведения"	27840928	БС-1457	РОМК КП 20134/8
Подставка под столовые приборы стелляная	начало XX в. - 1920-е гг.	предметы прикладного искусства, быта и этнографии	Россия	Государственное бюджетное учреждение культуры Ростовской области "Ростовский областной музей краеведения"	27840927	БС-1455	РОМК КП 20134/6

Рисунок 2 – Предварительный просмотр открытого набора данных на сайте Министерства культуры РФ

Преимущества:

- наличие связных данных, что объединят различные наборы в целостное пространство;
- возможность подписки на обновление наборов данных;
- возможность предварительного просмотра данных в таблице и на карте;
- морфологический поиск;
- фильтрация по темам, востребованности, дате создания и изменения и по названию;
- наличие программного интерфейса REST API в форматах JSON и XML;
- наличие подробной метаинформации о наборе данных;
- наличие обратной связи.

Недостатки:

- в связи с тем, что портал является государственным, отсутствует возможность авторизации и загрузки собственных наборов;
- отсутствие инструментария визуализации набора данных;
- ограниченный набор тем наборов данных и общее малое количество наборов;

На данный момент это самый крупный государственный портал открытых наборов данных, который имеет значительное число частных и корпоративных пользователей, таких как Яндекс, 2do2go и других.

1.2.3 Портал Data.world

Data.world специальный портал для публикации данных. После загрузки данных можно его интегрировать с различными инструментариями для анализа и визуализации данных [13]. На рисунке 3 представлен пример страницы с метаданной о наборе данных.

The screenshot shows the Data.world portal interface. The main content area displays the 'ANW Obstacle History' dataset by Matt Laessig. It includes a description, a summary, and a table of data. The table has columns for season, location, round stage, obstacle name, and obstacle order. The data shows the first five obstacles from the 1st season in Venice.

#	season	location	round stage	obstacle name	#	obstacle order
1	1	Venice	Qualifying	Quintuple Steps		1
2	1	Venice	Qualifying	Rope Swing		2
3	1	Venice	Qualifying	Rolling Barrel		3
4	1	Venice	Qualifying	Jumping Spider		4
5	1	Venice	Qualifying	Pipe Slider		5

Рисунок 3 – Страница с метаданной об открытом наборе данных

Преимущества:

- наличие инструментария поиска поисковая строка, поиск по тэгам, статусу, владельцу набора данных;

- поддержка различных типов файлов (ZIP, CSV, XLS, JSON, SQL) и возможно загрузки наборов через URL, т.е. загрузки набора данных через облачное хранилище файлов;
- для данных в табличном формате можно осуществлять выборку и фильтрацию с помощью языка, похожего на SQL;
- наличие программного интерфейса REST API и специального модуля на языке Python для работы с хранилищем набора данных;
- наличие инструментария для визуализации данных с возможностью агрегации данных по определенному признаку;
- возможность связывания данных.

Недостатки:

- отсутствие главной страницы со списком всех доступных наборов данных;
- неинтуитивный интерфейс, отсутствие возможности перехода из отчетов по набору данных к странице самих наборов данных;
- при предварительном просмотре выгружается весь набор данных, что очень замедляет работу портала.

Данный сервис обладает достаточно богатым инструментарием не только для публикации открытых наборов данных, но и для их визуализации и анализа. Но данный портал обладает недостатками, описанными выше, что делает его сложным и не особо удобным в использовании.

1.2.4 Сервис для проведения соревнований по машинному обучению Kaggle

Kaggle – это самая распространенная система организации конкурсов по исследованию данных, а также социальная сеть специалистов в области науки о данных.

Среда является публичной платформой, где пользователи могут открыто публиковать и исследовать наборы данных, а также взаимодействовать с другими специалистами [14]. На рисунке 4 представлен пример табличного предварительного просмотра данных с отображением частотных гистограмм.

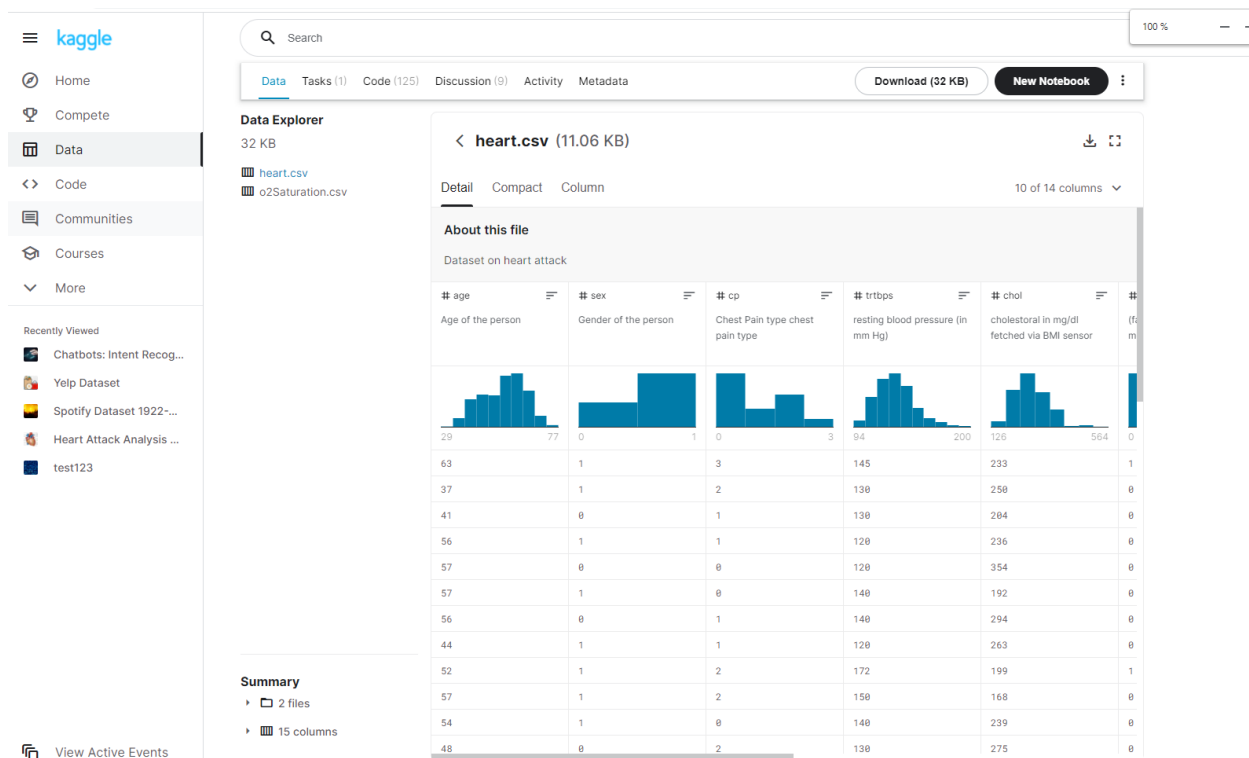


Рисунок 4 – Предварительный просмотр открытого набора данных на платформе Kaggle

Преимущества:

- возможность открыто публиковать наборы данных и проводить исследования;
- доступ к облачным инструментам для обработки данных и машинного обучения;
- каждый набор данных имеет набор тэгов, описание, возможность предварительного просмотра содержания набора данных;
- наличие большого сообщества специалистов в области машинного обучения и аналитики данных;
- для файлов формата csv для каждого столбца строится гистограмма частот.

Недостатки:

- инструментарий для анализа данных представлен в виде jupyter notebook, где требуется знание языка python;
- нет встроенного инструментария для анализа данных и построения графиков и создания дашбордов.

В целом данный сервис является самой популярной платформой в мире для проведения соревнований по машинному обучению. Также эта платформа является очень хорошим средством публикации открытых данных с возможностью анализа и визуализации данных с помощью языка Python.

1.2.5 Сервис для проведения соревнований по машинному обучению Drivendata

Drivendata тоже является платформой для проведения соревнований по машинному обучению и анализу данных для поиска лучших статистических моделей сложных задач прогнозирования [15].

В отличие от Kaggle, Drivendata обладает другим подходом к проведению соревнований. Для участия в соревновании необходимо принять пользовательское соглашение. Также отсутствует возможность свободно распространения своих наборов данных, и результатов анализа.

Платформа использует механизм оценки, с помощью которой модели автоматически ранжируются в зависимости от того, насколько хорошо они решают задачу предсказывают, в зависимости от выбранной метрики.

Преимущества:

- обладает встроенным средством оценки обученных моделей;
- возможность поиска по различным критериям и организации собственных соревнований по машинному обучению;
- имеется подробное описание о структуре данных и о задаче, которую необходимо решить.

Стоит сказать, что данная платформа является средой распространения наборов данных, но она не соответствует концепции открытых данных. Для получения доступа к данным для проведения исследования необходимо подписать пользовательское соглашение, которое часто запрещает изменять и распространять данный набор данных.

1.2.6 Сервис для анализа и визуализации данных Yandex DataLens

Yandex DataLens представляет набор связанных сервисов, в том числе и для визуализации данных. В отличие от сайтов по проведению соревнований по машинному обучению данная платформа предоставляет инструментарий, для визуализации данных. Является российским аналогом подобных сервисов от Google и Amazon. На рисунках 5 представлена страница предварительного просмотра данных таблицы.

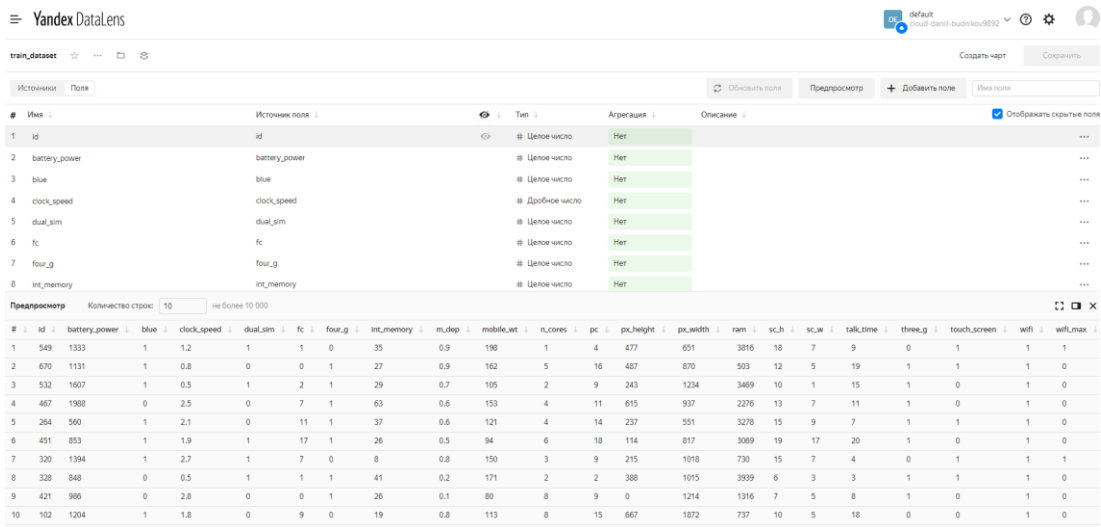


Рисунок 5 – Предварительный просмотр набора данных с помощью DataLens

На рисунке 6 виден пример построение линейной диаграммы с областями.

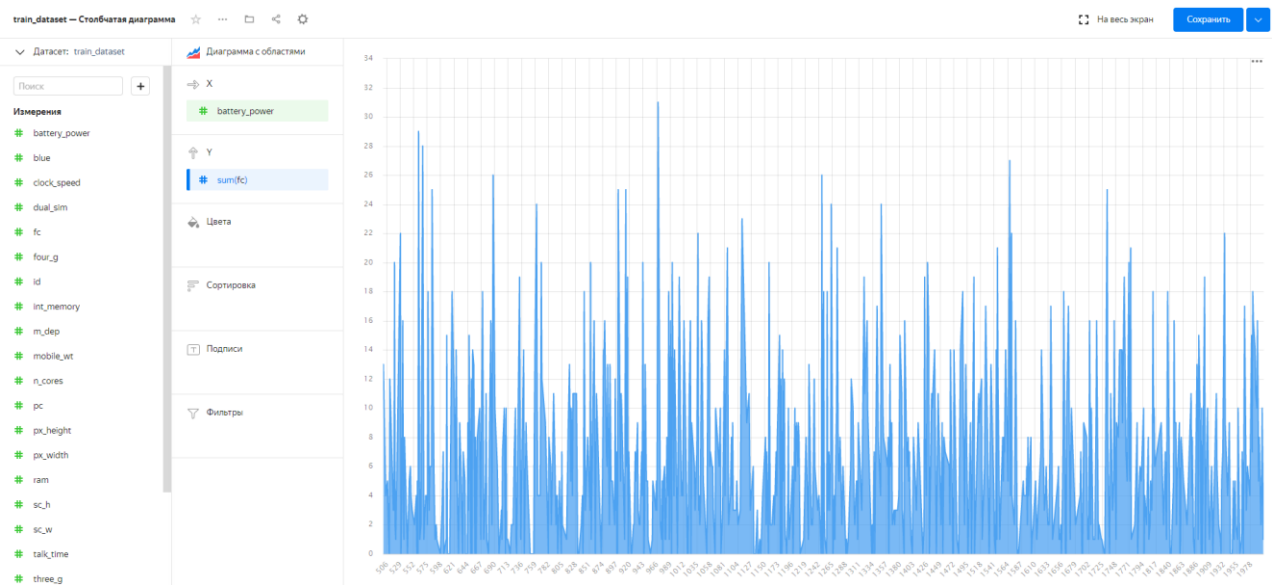


Рисунок 6 – Пример построения диаграммы с помощью DataLens

Преимущества:

- поддерживает разные типы файлов, такие как csv файлы, базы данных ClickHouse, MySQL и др.;
- можно строить линейные, круговые, точечные диаграммы, гистограммы, также есть возможность агрегировать данные;
- поддержка разных типов данных и возможность построения диаграмм на карте с использованием геоточек;
- есть возможность связывания наборов данных;
- из множества графиков можно создавать дашборды для аналитических отчетов;
- возможность оставлять наборы данных, графики и дашборды в открытом доступе по ссылке.

Недостатки:

- получить доступ к данным можно только по ссылке;
- отсутствуют механизмы поиска;
- нет возможности добавления метаданных к набору данных.

DataLens является мощным инструментом для визуализации табличных данных, но как платформа свободного распространения открытых данных она не подходит из-за отсутствия механизма поиска и добавления метаданных [16].

Сведем результаты анализа в таблицу 1.

Таблица 1 – Сравнительный анализ продуктов-аналогов

Критериальная оценка	Портал открытых данных НКО	Портал открытых данных Министерства культуры РФ	Data.world	Kaggle	Yandex DataLens
Количество открытых наборов данных	152 набора	71 набор	200 тыс. открытых наборов в данных	81 тыс. открытых наборов данных.	—
Возможность загружать наборы данных	Есть только для организаций	Нет	Да, файлы любого вида (CSV, JSON и т.д.)	Да, файлы любого вида (CSV, ZIP и т.д.)	Да, только файлы табличного вида

Продолжение таблицы 1

Критериальная оценка	Портал открытых данных НКО	Портал открытых данных Министерства культуры РФ	Data.world	Kaggle	Yandex DataLens
Строка поиска	Да	Да	Да	Да	Нет
Полнота метаданных о данных	Представлена достаточно полная информация о наборе, включая автора, лицензии, ссылки на источники и т.д.	Портал имеет очень полную метаданную о наборе данных, заполненную по международным стандартам	Можно заполнить большое количество информации (имя, описание, лицензия, тэги)	Имеется возможность достаточно полно заполнить информацию о наборе данных	Можно добавить описание столбцов таблиц данных
Фильтрация по тэгам, типом файлов, лицензии.	Присутствуют все типы фильтрации, также есть возможность фильтрации по организации	Есть фильтрации по темам. Все данные одного формата и доступны	Есть	Присутствуют все типы фильтрации, также есть возможность фильтрации по размеру набора данных	Нет
Наличие предпросмотра	Нет	Да	Да, но все данные выгружаются в память	Да, для всех типов файлов, кроме бинарного формата	Да, только первые 10 000 записей
Наличие встроенного инструмента для визуализации	Нет	Нет	Есть 3 типа диаграмм	Есть, строятся гистограммы частот для каждого столбца таблицы. Можно анализировать данные с помощью jupyter notebook	Есть 14 типов диаграмм

Результаты проведенного анализа показывают, что большинство систем имеют инструментарий для предоставления метаданных о наборах данных, поиска и фильтрации данных. При этом многие сервисы обладают слабым встроенным инструментарием для визуализации данных.

Следует также отметить, что в сервисе DataLens для визуализации данных полностью отсутствует инструментарий для распространения открытых данных, а встроенные средства визуализации на платформе Data.world предоставляют лишь ограниченный набор диаграмм.

Единственной платформой, где можно визуализировать любые типы данных являются Kaggle, так как она предоставляет облачный сервис для анализа данных, где за основу взят jupyter notebook. Но для анализа и визуализации необходимо хорошо знать язык Python и уметь пользоваться его модулями. Можно сделать вывод, что ни один сервис не предоставляет одновременно инструментария для публикации открытых данных и встроенное средство для их визуализации, которое будет понятно в использовании даже для людей, не особо знакомых с программированием.

1.3 Способы визуального представления данных

Визуализация данных – это представление данных в виде, который дают наиболее полную информацию о структуре и распределению данных в выборке. Визуализация широко применяется в научных и статических исследованиях (в прогнозировании, интеллектуальном анализе данных, бизнес-анализе). Данное направление анализа связано с инфографикой, разведочным анализом данных и статистической графикой. Также грамотная визуализация данных позволяет наглядно сравнивать различные наборы данных. В общем визуализация данных относится к представлениям, созданных алгоритмическим путем и отражающих большие объемы данных.

Визуализация может быть двух видов:

- презентационная – предназначенная для наглядного представления данных пользователю;
- исследовательская – предназначенная для анализа с целью поиска некоторых закономерностей в них.

Часто визуализация является важной частью систем интеллектуального анализа данных.

Для визуализации данных можно строить различные виды графиков, которые позволят более подробно разобраться в структуре данных. Данные можно визуализировать для разных целей:

- сравнить разные показатели;
- показать распределение данных;
- наглядно изобразить структуру данных;
- проследить взаимосвязь между различными показателями.

Можно выделить следующие основные виды графиков:

- линейный график;
- столбчатая диаграмма;
- гистограмма;
- круговая диаграмма;
- график рассеяния или точечная диаграмма.

Линейный график позволяет просмотреть динамику по одному или нескольким критериям. Например, удобно просматривать изменение какой-либо величины в разные периоды времени.

Столбчатая диаграмма идеально подходит для сравнения нескольких наборов данных по разным параметрам. Также при построении данных видов графиков можно агрегировать значения по столбцам. Например, сравнить все средние значения по одному критерию.

Гистограмма частот позволяет увидеть распределение данных и данный вид диаграмм можно отнести к визуализации одномерных данных. В общем она представляет собой графическое изображение зависимости частоты попадания элементов массива числовых данных от соответствующего интервала группировки.

Круговая диаграмма позволит узнать долю каждого значения внутри множества. На данной диаграмме длина дуги каждого среза пропорциональна количеству, которое она представляет.

График рассеяния позволяет просмотреть зависимость одной величины от другой. Данная диаграмма изображает значения двух переменных в виде

точек на декартовой плоскости. Можно использовать и полярные координаты для визуализации, например, когда данные имеют циклический характер.

Но данных средств визуализации бывает недостаточно для понятного отображения многомерных данных. Например, самый распространенный и простой способ визуализации многомерных – это построения попарных графиков рассеивания. Для каждой пары атрибутов (X , Y) строится график рассеивания. Он позволяет наблюдать за потенциальным отношением в двух измерениях для атрибутов данных. Но на данном графике сложно или практически невозможно найти более глубокие взаимосвязи, например, зависимость одного атрибута от множества критериев.

Для прослеживания таких зависимостей и для отображения многомерных данных можно воспользоваться методами уменьшения размерности данных.

1.4 Методы понижения размерности данных

Уменьшение размерности – это процесс уменьшения количества признаков до наиболее оптимального показателя [17]. При уменьшении размерности теряется часть информации, но уменьшая размерность данных до 2-х или 3-х измерений позволяет отобразить данные на графике, что позволит найти важные закономерности.

1.4.1 Методы нормализации данных

Перед этапом понижения размерности данных необходимо провести их нормализацию. То есть привести значение всех атрибутов к некоторой общей шкале без потери информации. Наиболее часто используется приведение к диапазонам $[0, 1]$ или $[-1, 1]$.

Для проведения нормализации можно воспользоваться min-max нормализацией. Новое значение данных находится по следующей формуле:

$$x'_i = \frac{x_i - X_{min}}{X_{max} - X_{min}},$$

где X_{min} – минимальное значение признака;

X_{max} – максимальное значение признака;

x_i – исходное значение признака;

x'_i – нормализованное значение признака в диапазоне $[0, 1]$.

Недостатком данного метода является то, что при наличии аномальных значение данных, данные будут концентрироваться только в узкой части диапазона. Чтобы избежать этого, можно воспользоваться нормализацией средним [18]. Новое значение вычисляется по следующей формуле:

$$x'_i = \frac{x_i - \mu_x}{\sigma_x},$$

где μ_x – среднее значение признака;

σ_x – стандартное отклонение признака.

В нормализованных данных значение $\mu_{x'} = 0$, а $\sigma_{x'} = 1$.

В методах уменьшения размерности данных предварительно используется нормализация средним.

1.4.2 Метод главных компонент

Метод главных компонент (PCA) – один из основных методов уменьшения размерности данных, с возможностью потерять наименьшее количество информации. Данный метод сводится к вычислению собственных векторов и значений ковариационной матрицы исходных данных.

По сути метод главных компонент вращает оси в направлении наибольшей дисперсии данных [18].

Первым этапом является вычисление ковариационной матрицы данных. Предположим, что матрица X , представляющие данные, была изначально нормализована методом нормализации средним. Тогда ковариационная матрица вычисляется по формуле:

$$S_{d \times d} = \mathbb{E}[X^T X],$$

где d – количество признаков;

$\mathbb{E}$ – математическое ожидание.

Имя квадратную матрицу S размерности $d \times d$ можно найти собственные значения λ_i , $i = 1..d$ и собственные вектора v_i , $i = 1..d$.

Полученные собственные вектора необходимо разместить в порядке убывания собственных значений данных векторов:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_d.$$

Получаем последовательность векторов v_i длиной d , отсортированную по значениям собственных значений. Построим матрицу V , значения столбцов которой являются собственными векторами v_i :

$$V_{d \times d} = [v_1 \quad v_2 \quad v_3 \quad \dots \quad v_d].$$

Для уменьшения размерности данных X достаточно взять первых k столбцов матрицы V (2 – 3 вектора для визуализации). В итоге получаем:

$$X'_{n \times k} = X_{n \times d} V_{d \times k},$$

где $X'_{n \times k}$ – данные с уменьшенной размерностью.

Данный алгоритм является «алгоритмом машинного обучения» без учителя. Алгоритм сам выявляет полезные структурные свойства набора данных.

1.4.3 Линейный дискриминантный анализ

Линейный дискриминантный анализ (LDA), также как и PCA, представляет собой метод линейного преобразования для уменьшения размерности данных. В отличие от PCA, LDA можно отнести к классу обучения с учителем, так как данный метод находит такой базис, который максимизирует расстояние между разными классами [18].

В отличие от PCA, нормализацию данных не получится провести методом нормализации средним. Изначально нужно выделить целевой признак Y :

$$y_i \in \{1, \dots, K\},$$

где y_i – это значение целевого признака (номер класса);

k – количество классов в выборке.

Первым этапом нужно найти матрицу средних значения признаков для классов $M_{d \times k}$, где d количество признаков:

$$M = \begin{bmatrix} \mu_{1,1} & \dots & \mu_{1,d} \\ \vdots & \ddots & \vdots \\ \mu_{k,1} & \dots & \mu_{k,k} \end{bmatrix},$$

где $\mu_{i,j}$ – среднее значение j -го признака, принадлежащего i -му классу.

Далее считается матрица рассеяния внутри классов W :

$$W = \sum_{i=1}^k \sum_{y_j=i} (X_j - M_i)(X_j - M_i)^T,$$

где k – количество классов;

y_j – значение j -го целевого признака;

X_j – j -ая строка исходного набора данных;

M_i – i -ая строка матрицы средних значения признаков для классов.

Также необходимо посчитать матрицу рассеяния B между классами. Она считается по формуле:

$$B = \sum_{i=1}^k N_i (M_i - \mu)(M_i - \mu)^T,$$

где N_i – количество элементов выборки, принадлежащей классу i ;

μ – вектор средних значений признаков по всей выборке.

Далее, как и в методе главных компонент, необходимо найти собственные вектора и собственные значения матрицы $W^{-1}B$ для получения линейных дискриминант. Количество полученных собственных векторов будет равно количеству классов минус один.

По аналогии с методом главных компонент, полученные собственные вектора сортируются по не возрастанию собственных значений. И для того, чтобы уменьшить размерность данных, нужно выбрать первых s собственных векторов (2 или 3 для визуализации) и умножить исходную матрицу X на данную матрицу.

Плюсом данного метода является то, что он достаточно прост в реализации, как и метод главных компонент, и является методом обучения с учителем, что позволяет более наглядно визуализировать данные по классам. Данный метод можно использовать для кластерного анализа данных.

1.4.4 Нейронная сеть типа автокодировщик

Автокодировщик – это специальная архитектура нейронной сети, позволяющая применять обучение без учителя.

Данная нейронная сеть имеет три основных части: кодировщик, декодер и функция потерь, которая помогает обучить сеть отображать данные друг в друга (из пространства большей размерности в пространства меньшей размерности и наоборот).

Общая структура автокодировщика представлена на рисунке 7.

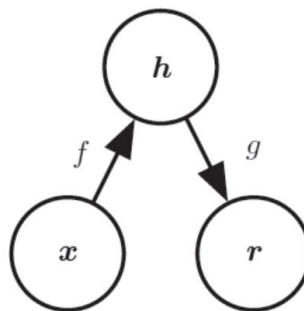


Рисунок 7 – Общая структура кодировщика

В общем данная сеть отображает вход x на выход r , через внутренне представление h . При этом f называется кодировщиком, а g – декодером. Для визуализации данных необходимо получить код h размерностью 2 или 3.

Процесс обучения можно описать как минимизацию функции потерь:

$$L(x, g(f(x))),$$

где L – функция, штрафующая $g(f(x))$ за непохожесть выхода на начальные данные x .

Если декодер – это линейная функция, а L – среднеквадратичная ошибка, то понижающий кодировщик обучается тому же подпространству, что и метод главных компонент. Следовательно, если функции f и g нелинейные, то сеть может обучиться более мощным нелинейным обобщениям [19].

Минусом данного подхода, что он достаточно сложен в реализации и для выбора подходящих параметров, таких как глубина нейронной сети, тип сети, выбора функции потерь, оптимизатора и количество эпох обучения нужно

проводить отдельные исследования. Также поэтому автоматизация понижения размерности данных через данные сети является достаточно сложной задачей.

Поэтому в качестве алгоритмов понижения размерности были выбраны PCA и LDA.

1.5 Постановка задачи

В результате анализа предметной области было принято решение разработать web-сервис.

Основная задача сервиса для хранения и поиска пользовательских наборов данных состоит в том, чтобы предоставить основным ее пользователям средства для свободного распространения наборов данных, имеющий инструментарий поиска, предпросмотра, оценивания и визуализации данных.

Ограничения на тип загружаемых данных не будет наложено, но при этом невозможно поддерживать аналитическую работу со всеми типами файлов. Поэтому загрузку данных можно осуществлять в любом формате, в том числе будет возможность загрузки архивов, а визуализировать данные можно только из csv файлов.

Также один из важных параметров открытых данных является метаданные о наборе данных, поэтому необходимо, что каждый набор данных хранил информацию о тематике, размер набор, типы файлов, описание набора, дата создания и т.д.

Из этапа анализа можно выделить следующие типы графиков, которые будут реализованы: столбчатая, линейная и круговая диаграмма, с возможностью агрегации данных, график рассеяния двух величин, с возможностью изобразить данные с уменьшенной размерностью.

Таким образом, разрабатываемое приложение должно решать следующие задачи:

- загрузка наборов данных: загрузка файлов и архивов данных, автоматическое формирование метаданных для анализа, добавление данных для упрощения поиска и использования (тегов, описания);

- просмотр списка наборов данных: показать все публичные наборы данных, отсортировать по критериям (тегам, типам файлов и т.д.), просмотр содержимого набора данных, возможность оставить отзыв;
- визуализация набора данных с помощью построения выделенных типов графиков.

Разрабатываемая система должна быть удобна в использовании, уметь обрабатывать табличные данные и предоставлять программный интерфейс для взаимодействия с основным функционалом системы.

1.6 Анализ требований

Анализ требований является важным этапом проектирования, на котором в соответствии с бизнес-процессами можно сформулировать функциональные требования к системе.

Проектируемый сервис предназначен для свободного распространения открытых наборов данных под одной из открытых лицензий. Для загрузки на сервер собственного набора данных необходима процедура авторизации и регистрации пользователя.

Для пользователей, которые хотят воспользоваться набором данных для дальнейшего исследования необходим отдельный блок с метаинформацией о наборе данных, а также возможностью предварительного просмотра данного набора. Каждый авторизованный пользователь имеет свой личный кабинет, где представлен список всех его наборов данных.

Поиск набора данных может осуществляется пользователем по нескольким критериям (по названию, дате, тэгам, популярности, типам файлов).

Также для визуализации набора данных необходим отдельный инструментарий, позволяющий выбрать тип графика и столбцы для визуализации.

1.6.1 Функциональные требования

Согласно поставленным задачам, были выделены следующие функциональные требования:

- регистрация и авторизация;
- просмотр списка открытых наборов данных (с возможность сортировки и фильтрации);
- предварительный просмотр;
- загрузка собственного набора данных;
- скачивание набора данных;
- оставление оценки;
- визуализация табличных данных.

На рисунке 8 представлена диаграмма вариантов использования [20].

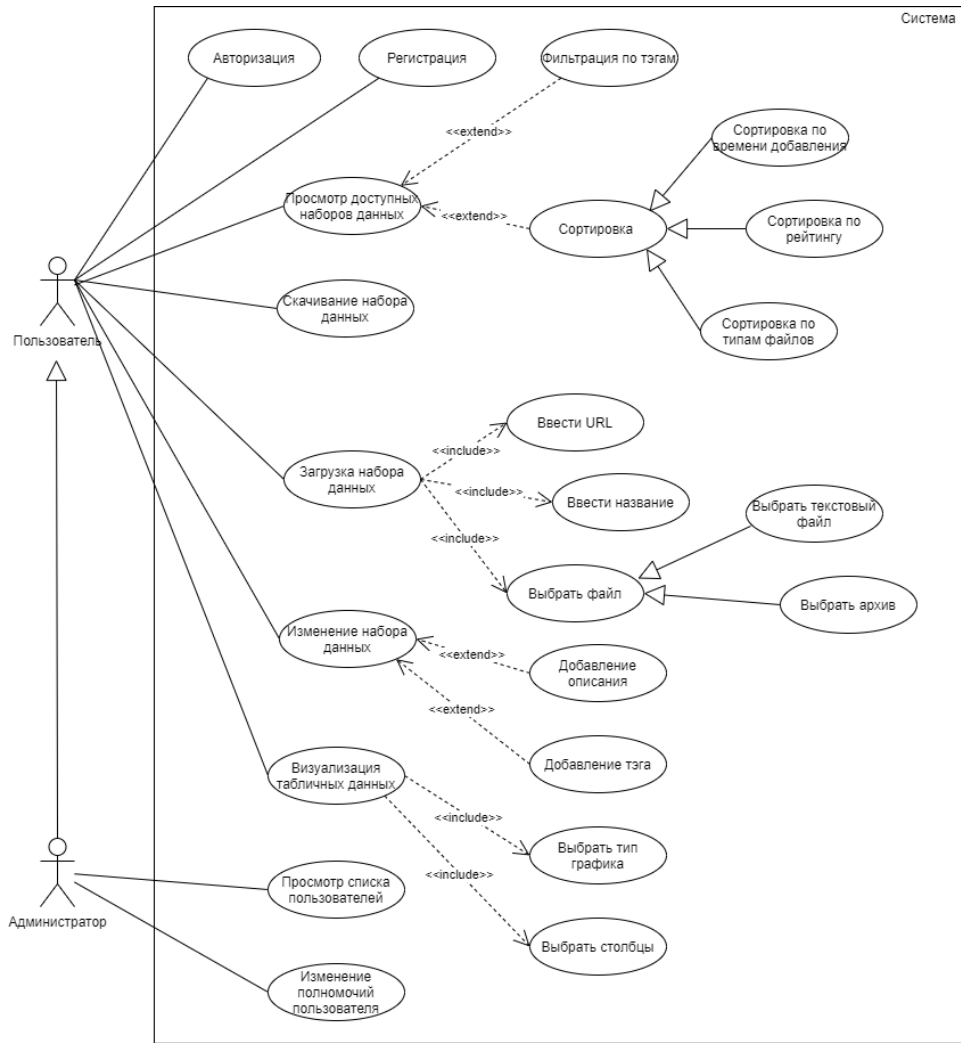


Рисунок 8 – Диаграмма вариантов использования

Спецификация прецедента «Загрузка набора данных» на рисунке 9.

Прецедент: Загрузка набора данных
ID: 1
Краткое описание: Пользователь загружает свой набор данных в виде архива или файла для анализа
Главные актеры: Система, пользователь
Второстепенные актеры: Нет
Предусловие: У пользователя есть токен авторизации
Основной поток: <ol style="list-style-type: none"> 1. Пользователь вводит имя набору данных (оно будет отражаться в URL набора данных) 2. Пользователь вводит заголовок (он будет отображаться в списке всех публичных наборов данных) 3. Пользователь выбирает файл или архив для загрузки. 4. Система сохраняет этот набор данных и формирует всю необходимую метainформацию 5. Вся информация сохраняется системой в БД
Постусловия: Набор данных сохранен в системе, и информация о нем сохранена в БД системы
Альтернативные потоки: Были введены некорректные данные

Рисунок 9 – Спецификация прецедента «Загрузка набора данных»

На рисунке 10 представлена спецификация прецедента «Регистрация».

Прецедент: Регистрация
ID: 2
Краткое описание: Пользователь регистрируется в системе
Главные актеры: Система, пользователь
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: <ol style="list-style-type: none"> 1. Если пользователь вводит некорректные данные (короткий пароль) <ol style="list-style-type: none"> 1.1. Система выдает предупреждение 2. Иначе, если пользователь с данной почтой или именем уже существует в системе <ol style="list-style-type: none"> 2.1. Система выдает предупреждение, что такой пользователь уже существует 3. Иначе <ol style="list-style-type: none"> 3.1. Пользователь добавляется в систему
Постусловия: Пользователь сохранен в БД системы.
Альтернативные потоки: Нет

Рисунок 10 – Спецификация прецедента «Регистрация»

На рисунке 11 представлена спецификация прецедента «Визуализация табличных данных».

Прецедент: Визуализация табличных данных
ID: 3
Краткое описание: Пользователь создает графическое представление табличных данных
Главные актеры: Система, пользователь
Второстепенные актеры: Нет
Предусловие: Пользователь авторизован и набор данных был загружен
Основной поток: 1. include «Выбрать тип графика» 2. include «Выбрать столбцы» 3. Система по выбранным параметрам формирует массив данных для визуализации 4. Система визуализирует сформированные данные
Постусловия: Был отображен график
Альтернативные потоки: Нет

Рисунок 11 – Спецификация прецедента «Визуализация табличных данных»

На рисунке 12 представлена спецификация прецедента «Фильтрация по тэгам».

Прецедент: Фильтрация по тэгам
ID: 4
Краткое описание: Пользователь формирует список наборов данных, имеющих указанные тэги
Главные актеры: Система, пользователь
Второстепенные актеры: Нет
Предусловие: Пользователь открыл страницу со списком наборов данных
Основной поток: 1. Пользователь отрывает панель с поиском доступных тэгов 2. Пользователь вводит в поле поиска необходимые тэги 3. Система выдает все тэги, сортированные по схожести с введенным словом 4. Пользователь выбирает необходимые тэги 5. Пользователь нажимает на поиск наборов данных
Постусловия: Был отображен список наборов с соответствующими тэгами
Альтернативные потоки: Нет

Рисунок 12 – Спецификация прецедента «Фильтрация по тэгам»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы при выполнении определенных функций.

1.6.2 Нефункциональные требования

Серверная часть приложения должна быть выполнена на языке Python с использованием фреймворка Flask. Язык Python был выбран так, как он обладает простым синтаксисом и множеством математических библиотек. Необходимо реализовать RESTful web-сервис, позволяющий использовать реализованные сервисы на любом клиентском устройстве. В качестве СУБД будет использоваться PostgreSQL. Для работы с БД необходимо использовать ORM SQLAlchemy.

Клиентская часть приложения будет реализована на языке JavaScript с использованием фреймворка React. Приложение должно работать в web браузерах, реализованных на движках Webkit и Blink (Chrome, Safari, Firefox).

При разработке необходимо использовать редактор исходного кода Visual Studio Code.

2 Проектирование сервиса открытых данных

2.1 Архитектура приложения

Архитектура сервиса хранения, поиска и визуализации наборов открытых данных основана на REST архитектуре и шаблоне проектирования MVC. MVC архитектура является наиболее распространенной при проектировании web-приложений, так как разделения на слои Model, View, Controller естественно ложится на структуру подобных сервисов. На рисунке 13 представлена схема взаимодействия приложений с REST API, построенного по MVC архитектуре.

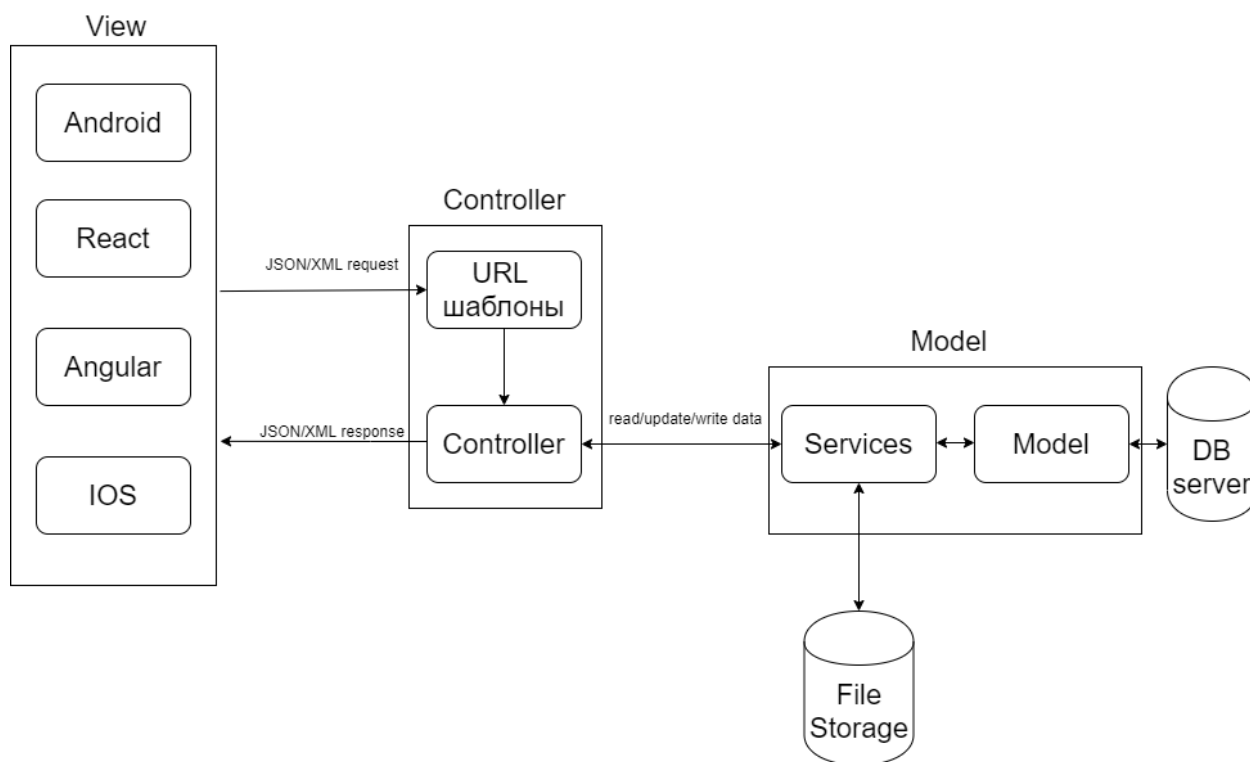


Рисунок 13 – Схема взаимодействий приложений с API в MVC архитектуре

Также из данной модели видно, что данный сервис основан на трехуровневой модели, состоящей из клиента, сервера приложений и сервера базы данных. Данная модель накладывает несколько основных ограничений на приложение:

- уровень клиента содержит минимум бизнес-логики приложения;
- уровень клиента не имеет прямой доступ к серверу БД;

– уровень сервера приложений должен обеспечивать возможность горизонтального масштабирования.

Клиентом в данной системе может являться любое клиентское устройство, например, Android, IOS или браузер.

2.2 Проектирование структур данных

2.2.1 Логическая модель базы данных сервиса

На основе проведенного анализа предметной области и функциональных требований была построена логическая модель базы данных с использованием нотации «сущность-связь». Логическая модель отражает суть выбранной предметной области и выделяет основные сущности, атрибуты и связи в системе.

В начале необходимо выделить список всех основных сущностей данной предметной области. Данный список представлен в таблице 2.

Таблица 2 – Описание сущностей предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Списочный состав пользователей, зарегистрированных в системе	Стержневая	1000	5% в неделю
2	Группа	Список групп с их разрешениями	Вспомогательная	10	Очень редко
3	Группы пользователей	Список групп и принадлежащих к ним пользователей	Вспомогательная	1000	5% в неделю
4	Набор данных	Списочная информация о загруженных в системе наборах данных	Стержневая	10000	5% в день
5	Категории наборов данных	Список категорий, к которым относится набор данных	Вспомогательная	50	Очень редко
6	Информация о наборе данных	Список файлов набора, которые доступны в системе	Вспомогательная	10000	5% в день

Продолжение таблицы 2

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
7	Оценка пользователя	Список оценок пользователей определенных наборов данных	Стержневая	1000	10% в день
8	Тип файлов	Список типов файлов, принадлежащих наборам данных	Стержневая	10	Очень редко
9	Табличный набор данных	Список табличных данных типа csv, которые можно менять и визуализировать	Стержневая	1000	5% в день
10	Исходные столбцы таблицы данных	Список неизменных столбцов таблицы данных, состоящий из названия, типа агрегации и типа данных столбца csv файла	Вспомогательная	10000	5% в день
11	Измененные столбцы таблицы данных	Список столбцов таблицы, где пользователь может менять их название, тип данных и агрегацию. Ссылаются на исходные столбцы, чтобы не произошло потери данных	Стержневая	10000	10% в день
12	График таблицы данных	Список всех графиков, построенных на основе определенной таблицы	Стержневая	1000	5% в день

На основании данных сущностей необходимо построить модель. Для создания модели использовался язык IDEF1X с использованием CASE-средства Erwin Data Modeler [21]. На рисунке 14 изображена построенная логическая модель сервиса.

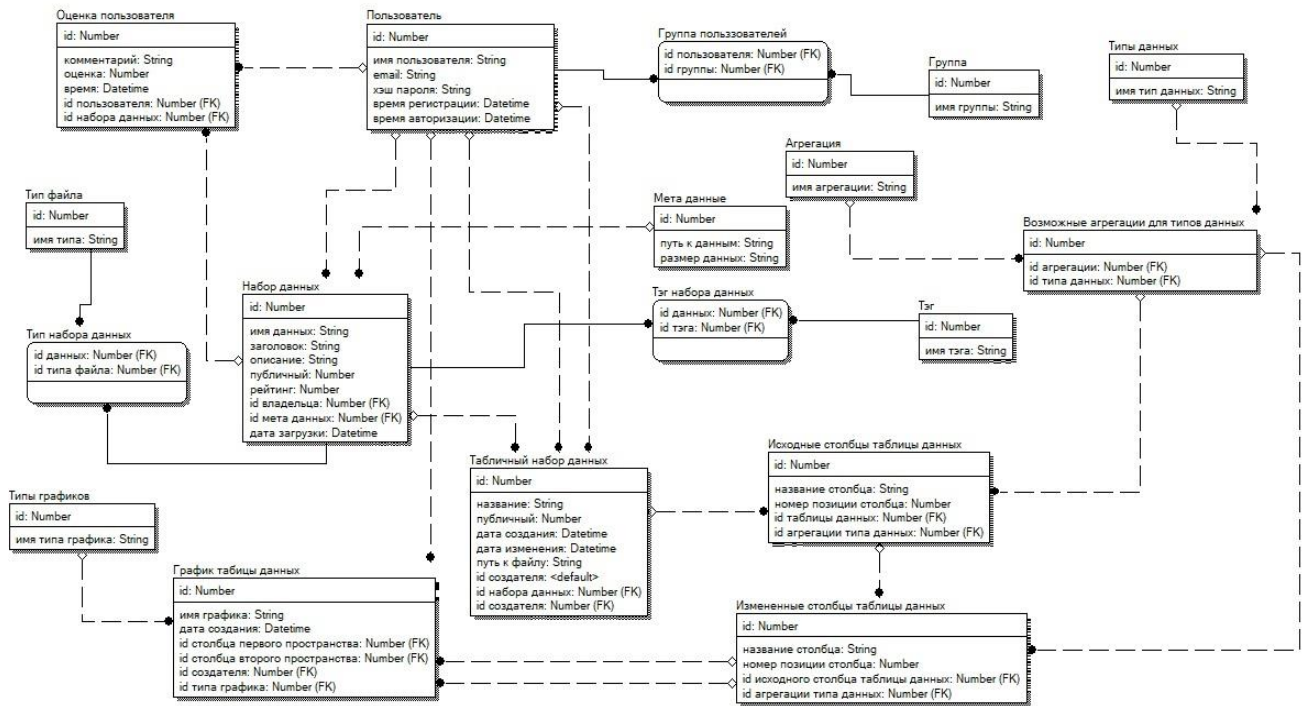


Рисунок 14 – Логическая схема БД

От данного уровня модели БД нужно перейти к физическому.

2.2.2 Физическая модель базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД. Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. В качестве СУБД будет использован PostgreSQL версии 12.

Плюсами данной СУБД является то, что она свободно распространяемая и имеет реализацию для множества UNIX-подобных платформ. Также данная СУБД имеет поддержку множества различных типов данных, такие как JSON, XML, массивы, uuid идентификаторы и т.д.

На основании логической модели была разработана физическая модель базы данных. Физическая модель БД в нотации IDEF1X представлена на рисунке 15.

наследованные от Creator занимаются формированием записей, которые будут сохранены в БД. Классы Facade, содержат в себе запросы к БД с использованием ORM.

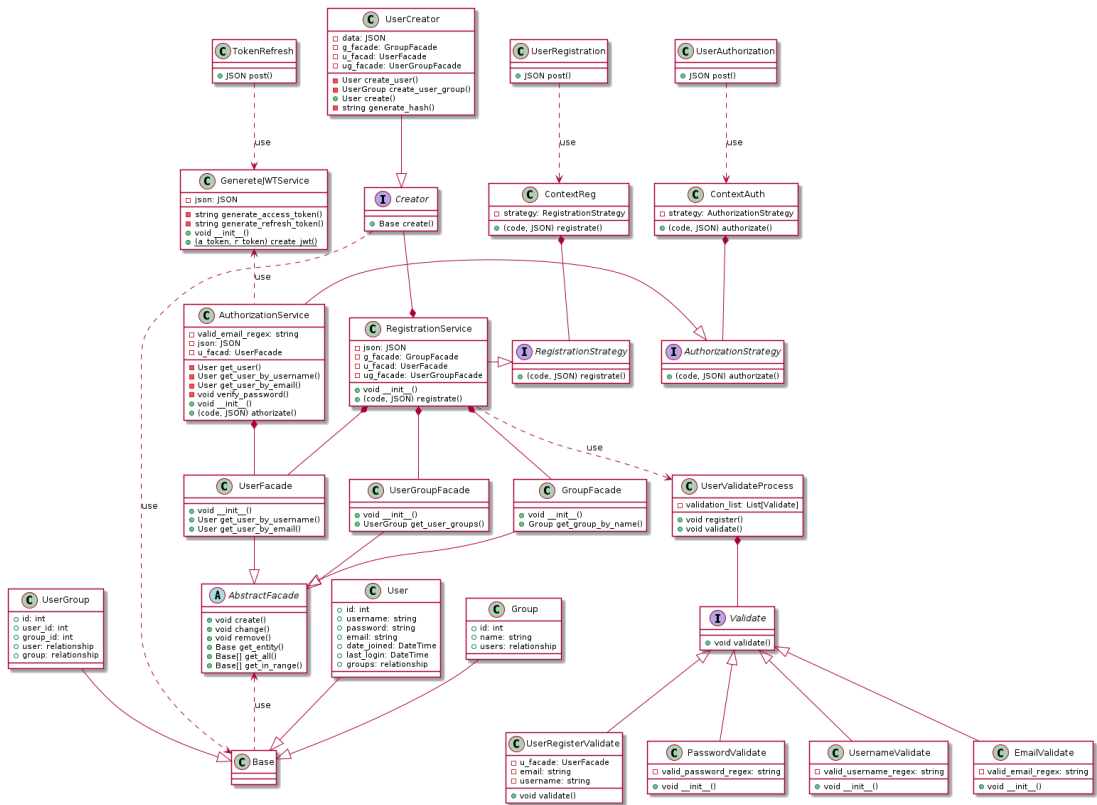


Рисунок 16 – Диаграмма классов для сервиса авторизации и регистрации

В качестве примера, для классов сервиса авторизации и регистрации была составлена спецификация. Спецификация класса `RegistrationService` описана в таблице 3.

Таблица 3 – Спецификация класса «RegistrationService»

Наименование класса: RegistrationService
Назначение класса: содержит в себе бизнес логику регистрации пользователя
Поля класса: json: dict – Json в теле запроса пользователя при обращении к контроллеру регистрации g_facade: GroupFacade – Фасад для обращения и взаимодействия с таблицей Group в БД u_facade: UserFacade – Фасад для обращения и взаимодействия с таблицей User в БД ug_facade: UserGroupFacade – Фасад для обращения и взаимодействия с таблицей UserGroup в БД.

Продолжение таблицы 3

<p>Методы класса:</p> <p>create_user – создания объекта модели таблицы User</p> <p>create_user_group – создания объекта таблицы UserGroup</p> <p>check_user – проверка на существование пользователя с данным именем или email'ом</p> <p>generate_hash – создание хэша пароля пользователя</p> <p>registrate – метод регистрации пользователя</p>

В таблице 4 представлена спецификация класса AuthorizationService, ответственного за бизнес -логику процесса авторизации.

Таблица 4 – Спецификация класса «AuthorizationService»

Наименование класса: AuthorizationService
Назначение класса: содержит в себе бизнес логику авторизации пользователя по JWT токену
<p>Поля класса:</p> <p>json: dict – Json в теле запроса пользователя при обращении к контроллеру авторизации</p> <p>u_facade: UserFacade – Фасад для обращения и взаимодействия с таблицей User в БД</p> <p>valid_email_regex: string – Регулярное выражение для проверки строки на содержание в ней email'a</p>
<p>Методы класса:</p> <p>get_user – получение записи о пользователя из БД</p> <p>get_user_by_username – получение записи о пользователя из БД по его имени</p> <p>get_user_by_email – получение записи о пользователя из БД по его email</p> <p>verify_password – метод верификации пароля пользователя</p> <p>athorizate – метод авторизации пользователя</p>

В таблице 5 представлена спецификация класса GenerateJWTService, содержащий в себе логику создания JWT токенов.

Таблица 5 – Спецификация класса «GenerateJWTService»

Наименование класса: GenerateJWTService
Назначение класса: содержит в методы генерации access и refresh JWT токенов для авторизации
<p>Поля класса:</p> <p>json: dict – Json в теле запроса пользователя при обновить access token</p>
<p>Методы класса:</p> <p>generate_access_token – генерация access токена (основного токена для подтверждения авторизации пользователя)</p> <p>generate_refresh_token – генерация refresh токена (он необходим для получения нового токена авторзизии, он не имеет время жизни)</p> <p>create_jwt – создание JWT токена (пары access и refresh токенов по identity – записи, однозначно идентифицирующей пользователя)</p>

На рисунке 17 представлена диаграмма классов для сервиса загрузки набора данных.

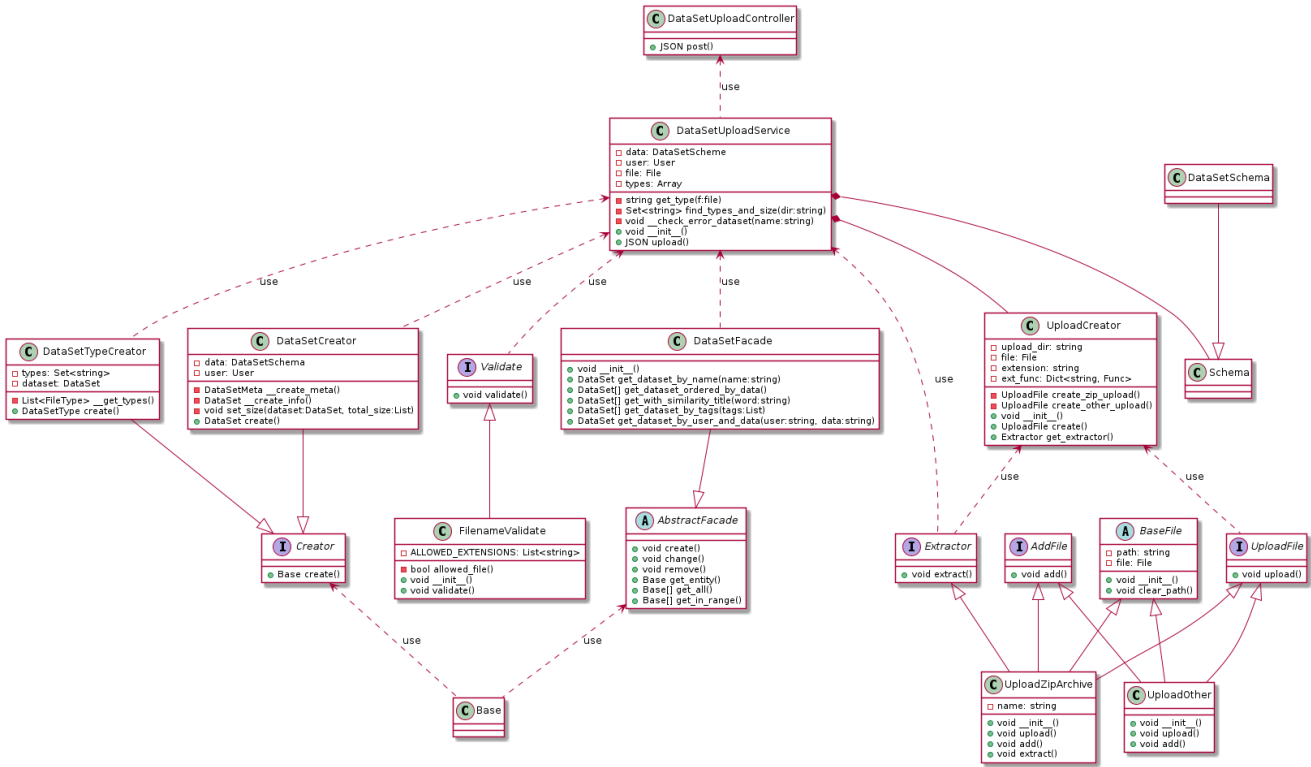


Рисунок 17 – Диаграмма классов для сервиса загрузки набора данных

Класс UploadCreator сохраняет на диске набор данных в зависимости от типа файла. Основная логика сохранения набора данных содержится в классе DataSetUploadService. Спецификация этого класса представлена в таблице 6.

Таблица 6 – Спецификация класса «DataSetUploadService»

Наименование класса: DataSetUploadService
Назначение класса: содержит в себе бизнес логику загрузки открытого набора данных
Поля класса: data: DataSetScheme – десериализованное тело запроса с информацией о загружаемом наборе данных user: User – пользователь, загружающий набор данных, информация получена из токена file: File – файл, который пользователь хочет сохранить в системе types: List<string> – список основных типов файлов для загрузки
Методы класса: get_type – получение расширения из имени файла find_types_and_size – получение всех типов файлов в наборе и размер набора __check_error_dataset – проверка набора на правильность имени файла и имени набора данных upload – метод сохранения файла набора данных и информации о нем в БД

Таким образом для основных модулей системы были спроектированы классы. Если в систему загружается архив, то он распаковывается. Данный метод реализуется в классах наследников от `Extractor`.

2.3.2 Проектирование алгоритма загрузки данных

С помощью диаграммы деятельности можно разбить некоторую длительную операцию на составные части. При этом отображается последовательность данных операций, где каждое переход от одной операции срабатывает только при завершении операции в предыдущем состоянии.

Одной из важнейших операций является загрузка набора открытых данных. Данная операция является основной для распространения открытых наборов данных. После ее выполнения на файловом хранилище сервера будет лежать загруженный набор данных, к которому в дальнейшем посредством API можно будет обратиться.

Данная операция должна удовлетворять нескольким важным условиям, чтобы обеспечить приемлемое время ожидания обработки набора данных пользователем:

- должна давать информацию об окончании процесса передачи набора данных на сервер;
- должна информировать пользователя о завершении процесса обработки загруженного набора данных.

Диаграмма деятельности загрузки набора данных представлена на рисунке 18.

Пользователь перед загрузкой набора данных должен авторизоваться и получить JWT токен для выполнения операции загрузки. После получения токена пользователь должен ввести URL и имя будущего набора данных, а также выбрать сам файл для загрузки. Изначально система проверит введенные данные и сообщит об ошибке.

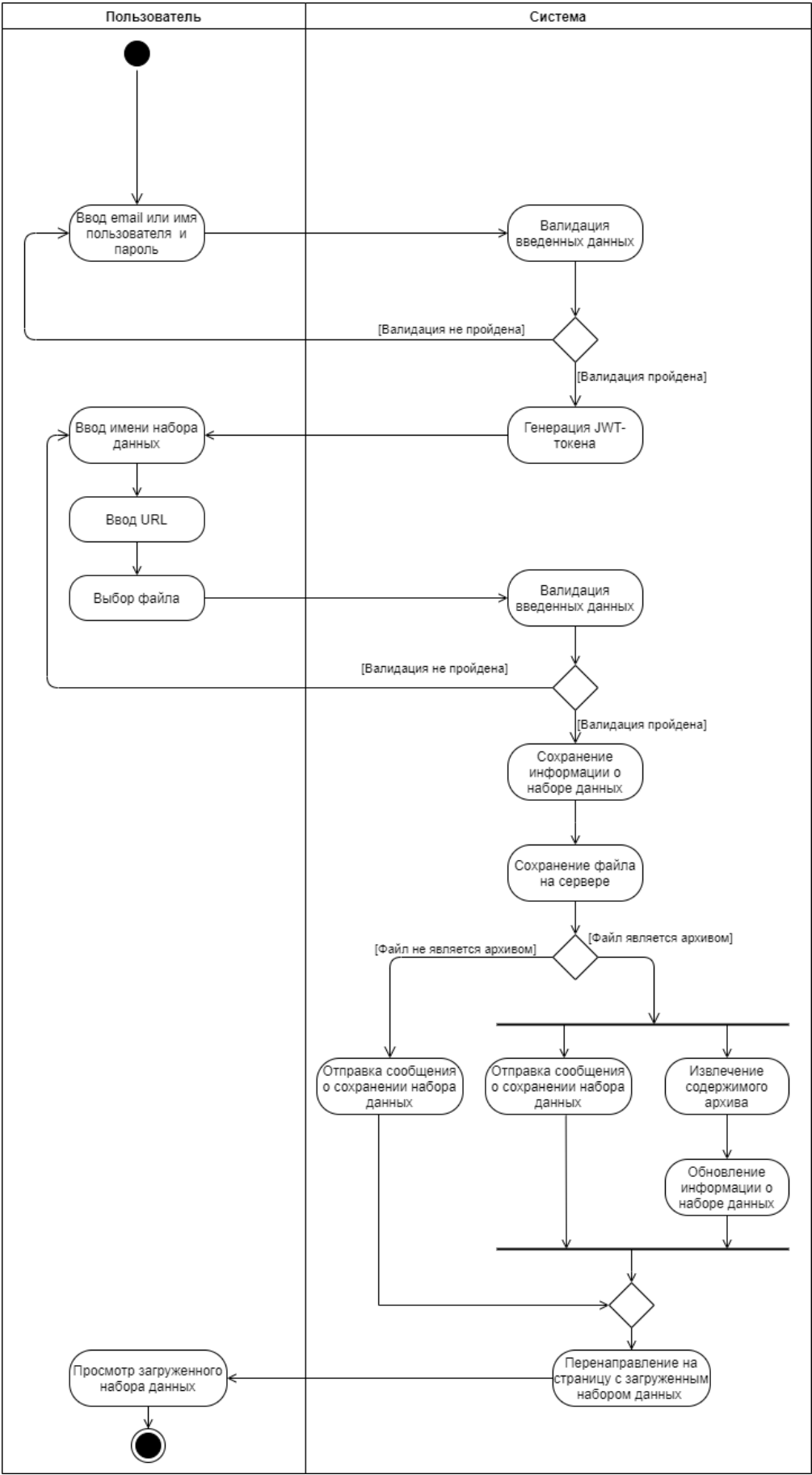


Рисунок 18 – Диаграмма деятельности «Загрузка набора данных»

После валидации введенных данных серверу будет передан файл для дальнейшей обработки. Когда система получит файл, то пользователь будет об этом уведомлен и сервис продолжит параллельно обрабатывать загруженный набор данных. В конце обработки пользователя перенаправят на страницу с данным набором данных.

2.3.3 Проектирование алгоритма создания табличного набора данных

Диаграмма последовательности описывает коммуникационный аспект между основными объектами системы в некотором сценарии. На данной диаграмме описан процесс создания табличного набора данных (рисунок 19).

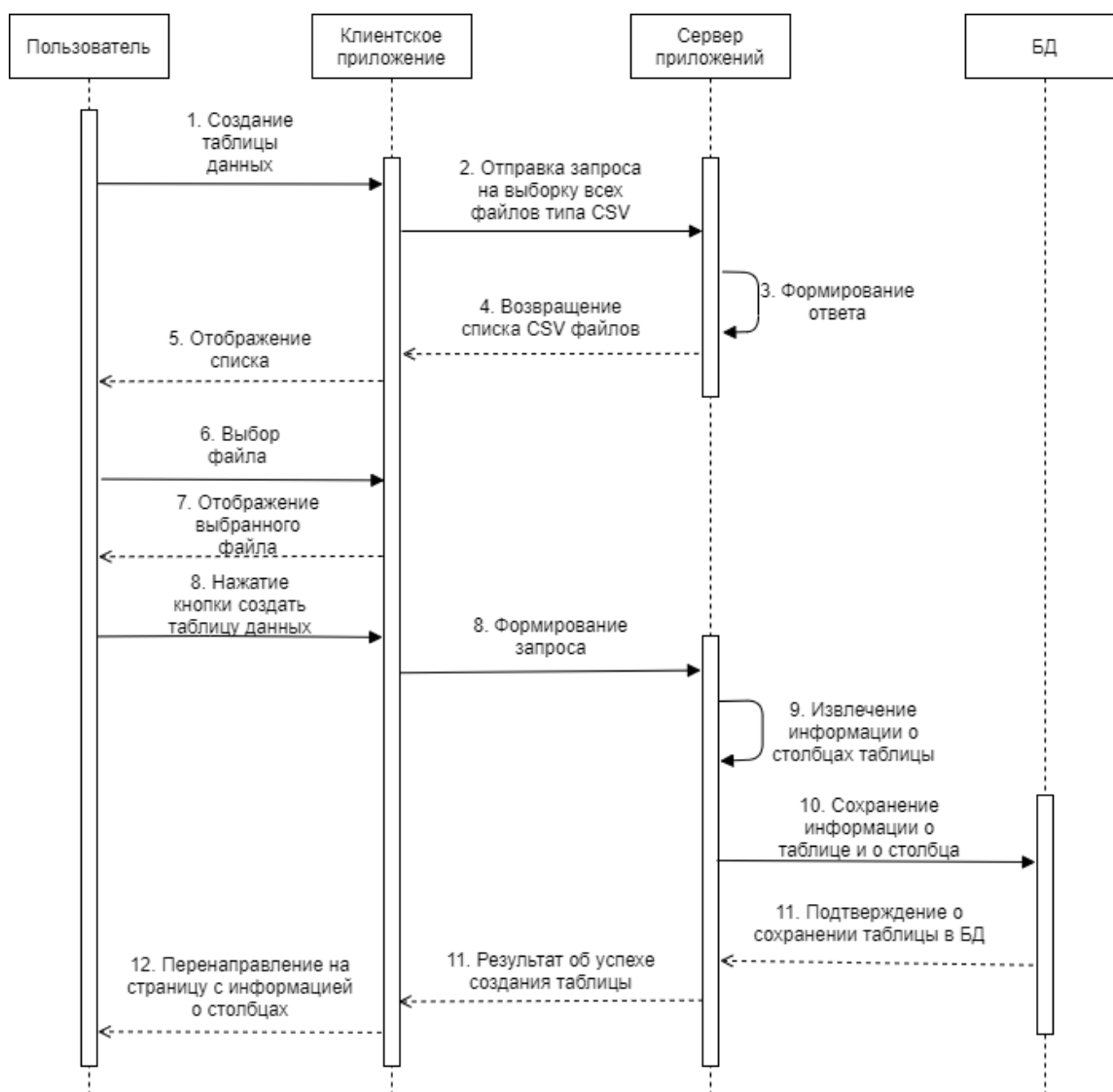


Рисунок 19 – Диаграмма последовательности процесса создания табличного набора данных

Эта диаграмма включает 4 основных объекта: пользователь, клиентское приложение, сервер приложений и сервер БД. Первым действием пользователь открывает панель создания табличного набора данных, после чего клиентское приложение формирует запрос о выборке возможных кандидатов для создания таблицы из загруженного набора данных. На стороне сервера приложений формируется список файлов CSV формата.

После этого пользователь выбирает необходимый файл и дает ему название. После подтверждения запроса на создании таблицы данных клиентское приложение формирует запрос и отправляет его на сервер приложений.

Сервер приложений собирает всю необходимую информацию о файле и отправляет запрос на сервер БД, где сохраняется таблица данных. Далее пользователь получает подтверждение о том, что таблица данных была создана.

2.3.4 Проектирования алгоритма визуализации данных

При создании табличного набора данных можно сказать, что создается признаковое описание множества объектов X , или же нашей таблицы данных. Имея признаковое описание объектов:

$$f_j: X \rightarrow D_j, j = 1 \dots n,$$

где X – множество объектов;

D_j – значение j -го признака объекта;

n – количество признаков.

При этом можно выделить следующие типы признаков:

- бинарный признак, где $D_j = \{0, 1\}$;
- номинальный признак (адрес, профессия), где $|D_j| < \infty$;
- порядковый признак (образование), где $|D_j| < \infty$, D_j упорядочено;
- количественный признак, где $D_j = R$.

Количество признаков для объектов может быть сколь угодно большим. Изначально количество признаков равно количеству столбцов таблицы. Но

при этом возможно порождать их сколь угодно количество. Для порождения нового признака можно воспользоваться функциями агрегации. Функция агрегации – это функция, в которой значения нескольких строк сгруппированы вместе, чтобы сформировать единое итоговое значение по средствам некоторой функции.

Для любого типа признака, которого группируют, можно посчитать значение функции COUNT – количество. Для бинарных и количественных признаков можно посчитать значения следующих функций:

- AVERAGE – среднее значение (для бинарного признака очевидно покажет процентное соотношение значений поля);
- MAXIMUM – максимальное значение;
- MINIMUM – минимальное значение;
- SUM – сумма всех элементов.

Стоит отметить, что порожденные признаки являются зависимыми, что очевидно, что допустим не поможет в решении задач машинного обучения. Но используя данные функции можно найти некоторые зависимости в исходном признаковом пространстве, что также поможет более наглядно визуализировать данные.

Не был упомянут упорядочено-порядковый признак, так как без знания ПО решаемой задачи нельзя отличить данный признак от номинального.

Диаграмма состояний отображает переход объекта от одного состояния к другому.

Для создания графиков разработан отдельный инструментарий, который при определенных действиях меняет свое состояние. Изначально отображается список всех возможных столбцов таблицы данных.

После отображения пользователь должен выбрать тип графика и значения переменных x и y . Если тип графика не рассеивающая диаграмма, то при извлечении данных будет применена агрегатная функция для обеспечения функционального отношения

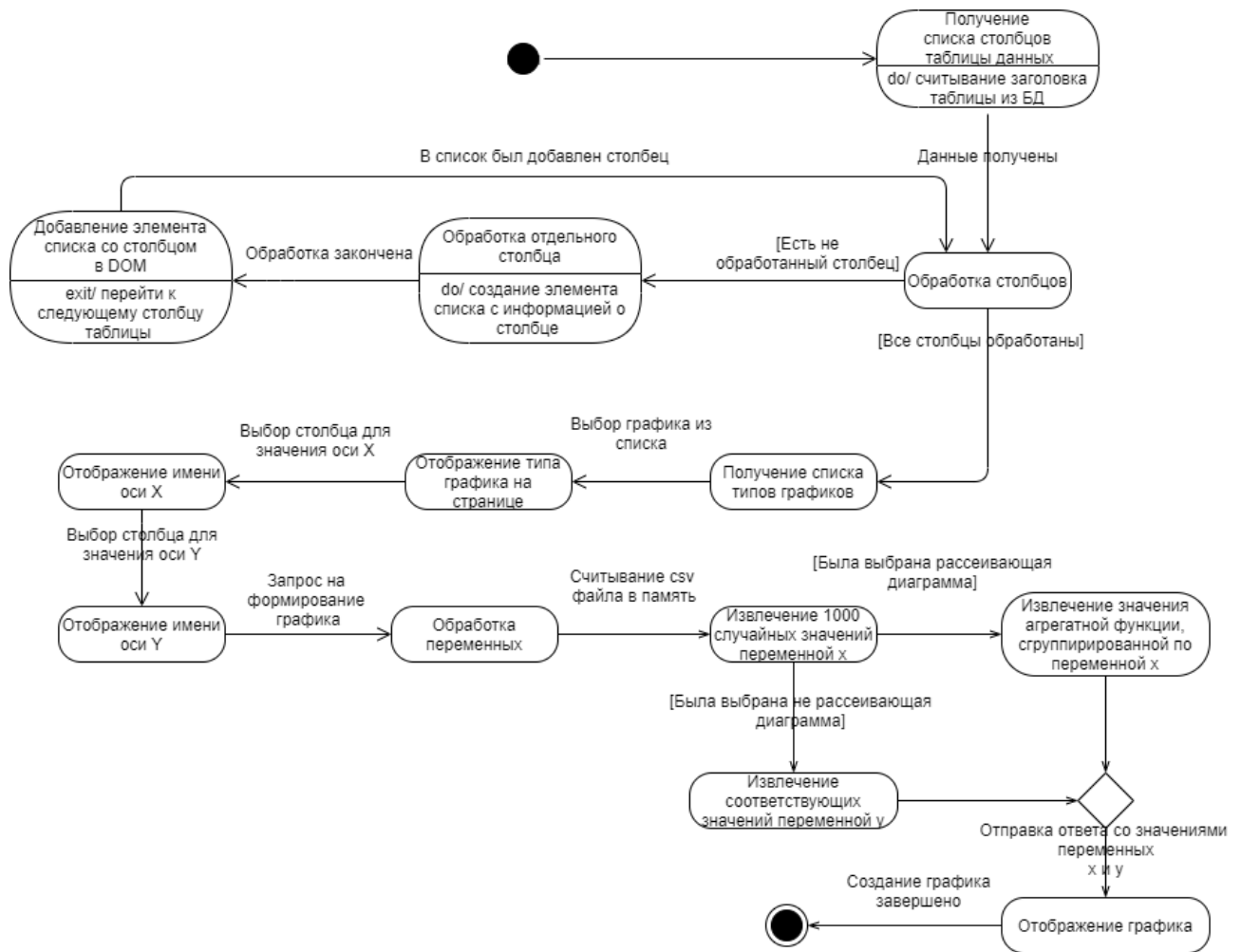


Рисунок 20 – Диаграмма состояний при создании графика

Стоит отметить, что при уменьшении данных и при визуализации происходит выборка только из 1000 строк. Это сделано для того, чтобы облегчить нагрузку на клиентское устройство при визуализации данных. Данные строки выбираются случайно, что позволяет более точно визуализировать исходный набор данных.

После того, как сервер отправит клиенту данные для визуализации, клиентское приложение в зависимости от типа выбранного графика визуализирует данные.

2.3.5 Проектирования алгоритма уменьшения размерности данных

При визуализации данных часто возникает необходимость визуализации полного пространства объектов, а не только конкретных признаков.

Используемые методы для уменьшения размерности данных и их работа были описаны ранее.

Стоит отметить, что используемые методы PCA и LDA являются методами проекции данных многомерного пространства на подпространство меньшей размерности. При этом данные методы являются линейными, что при наличии данных с нелинейными зависимостями невозможно будет говорить о наилучшем выборе проекции для отображения данных.

Для решения данной задачи необходимо реализовать методы нелинейного понижения размерности, например, сети типа автоенкодер.

При визуализации данных с уменьшением размерности с помощью PCA, пользователь должен выбрать только соответствующий тип графика. При использовании LDA, пользователь также должен выбрать столбец с классами метод находит такой базис, который будет использован для нахождения оптимального базиса.

Диаграмма деятельности для процесса понижения размерности данных для визуализации представлена на рисунке 21. При уменьшении размерности данных необходимыми этапами являются очистка данных от нулевых значений.

Можно воспользоваться разными техниками для очистки данных, например, заполнение нулевых значений средними, но при этом качество визуализации испортится. Поэтому в качестве меры для очищения данных было выбрано удаление столбцов с нулевыми значениями в столбцах, так в данное решение не смещает значения выборки.

Для всех номинальных признаков нужно провести факторизацию – процесс присвоения строковым значениям уникальных числовых идентификаторов.

После этапа факторизации для вычисления PCA и LDA необходимо провести нормализацию средним. Получившееся множество объектов подается на вход методам для вычисления LDA и PCA.

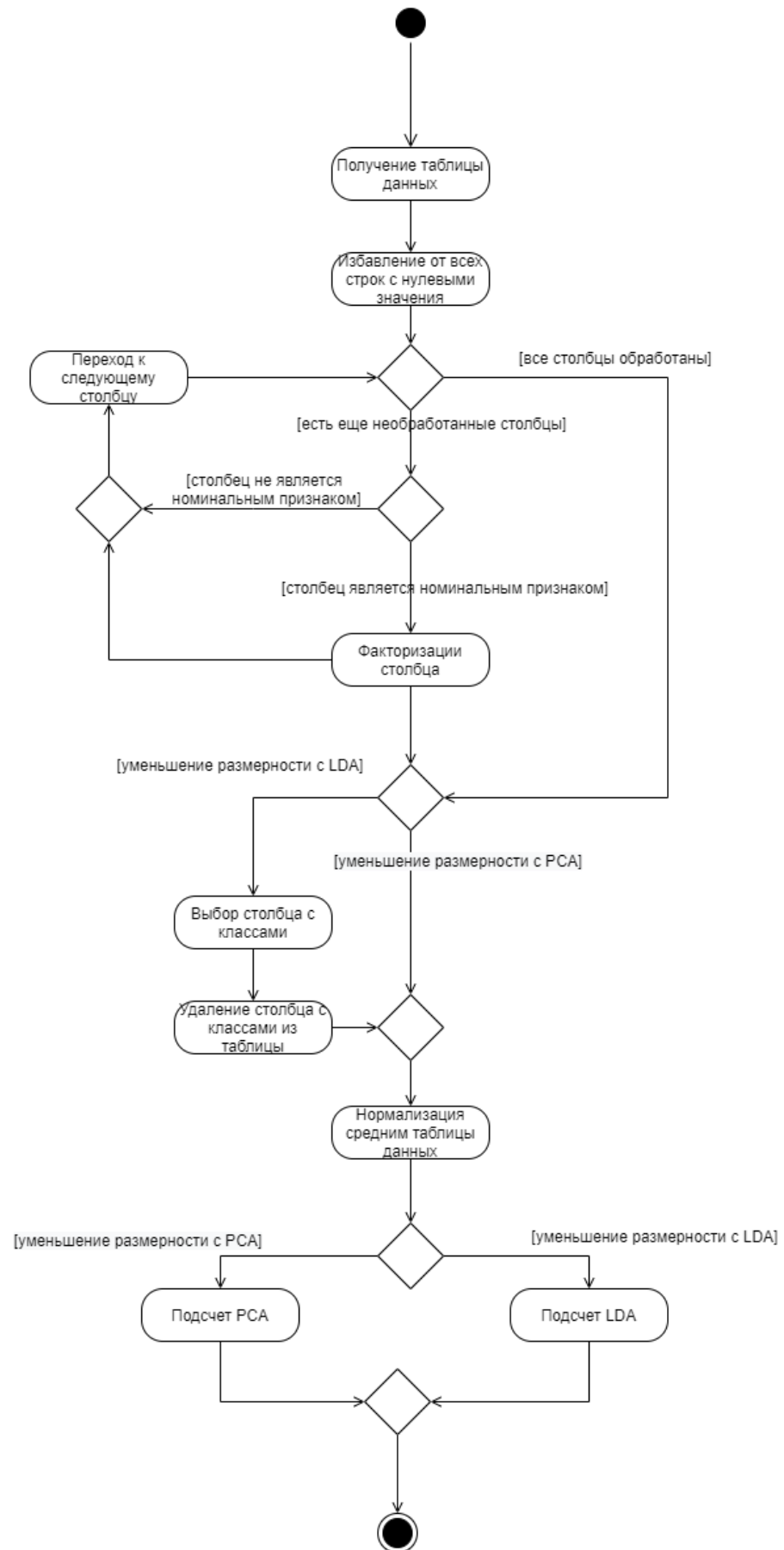


Рисунок 21 – Диаграмма деятельности при уменьшении размерности данных

Получившийся массив данных отправляется клиенту, где клиентское приложение строит диаграмму рассеивания. Таким образом спроектирован сервис для хранения, обработки и визуализации открытых данных.

2.4 Проектирование пользовательского интерфейса web-приложения

На основе диаграммы вариантов использования, для клиентского приложения был спроектирован графический интерфейс. Пользовательский интерфейс должен обеспечивать простое и эффективное выполнение основных функций пользователей.

Интерфейс пользователя сервиса для хранения, поиска и визуализации открытых наборов данных будет состоять из нескольких окон:

- окно со списком всех наборов данных и возможностью поиска и сортировки;
- окно с возможностью скачивания и просмотра детальной информации о наборе данных и о его содержимом, а также возможность оставить свой отзыв;
- окно авторизации;
- окно регистрации
- модальное окно загрузки пользовательских наборов данных
- окно профиля пользователя;
- административная панель со списком пользователей;
- модальное окно для добавления пользователя в группу;
- модальное окно создания табличного набора данных;
- окно изменения табличного набора данных;
- окно создания графиков для табличных данных.

На рисунке 22 представлен интерфейс окна со списком всех наборов данных.

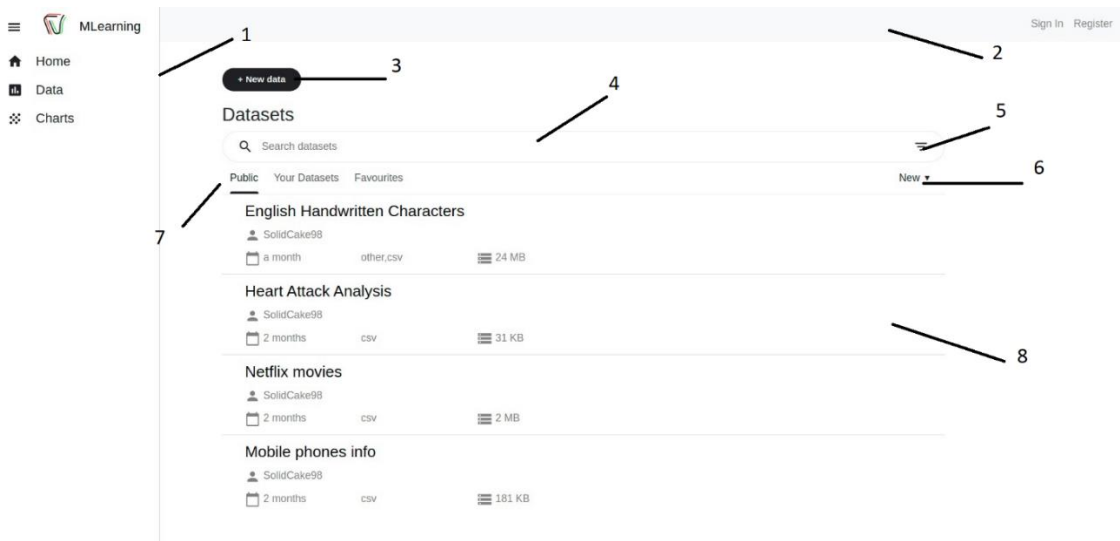


Рисунок 22 – Окно со списком всех наборов данных

Основные элементы интерфейса данного окна, представленного на рисунке 22:

- 1 – боковая навигационная панель для перехода к главным страницам;
- 2 – верхняя навигационная панель;
- 3 – кнопка перехода к загрузке пользовательского набора данных;
- 4 – панель поиска набора данных по заголовку;
- 5 – кнопка перехода к панели фильтрации наборов данных по тэгам;
- 6 – выпадающее меню для выбора способа сортировки наборов данных;
- 7 – панель выбора типов наборов данных;
- 8 – список наборов данных.

При нажатии на кнопку фильтрации на экране появится модульное окно с возможностью выбора набора тэгов для фильтрации.

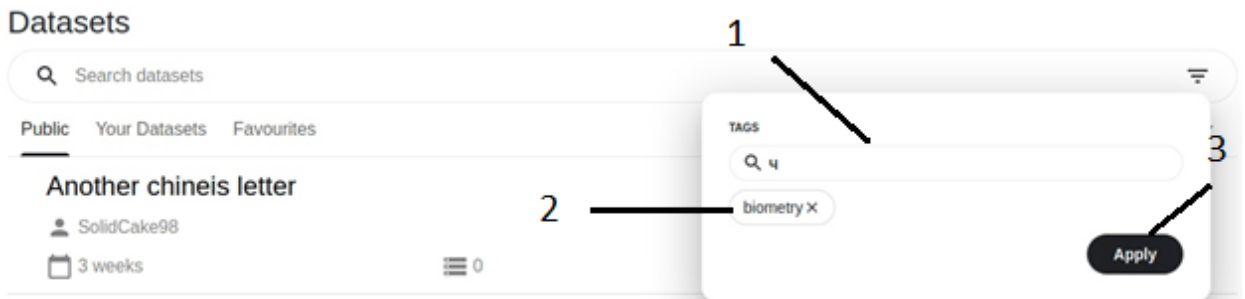


Рисунок 23 – Панель фильтрации по тэгам

Интерфейс панели фильтрации состоит из:

- 1 – строка поиска тэгов;
- 2 – список всех добавленных в фильтр тэгов;
- 3 – кнопка для применения фильтра.

При нажатии кнопка применения фильтра выводится отфильтрованный список наборов данных. Также при поиске тэгов будет отображен список возможных тэгов для выбора, что отображено на рисунке 24.

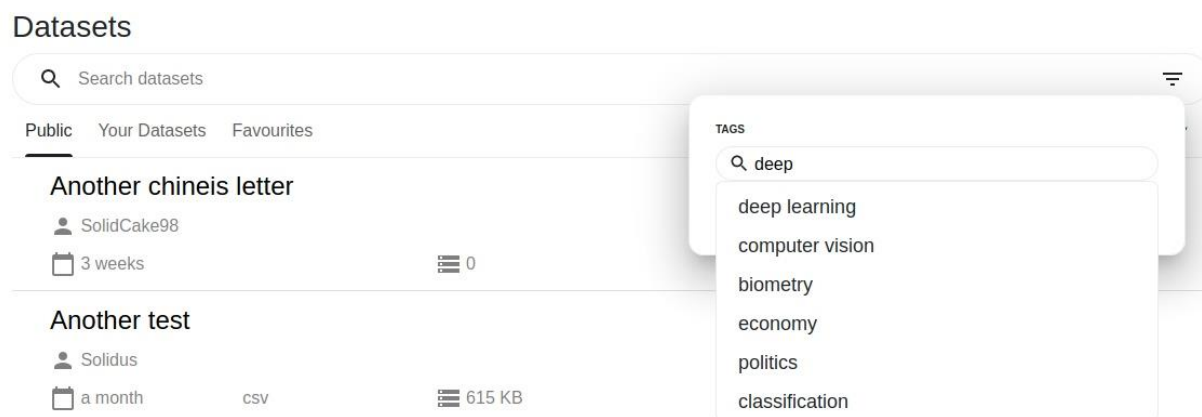


Рисунок 24 – Поиск тэгов для фильтрации

При нажатии кнопки для загрузки набора данных откроется модальное окно, представленное на рисунке 25.

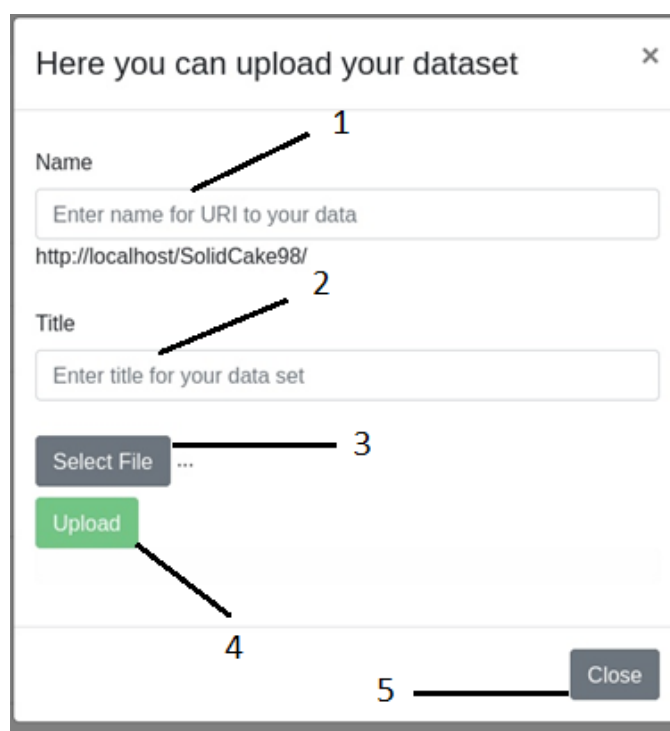


Рисунок 25 – Модальное окно для загрузки набора данных

Интерфейс модального окна включает в себя:

- 1 – поле ввода URL набора данных или его уникального названия;
- 2 – поле ввода его названия для отображения на главной странице;
- 3 – кнопка выбора файла для загрузки;
- 4 – кнопка загрузки набора данных;
- 5 – кнопка для закрытия модального окна.

Для возможности загрузки собственного набора данных необходимо пройти процесс авторизации и регистрации. На рисунке 26 представлено окно регистрации пользователя.

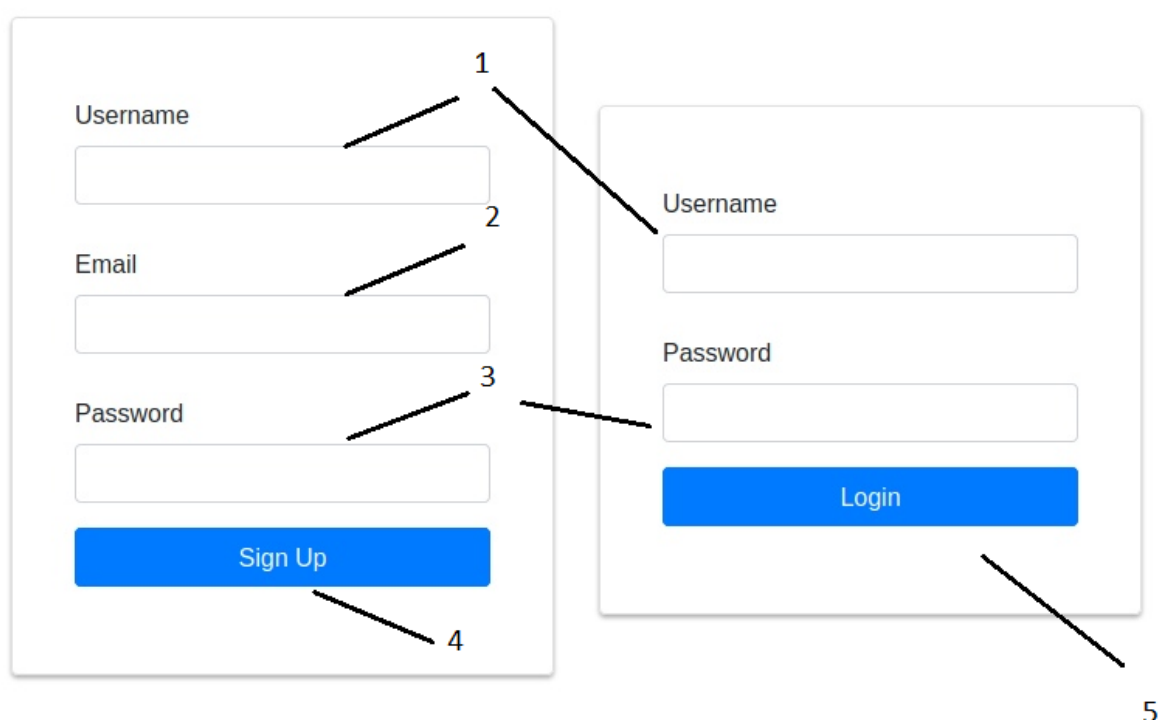


Рисунок 26 – Окна регистрации и авторизации

Основные элементы интерфейса данных окон, представленных на рисунке 26:

- 1 – поле ввода имени пользователя;
- 2 – поле ввода email пользователя;
- 3 – поле ввода пароля;
- 4 – кнопка регистрации;
- 5 – кнопка авторизации.

После авторизации пользователя будет открыта его домашняя страница с краткой информацией про пользователя и со списком его наборов данных. Окно авторизации представлено на рисунке 27.

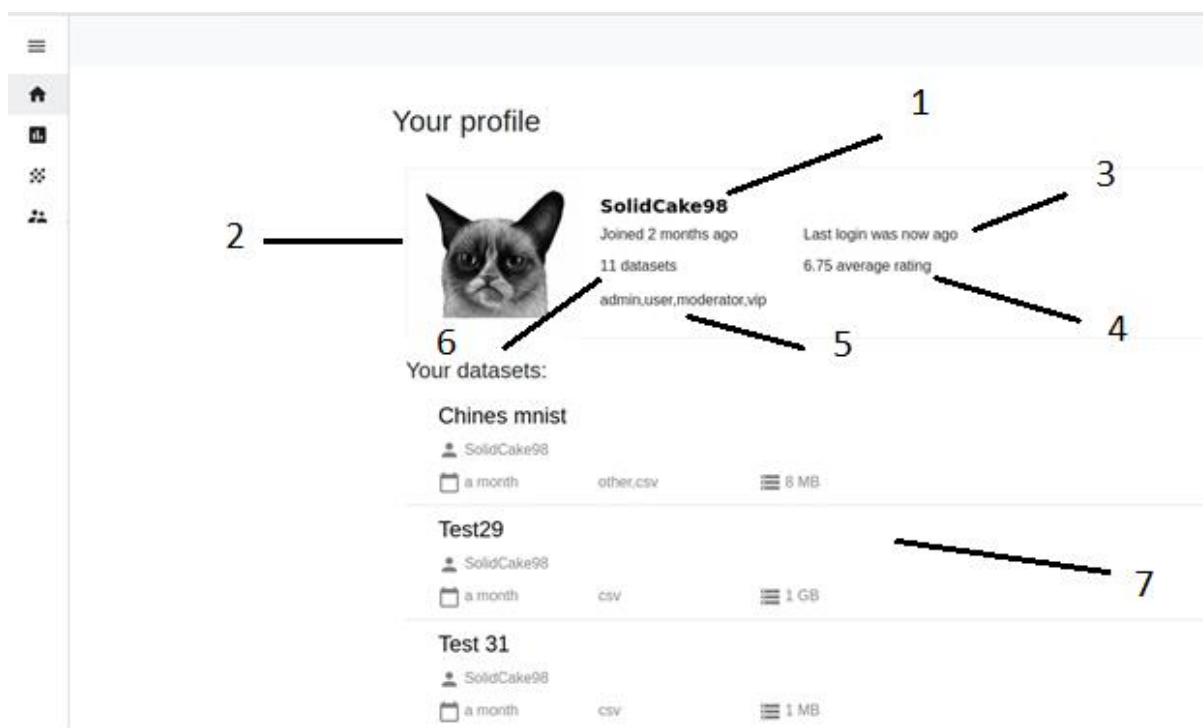


Рисунок 27 – Страница профиля пользователя

Основные элементы интерфейса данного окна, представленного на рисунке 27:

- 1 – имя пользователя;
- 2 – стандартный аватар пользователя;
- 3 – дата последней регистрации и авторизации;
- 4 – средний рейтинг его наборов данных;
- 5 – список всех ролей данного пользователя;
- 6 – количество загруженных наборов данных;
- 7 – список наборов данных.

При нажатии на любой из наборов данных откроется страница с подробной информацией о данном наборе. Интерфейс страницы с набором данных изображен на рисунке 28.

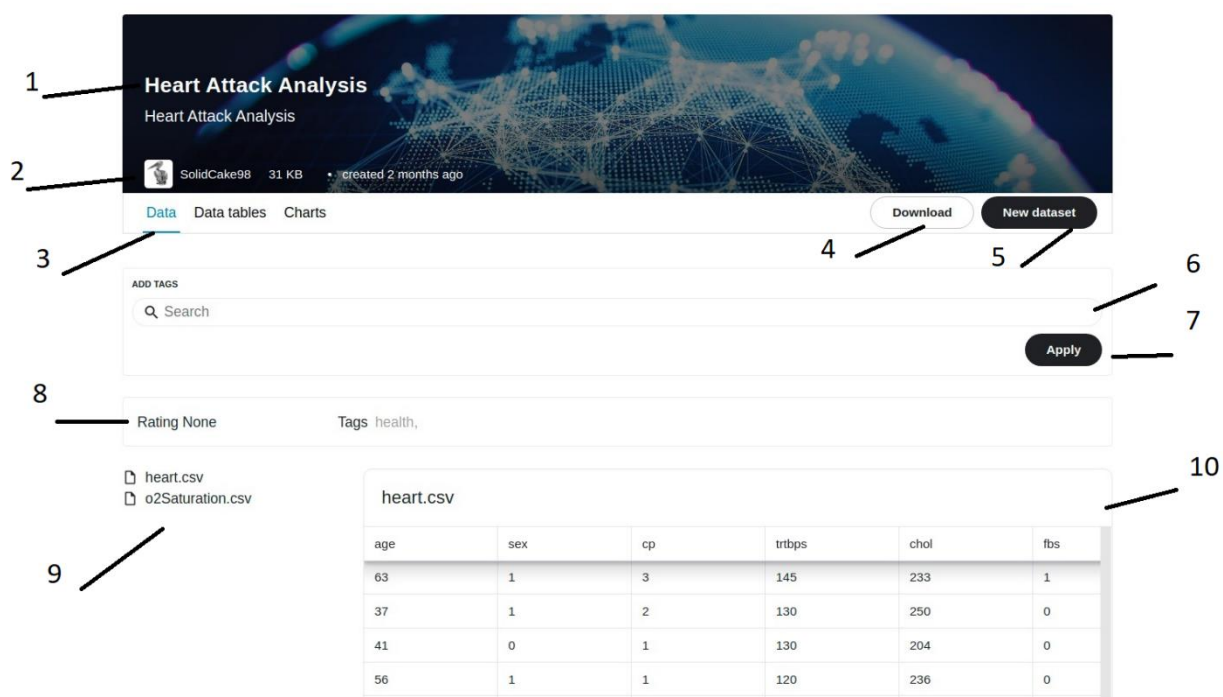


Рисунок 28 – Окно с возможностью скачивания и просмотра детальной информации о наборе данных

Основные элементы интерфейса окна набора данных, представленные на рисунке 28:

- 1 – имя набора данных;
- 2 – краткая мета информация о наборе данных (автор, объем, дата);
- 3 – панель с сущностями, связанными с данным набором данных;
- 4 – кнопка для скачивания данного набора данных;
- 5 – кнопка создания табличных данных;
- 6 – поле для добавления тэгов к набору (доступна только владельцу);
- 7 – кнопка для добавления тэгов;
- 8 – информация о рейтинге набора и тэгах;
- 9 – дерево файлов данного набора данных;
- 10 – окно предпросмотра файлов набора данных.

Пользователи имеют возможность оставления оценок и отзывов для набора данных. Интерфейс для отзывов представлен на рисунке 29.

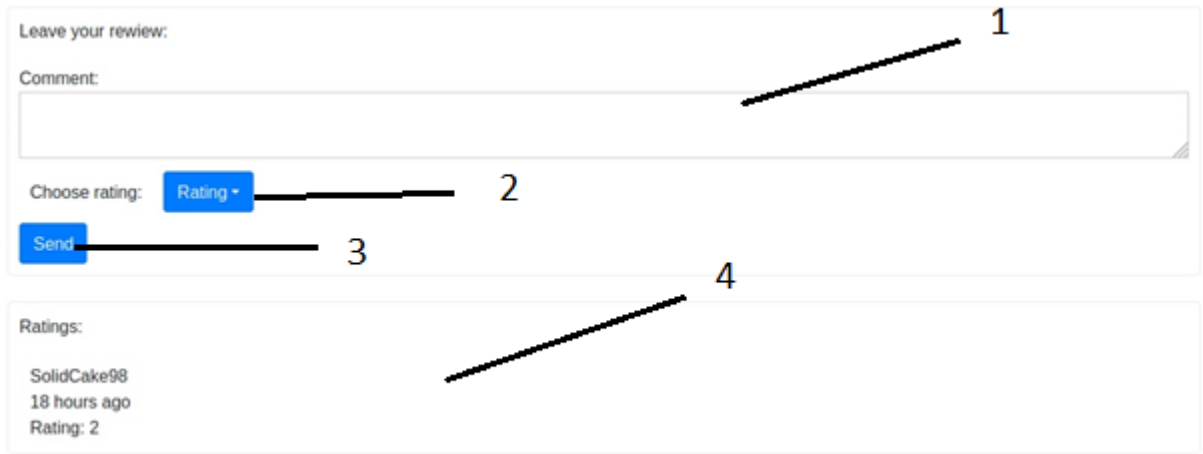


Рисунок 29 – Панель добавления оценки

Основные элементы интерфейса данной панели, представленные на рисунке 29:

- 1 – поле для написания отзыва;
- 2 – меню выбор оценки;
- 3 – кнопка отправки отзыва;
- 4 – список оставленных отзывов.

Для визуализации набора данных нужно изначально выбрать табличный файл, который будет в дальнейшем визуализирован. Модальное окно для создания табличного набора данных представлено на рисунке 30.

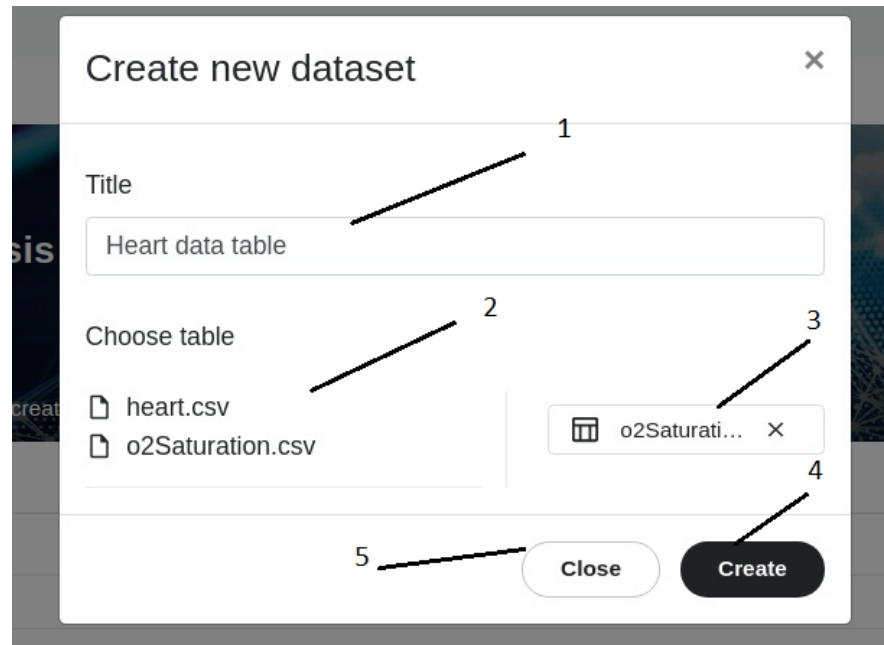


Рисунок 30 – Окно создания табличных данных

Основные элементы интерфейса окна создания табличных данных, представленные на рисунке 30:

- 1 – поле ввода имени табличных данных;
- 2 – список файлов типа csv для создания табличных данных;
- 3 – имя выбранного файла;
- 4 – кнопка для создания табличных данных;
- 5 – кнопка закрытия модального окна.

После создания табличных данных пользователь будет перенаправлен на страницу редактирования данной таблицы, представленную на рисунке 31.

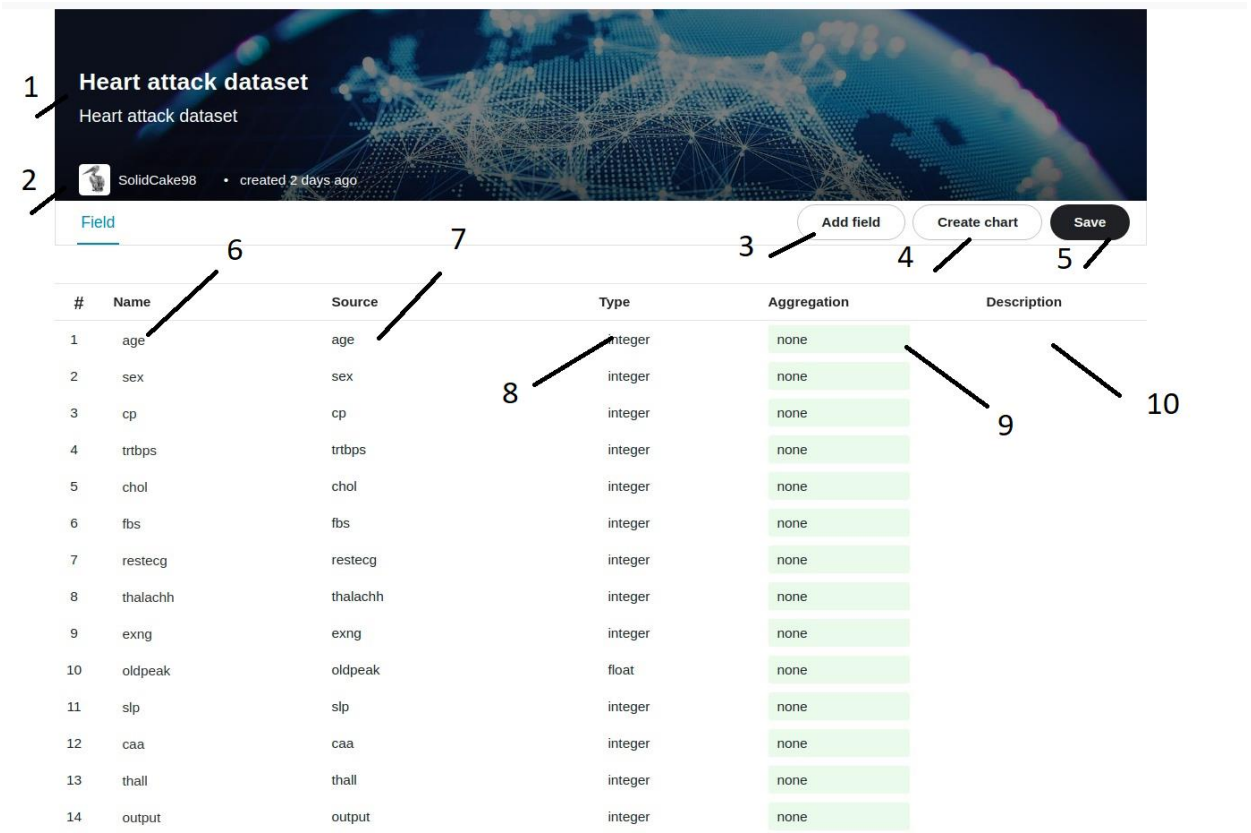


Рисунок 31 – Окно модификации табличных данных

Интерфейс данной страницы включает в себя:

- 1 – имя таблицы данных;
- 2 – автор и время создания данной таблицы;
- 3 – кнопка добавления нового столбца данных;
- 4 – кнопка создания графика для данной таблицы;
- 5 – кнопка сохранения изменений;
- 6 – поле изменения названия столбца таблицы;

- 7 – название столбца источника;
- 8 – поле типа данных столбца;
- 9 – тип агрегации над данным столбцом данных;
- 10 – поле описания столбца данных.

После внесения всех необходим изменений пользователь может визуализировать данную таблицу данных. Интерфейс для построения графиков представлен на рисунке 32.

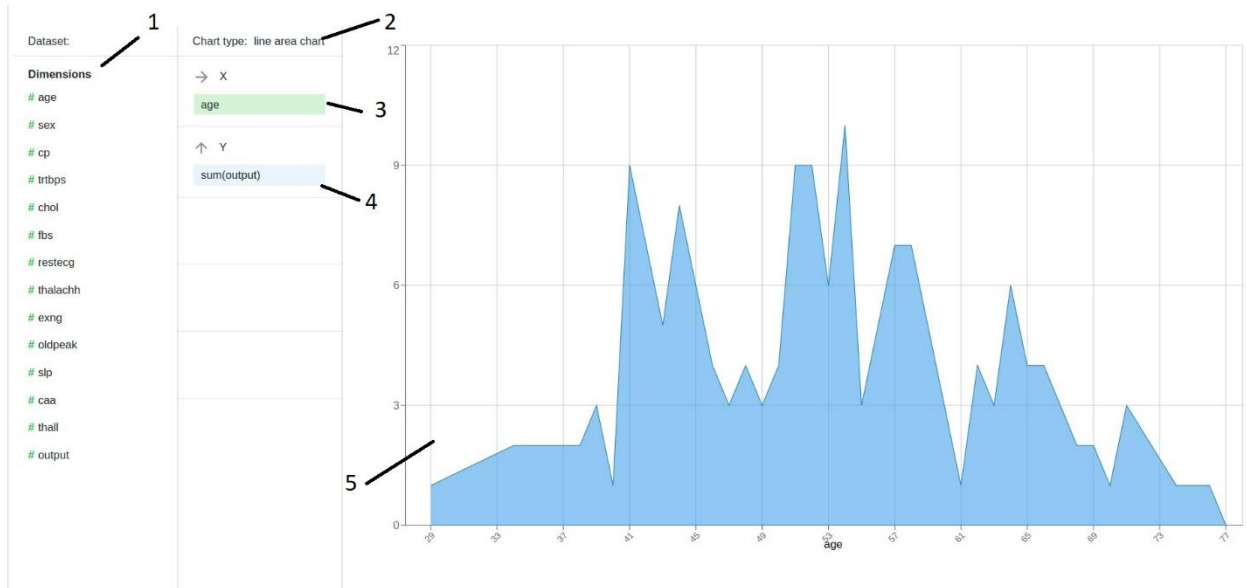


Рисунок 32 – Окно построения графиков

Интерфейс данной страницы включает в себя:

- 1 – список столбцов таблицы данных;
- 2 – поле выбора типа графика для визуализации;
- 3 – столбец данных для оси X;
- 4 – столбец данных для оси Y;
- 5 – построенный график для выбранных столбцов данных.

Также, помимо инструментария для построения графиков, загрузки данных и их поиска, для пользователей с ролей админа предоставлен доступ к административной панели, где пользователя может добавлять пользователей в определенные группы. Административная панель представлена на рисунке 33.

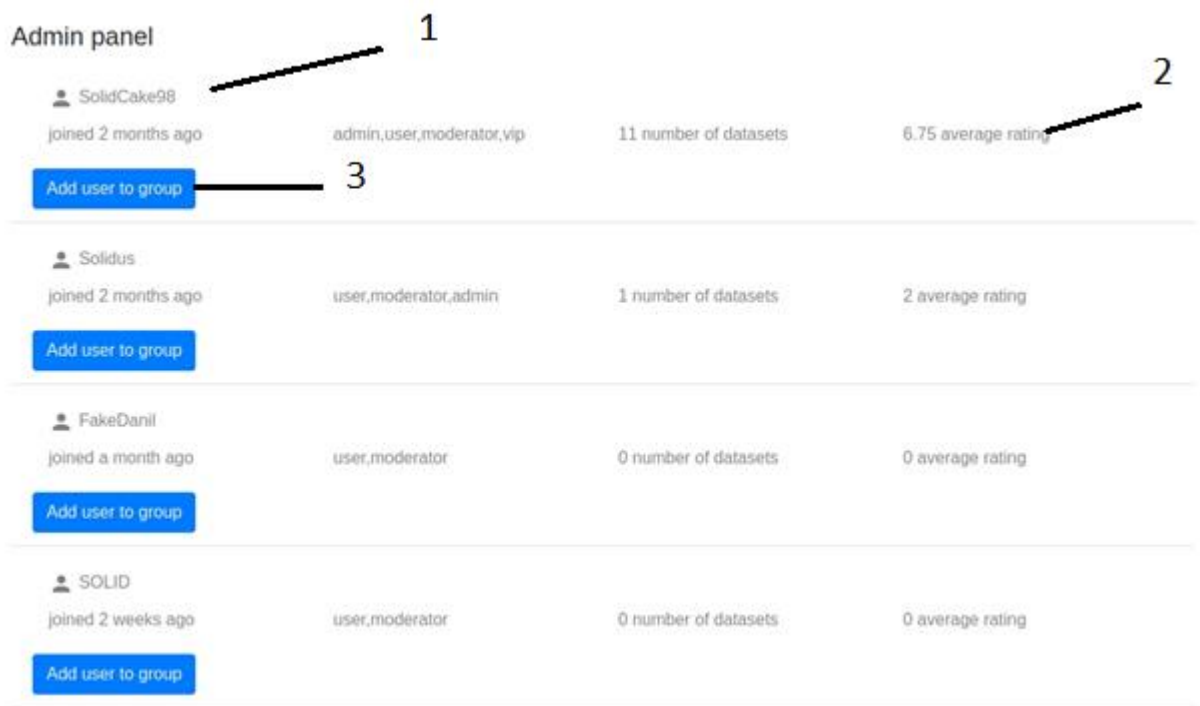


Рисунок 33 – Административная панель

Интерфейс данной страницы включает в себя:

- 1 – список всех пользователей;
- 2 – краткая информация о пользователе в списке;
- 3 – кнопка добавления пользователя в группу пользователей.

При нажатии кнопки пользователь будет открыто модальное окно для выбора группы, куда будет добавлен пользователь. Интерфейс данного окна представлен на рисунке 34.

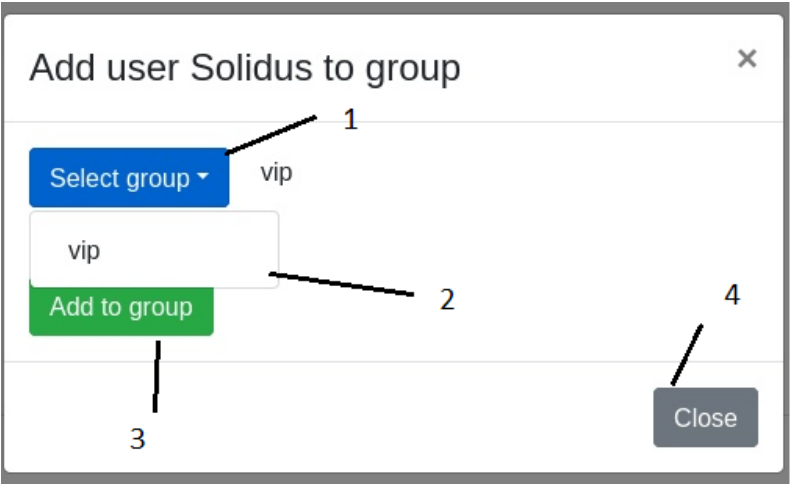


Рисунок 34 – Модальное окно для добавления пользователя к группе пользователей

Интерфейс данной страницы включает в себя:

- 1 – кнопка показа списка доступных групп;
- 2 – список групп пользователей, куда можно добавить пользователя;
- 3 – кнопка добавления пользователя в группу пользователей;
- 4 – кнопка закрытия модального окна.

Каждая страница и панель содержит в себе множество React компонентов. Клиентское приложение связывается с сервером приложений по средствам библиотеки `axios`, предоставляющий удобный механизм для составления `http` запросов.

2.5 Проектирование клиентского приложения

Для написания клиентского приложения был выбран фреймворк `React.js`. `React` приложение использует компонентный подход. Каждый компонент представлен в виде отдельного класса. В `React.js` есть возможность создавать инкапсулированные компоненты, которые управляют своим состоянием, а затем компоновать их для создания сложных пользовательских интерфейсов. Поскольку логика компонента написана на `JavaScript` вместо использования шаблонов, есть возможность легко передавать различного типа данных через своё приложение и сохранять состояние вне `DOM`.

`React`-компоненты реализуют метод `render()`, который принимает входные данные и возвращает что-то для вывода. Используется XML-подобный синтаксис под названием `JSX`. Входные данные, передаваемые в компонент, доступны в `render()` через `this.props`. Помимо входных данных (доступных через `this.props`), компонент поддерживает внутренние данные состояния (доступные через `this.state`). Когда данные состояния компонента изменяются, `React` ещё раз вызовет `render()` и обновит отрендеренную разметку [22].

На рисунке 35 представлена диаграмма классов клиентского приложения.

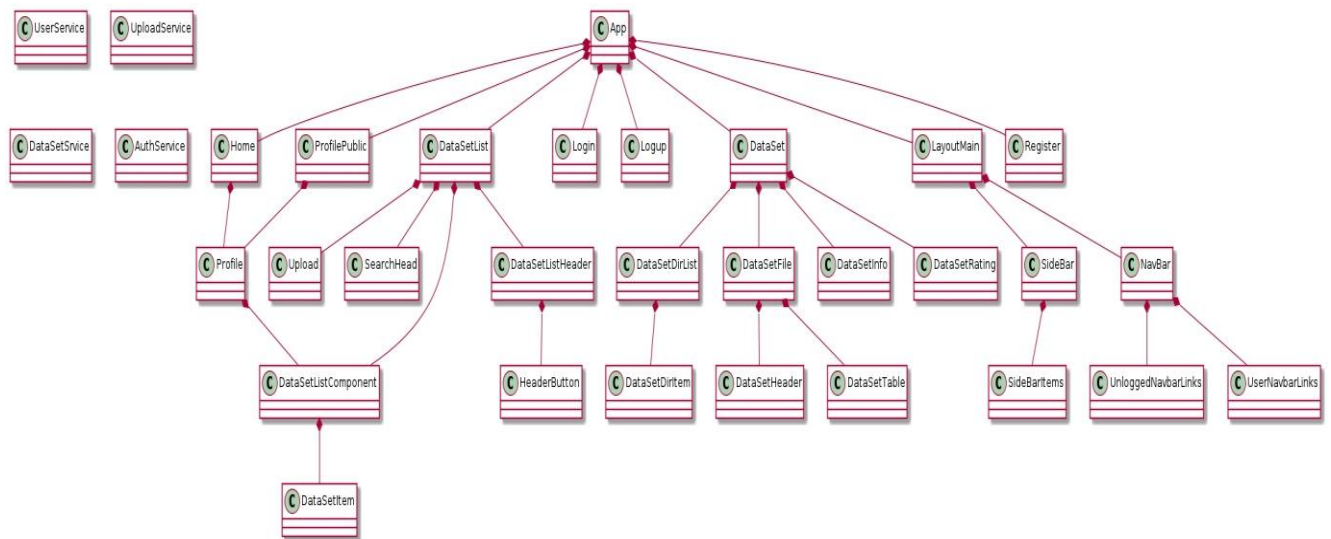


Рисунок 35 – Диаграмма классов клиентского приложения

Можно выделить два типа классов:

- классы сервисы, чья задача отправлять запросы на сервер приложений;
- классы компоненты, которые содержат в себе логику работы и описание UI.

3 Реализация программного обеспечения сервиса

Программные средства сервиса хранения, поиска и визуализации открытых данных реализован на языке Python, с помощью фреймворка для создания web-приложений Flask. Компоненты данного сервиса представлены на рисунке 36.

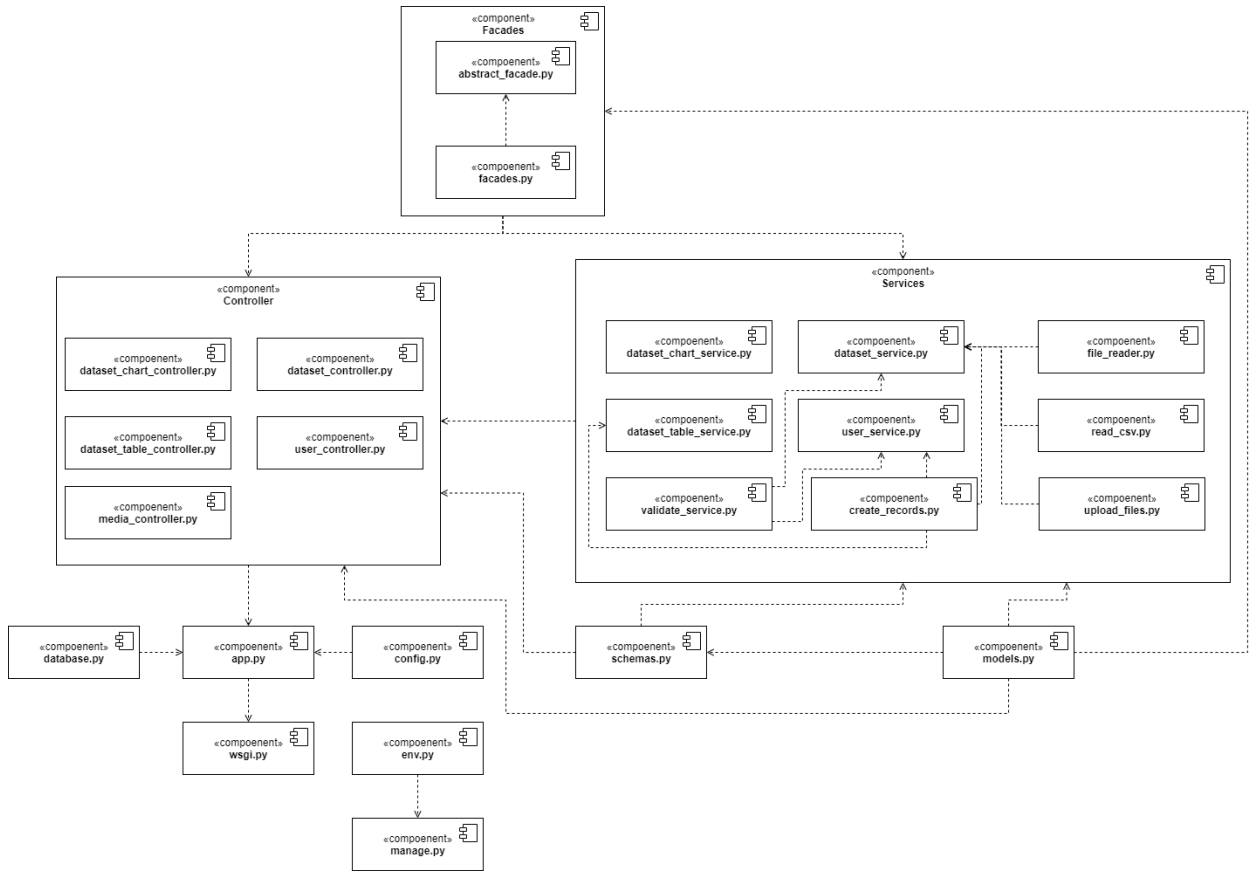


Рисунок 36 – Диаграмма компонентов

Таблица 7 – Описание компонентов системы

№	Название	Описание
1	app.py	Является точкой входа, содержит все сервисы приложения.
2	database.py	Описание подключения к БД и вспомогательные классы для работы с триггерами и функциями при миграции.
3	config.py	Конфигурация приложения.
4	env.py	Скрипт установки переменных окружения приложения.
5	manage.py	Скрипт для команд сборки, запуска, тестирования приложения и миграции БД.
6	schema.py	Исходный код с классами для сериализации моделей БД.
7	models.py	Описание таблиц БД с помощью языка ORM.

Продолжение таблицы 7

№	Название	Описание
8	dataset_chart_controller.py	Исходный код с сервисами визуализации данных.
9	dataset_controller.py	Исходный код с сервисами для работы с наборами открытых данных.
10	dataset_table_controller.py	Исходный код с сервисами для работы с табличными данными.
11	user_controller.py	Исходный код с сервисами для работы с пользователями.
12	media_controller.py	Исходный код с сервисами для работы с медиа контентом.
13	dataset_chart_service.py	Бизнес логика сервисов для визуализации данных.
14	dataset_service.py	Бизнес логика сервисов для создания, загрузки и работы с наборами открытых данных.
15	dataset_table_service.py	Бизнес логика сервисов для работы с табличными данными.
16	user_service.py	Бизнес логика сервисов, связанных с пользователями.
17	create_records.py	Исходный код с классами для создания записей в БД.
18	file_reader.py	Исходный код с классами для чтения файлов
19	read_csv.py	Исходный код с классами для чтения табличных данных.
20	upload_files.py	Исходный код с классами для сохранения наборов данных на сервере.
21	abstract_facade.py	Исходный код с классом для базовых запросов к БД.
22	facades.py	Исходный код с классами для специальных запросов к БД.

БД была развернута с помощью механизма миграции. Миграция – обновление структуры базы данных от одной версии до другой (обычно более новой).

Перед первичной миграцией необходимо создать БД, к которой подключится сервер приложений. Для данной цели был написан скрипт на языке Python.

Механизм миграций необходим для версионирования БД и поддержания ее консистентности. То есть этот механизм схож с работой систем контроля версий, только в отличие от контроля версий он синхронизирует структуру БД. В данной работе использовалась библиотека Alembic для миграции БД.

Результат развертывания БД представлен на рисунке 37.

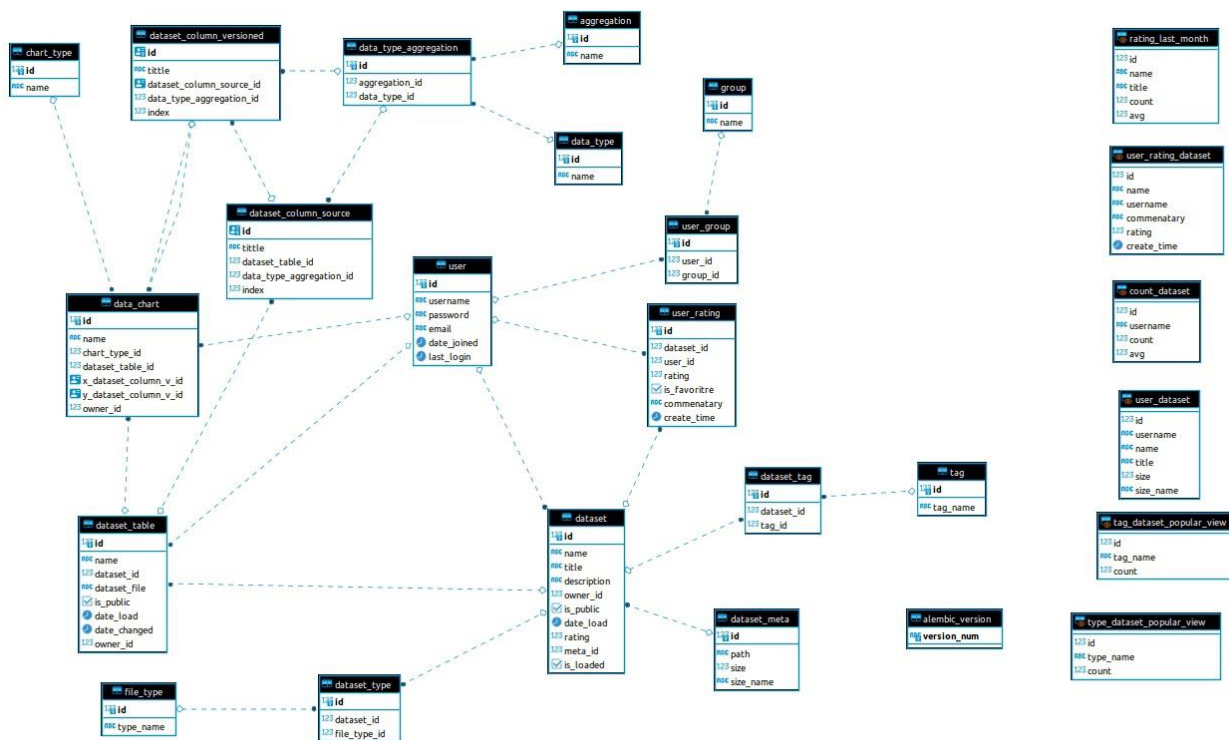


Рисунок 37 – Результат развертывания БД

Так как сервис представляет собой REST API приложение, то необходимо представить спецификацию всех сервисов данного приложения. К каждому сервису можно обратиться по специализированному URI, а основным средством обмена информацией является JSON объекты. Спецификация сервисов представлена в таблице 8.

Таблица 8 – Спецификация REST сервисов web приложения

№	Назначение сервиса	URI сервиса	Класс контроллера
1	Сервис регистрации пользователя	/api/v1/jwt/register	UserRegistration
2	Сервис авторизации пользователя	/api/v1/jwt/auth	UserAuthorization
3	Обновление токена авторизации	/api/v1/jwt/refresh	TokenRefresh
4	Формирование списка пользователей	/api/v1/user	UserListController
5	Отправка информации о конкретном пользователе	/api/v1/user/<user_id>	UserController
6	Формирование профиля пользователя	/api/v1/user/profile/<user_id>	UserProfileController
7	Формирование списка профилей	/api/v1/user/profile	UserProfileListController

Продолжение таблицы 8

№	Назначение сервиса	URI сервиса	Класс контроллера
8	Отправка списка групп пользователей за исключением тех, в которых данный пользователь состоит	/api/v1/user/groups/excluded/<user_id>	UserGroupExcludedController
9	Добавление пользователя в группу пользователей	/api/v1/user/group	UserGroupAddController
10	Отправка стандартного аватара пользователя	/api/v1/user/media/default	UserDefaultAvatar
11	Скачивание пользовательского набора данных	/api/v1/dataset/download/<user>/<data>	DataSetDownloadController
12	Загрузка пользовательского набора данных	/api/v1/dataset/upload	DataSetUploadController
13	Формирование списка пользовательских наборов данных	/api/v1/dataset	DataSetListController
14	Сортировка списка пользовательских наборов данных по дате	/api/v1/dataset/new	DataSetListNewController
15	Сортировка списка пользовательских наборов данных по популярности	/api/v1/dataset/popular	DataSetListPopularController
16	Отправка наборов данных владельца	/api/v1/dataset/owner	DataSetListUserController
17	Поиск наборов данных по заголовку	/api/v1/dataset/search	DataSetListWordTitleController
18	Поиск наборов данных по тэгам	/api/v1/dataset/search/tag	DataSetListTagController
19	Поиск тэгов наборов данных по имени	/api/v1/dataset/tag	TagListWordNameController
20	Отправка набора данных определенного пользователя	/api/v1/dataset/<user>/<data>	DataSetController
21	Отправка файловой структуры набора данных	/api/v1/dataset/dir	DataSetDirReadController
22	Отправка информации и содержание определенного файла в наборе данных	/api/v1/dataset/file	DataSetFileReadController

Окончание таблицы 8

№	Назначение сервиса	URI сервиса	Класс контроллера
23	Отправка файла изображения, находящегося в наборе	/api/v1/dataset/img/<path:path>	DataSetImageController
24	Добавление тэгов к набору данных	/api/v1/dataset/add_tags/<name>	DataSetAddTags
25	Добавление оценки	/api/v1/dataset/rating	DataSetUserRatingController
26	Создание табличных данных	/api/v1/dataset_table	DataSetTableCreateController
27	Формирование списка всех табличных данных для заданного набора	/api/v1/dataset_table/list/<int:data_id>	DataSetTableListController
28	Формирование информации о таблице данных	/api/v1/dataset_table/<int:table_id>	DataSetTableController
29	Формирование списка всех доступных типов данных и агрегаций	/api/v1/dataset_table/types	DataTypeAggregationController
30	Формирование данных из таблицы для дальнейшей визуализации	/api/v1/dataset_chart/calculate	DataChartController
31	Формирование списка всех типов графиков	/api/v1/dataset_chart/type	ChartTypeController

Далее на рисунке 38 представлена диаграмма развертывания приложения, которая изображает основные компоненты системы.

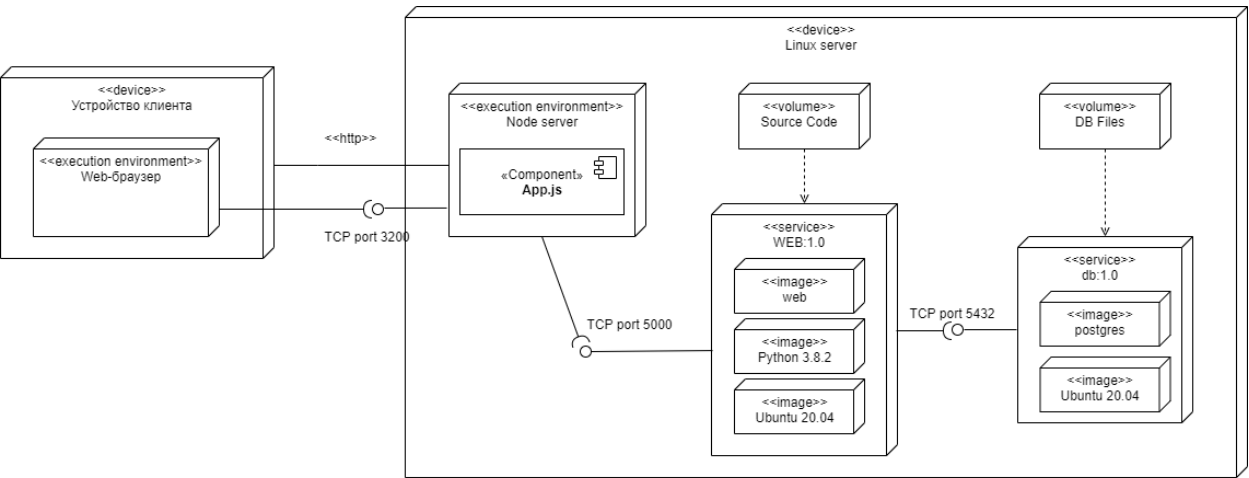


Рисунок 38 – Диаграмма развертывания

На диаграмме видно, что приложение соответствует трех уровневой архитектуре. Сервер приложений и сервер БД обернуты в docker контейнеры, что позволяет в дальнейшем беспрепятственно переносить данное приложение на другие платформы.

Docker позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему, а также предоставляет среду по управлению контейнерами. За описание запуска сервера БД отвечает сервис db. Образ бд скачивается из репозитория docker-hub. Для связи с данным контейнером был открыт порт 5432. А за описание запуска приложения отвечает сервис web. Для взаимодействия с данным контейнером был открыт порт 5000.

Клиентское приложение работает на отдельном сервере, но при этом можно скомпилировать данное приложение и избавиться от данного компонента системы, соединив его с сервером приложений.

4 Контроль качества программного обеспечения

4.1 Функциональное тестирование

Функциональное тестирование методом черного ящика – это тестирование функциональных возможностей ПО, в условиях, когда внутренняя структура и реализация неизвестна.

Произведем функциональное тестирование загрузки набора данных на сервер. Пользователь должен иметь возможность загрузить набор данных в виде любого файла или zip архива. Имя файла или архива при этом должно содержать только разрешенные символы. Также набору данных присваивается свой псевдоним, который тоже должен пройти валидацию. Не могут быть загружены наборы данных под одним псевдонимом. В таблице 9 представлены классы эквивалентности.

Таблица 9 – Классы эквивалентности

Входные условия	Правильные классы эквивалентности	Неправильные классы эквивалентности
Длина псевдонима	≥ 6 и ≤ 40	< 6 или > 40
Псевдоним	Состоит из букв латинского алфавита, цифр, нижнего подчеркивания или тире	Содержит другие символы
Занятость псевдонима	Псевдоним свободен	Псевдоним занят
Имя файла или архива	Состоит из букв латинского алфавита, цифр, нижнего подчеркивания, точки или тире	Содержит другие символы
Первый символ в имени файла или архива	Нет точки в начале	Точка в начале

Определим граничные условия. Длина псевдонима: $n \in [6, 40]$. Следовательно, длины 6 и 40 находятся на границах допустимых значений. Эти значения следует проверить. Также имеет смысл проверить ближайшие неверные значения 5 и 41.

Граничные условия для состава символов в имени файла и псевдонима размыты. Следует проверить работу программы на корректных последовательностях символов в имени файла и псевдонима, так и последовательности, содержащие некорректные символы.

Для признака занятости псевдонима существует только два возможных варианта. Либо псевдоним занят, либо свободен. Следует протестировать оба варианта.

Для того, чтобы построить эффективно-результативные тесты, можно воспользоваться методом функциональных диаграмм. При построении тестов с использованием данного метода в спецификации определяются причины и следствия. Причина – входное условие, а следствие – выход системы. Далее спецификация преобразуется в управляющий граф.

Для процесса загрузки набора данных была построена функциональная диаграмма, представленная на рисунке 39.

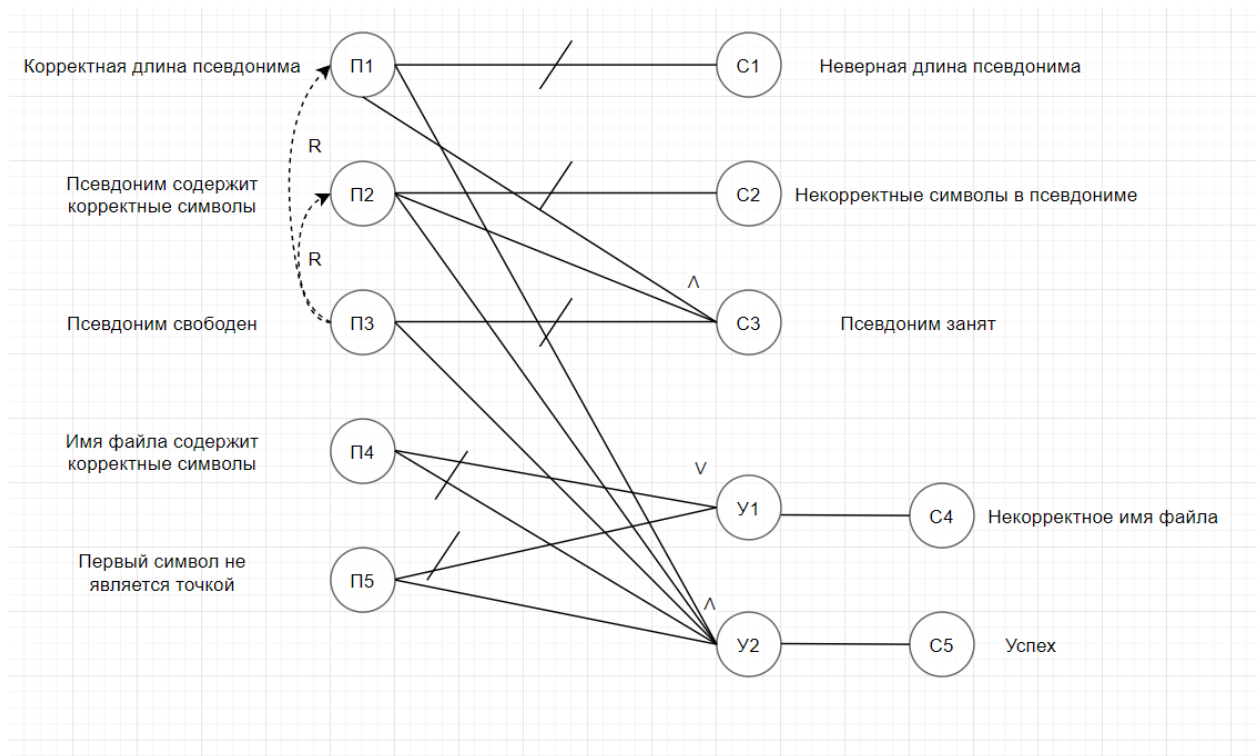


Рисунок 39 – Функциональная диаграмма

Причины:

- корректная длина псевдонима;
- псевдоним содержит корректные символы;
- псевдоним свободен;
- имя файла содержит корректные символы;
- имя файла не начинается с точки.

Следствия:

- сообщение об ошибке «Неверная длина псевдонима»;
- сообщение об ошибке «Некорректные символы в псевдониме»;
- сообщение об ошибке «Псевдоним занят»;
- сообщение об ошибке «Некорректное имя файла»;
- успех.

На основании функциональной диаграммы была построена таблица решений. Таблица решений представлена в таблице 10.

Таблица 10 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
П1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0
П2	0	0	1	0	0	1	1	1	1	1	1	1	0	0	1	1	0
П3	-	-	-	-	-	0	1	1	1	0	0	1	-	-	-	-	-
П4	0	0	0	1	0	0	0	1	0	1	1	1	0	1	1	1	1
П5	0	0	0	0	1	0	0	0	1	1	0	1	1	1	1	0	1
С1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
С2	1	1	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1
С3	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
С4	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0
С5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

В таблице 11 представлен набор тестов, составленные на основе таблицы решений.

Таблица 11 – Набор тестов

№	Входные данные	Ожидаемый результат	Результат
1	Псевдоним: Co! Файл: .re!.csv	(C1, C2, C4)	Тест пройден (рисунок В.1)
2	Псевдоним: Corona! Файл: .re@.csv	(C2, C4)	Тест пройден
3	Псевдоним: Cor Файл: .re@.csv	(C1, C4)	Тест пройден
4	Псевдоним: Co\$ Файл: .re.csv	(C1, C2, C4)	Тест пройден
5	Псевдоним: Co\$ Файл: re!.csv	(C1, C2, C4)	Тест пройден
6	Псевдоним: HeartAttack Файл: .re!.csv	(C3, C4)	Тест пройден (рисунок В.2)
7	Псевдоним: CoronaVirus Файл: .re!.csv	(C4)	Тест пройден (рисунок В.3)

Продолжение таблицы 11

№	Входные данные	Ожидаемый результат	Результат
8	Псевдоним: CoronaVirus Файл: .re2021.csv	(C4)	Тест пройден
9	Псевдоним: CoronaVirus Файл: re2021"".csv	(C4)	Тест пройден
10	Псевдоним: HeartAttack Файл: re2021.csv	(C3)	Тест пройден (рисунок В.4)
11	Псевдоним: HeartAttack Файл: .re2021.csv	(C3, C4)	Тест пройден
12	Псевдоним: CoronaVirus Файл: re2021.csv	(C5)	Тест пройден (рисунок В.5)
13	Псевдоним: Corona! Файл: re@.csv	(C2, C4)	Тест пройден
14	Псевдоним: Corona! Файл: re.csv	(C2)	Тест пройден
15	Псевдоним: Cor Файл: re.csv	(C1)	Тест пройден
16	Псевдоним: Cor Файл: .re.csv	(C1, C4)	Тест пройден
17	Псевдоним: Co! Файл: re.csv	(C1, C2)	Тест пройден

Функциональное тестирование показало, что при загрузке набора данных программа работает правильно. Результаты тестирования представлены в приложении В.

4.2 Модульное тестирование

Модульное тестирование позволяет проверить в автоматическом режиме корректность работы отдельных модулей программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Также при модульном тестировании необходимо знать метрику покрытия кода. Она показывает процент исходного кода программы, который был выполнен в процессе тестирования.

Для проведения модульного тестирования была выбрана библиотека `pytest`. Плюсом данной библиотеки является то, что в ней присутствует механизм

создания соответствующего окружения для проведения тестирования, что облегчает, например, работу с БД.

Было проведено модульное тестирование подсистемы авторизации и регистрации (таблица 12).

Таблица 12 – Описание модульных тестов

Тестовый метод	Описание	Ожидаемый результат
test_register_user	Регистрация при всех валидных данных: {"username": "danil", "email": "some.email@server.com", "password": "1234567f"}	Вернется код успеха 200, пользователь будет добавлен в БД
test_register_not_valid_password_user	Регистрация при не валидном пароле: {"username": "danil", "email": "some.email@server.com", "password": "1234567"}	Вернется код неудачи 400, вернется сообщение об ошибке {'error': 'Password must be at least 8 characters, contain 1 number and 1 letter'}
test_register_not_valid_username_user	Регистрация при не валидном имени пользователя: {"username": "12345", "email": "some.email@server.com", "password": "1234567f" }	Вернется код неудачи 400, вернется сообщение об ошибке {'error': 'Username must be at least 5 characters and contain 1 letter'}
test_register_user_is_already_exist	Регистрация в случае, если пользователь с таким именем уже был зарегистрирован. Повторно регистрируется пользователь с данными: {"username": "danil", "email": "some.email@server.com", "password": "1234567f"}	Вернется код неудачи 400, вернется сообщение об ошибке {'error': 'User danil already exist'}
test_register_user_email_is_already_exist	Регистрация в случае, если пользователь с таким email'ом уже был зарегистрирован: {"username": "daniil", "email": "some.email@server.com", "password": "1234567f"}	Вернется код неудачи 400, вернется сообщение об ошибке {'error': 'Email some.email@server.com already registred'}
test_auth_user	Авторизация при всех валидных данных: {"username": "some.email@server.com", "password": "1234567f" }	Вернется код успеха 200, будет возвращен токен авторизации
test_auth_user_not_vaild_email	Авторизация при неправильном вводе email'a: {"username": "some.emal@server.com", "password": "1234567f" }	Вернется код неудачи 400 и сообщение: {'error': 'The username or password provided is incorrect.'}
test_auth_not_val_id_name	Авторизация при неправильном вводе имени {"username": "daniil", "password": "1234567f"}	Вернется код неудачи 400 и сообщение: {'error': 'The username or password provided is incorrect.'}
test_auth_not_val_id_password	Авторизация при неправильном вводе пароля {"username": "danil", "password": "1234" }	Вернется код неудачи 400 и сообщение: {'error': 'The username or password provided is incorrect.'}

Исходный код модульного тестирования представлен в приложении Г. Все тесты были пройдены успешно (рисунок В.6). При этом покрытие модуля авторизации и регистрации составило 85%, что говорит о том, что данный модуль работает корректно (рисунок В.7).

4.3 Метрики кода

В качестве первой вычисленной метрики будут вычислены количественные метрики. Данный тип метрик был вычислен с использованием утилиты cloc, которая подсчитывает количество пустых строк, количество комментариев, число строк, содержащих исходный код для функций (классов, файлов). Результаты представлены на рисунке 40.

```

62 text files.
61 unique files.
17 files ignored.

github.com/AlDanial/cloc v 1.82 T=0.68 s (76.0 files/s, 5380.9 lines/s)
-----
Language             files      blank      comment      code
-----
Python                42         862         470         2050
JSON                  4           2           0          115
INI                   1          20           0           65
YAML                  2           4           2           40
Mako                  1           7           0           17
Markdown              1           7           0            9
Dockerfile            1           5           1            7
-----
SUM:                  52         907         473         2303
-----

```

Рисунок 40 – Подсчитанная метрика количества строк кода

Также были подсчитаны метрики устойчивости и абстрактности. Метрика устойчивости для компоненты отражает количественную меру работы, которую необходимо произвести при изменении других зависимых компонент.

Для вычисления позиционной устойчивости кода используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где I – мера неустойчивости, которая изменяется в диапазоне от $[0, 1]$ (если $I = 0$, то компонент максимально устойчив, если $I = 1$, то компонент максимально неустойчив);

C_e – количество классов вне данного компонента, зависящих от классов внутри компонента;

C_a – количество классов внутри данного компонента, зависящих от классов вне компонента.

В созданном приложении можно выделить 6 логических компонент:

- слой Controller;
- компонент Services;
- компонент валидации данных;
- компонент работы с БД (фасад);
- компонент моделей БД;
- компонент создания объектов моделей БД.

Для каждой компоненты была просчитана величина I – метрика неустойчивости.

- $I_1 = \frac{0}{0+3} = 0;$
- $I_2 = \frac{3}{3+3} = 0.5;$
- $I_3 = \frac{1}{1+1} = 0.5;$
- $I_4 = \frac{4}{4+1} = 0.8;$
- $I_5 = \frac{4}{4+0} = 1;$
- $I_6 = \frac{1}{1+1} = 0.5.$

Метрика A , являющаяся мерой абстрактности компонента, равна отношению количества абстрактных классов к общему количеству классов в компоненте.

Абстрактностью A называется величина

$$A = \frac{Na}{Nc},$$

где Nc – количество классов в компоненте;

Na – количество абстрактных классов в нем же.

Метрика A изменяется от 0 до 1, причем 0 означает, что в компоненте вообще нет абстрактных классов, а 1 – что компонент содержит только абстрактные классы.

- $A_1 = \frac{0}{3} = 0;$
- $A_2 = \frac{2}{7} = 0.28;$
- $A_3 = \frac{1}{6} = 0.16;$
- $A_4 = \frac{1}{4} = 0.25;$
- $A_5 = \frac{1}{3} = 0.33;$
- $A_6 = \frac{1}{2} = 0.5.$

Вычислив меру устойчивости компоненты и ее абстрактность можно отобразить их на графике. На вертикальной оси необходимо отложить меру абстрактности A , а на горизонтальной оси – меру неустойчивости I .

Компонент, который располагаются в верхнем левом углу или в нижнем правом являются максимально качественным. На практике часто случается, так что полученная точка лежит на диагонали между точкой (1,0) и точкой (0,1). Такая диагональ называется главная последовательность (рисунок 4).

Для вычисления количественной оценки используется формула вычисления расстояния до главной последовательности:

$$D = \left| \frac{(A + I - 1)}{\sqrt{2}} \right|,$$

где A – мера абстрактности;

I – мера неустойчивости.

Чем ближе точка располагается к главной последовательности, тем лучше. На рисунке 41 главная последовательность и зоны неприятностей, где компонент является слишком неустойчивым, но при это не является абстрактным и наоборот.

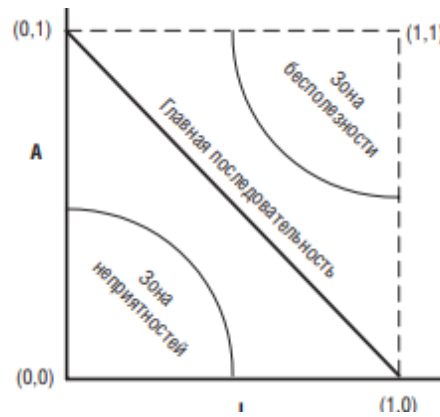


Рисунок 41 – Зоны исключения

Был построен соответствующий график для логических компонент приложения (рисунок 42).

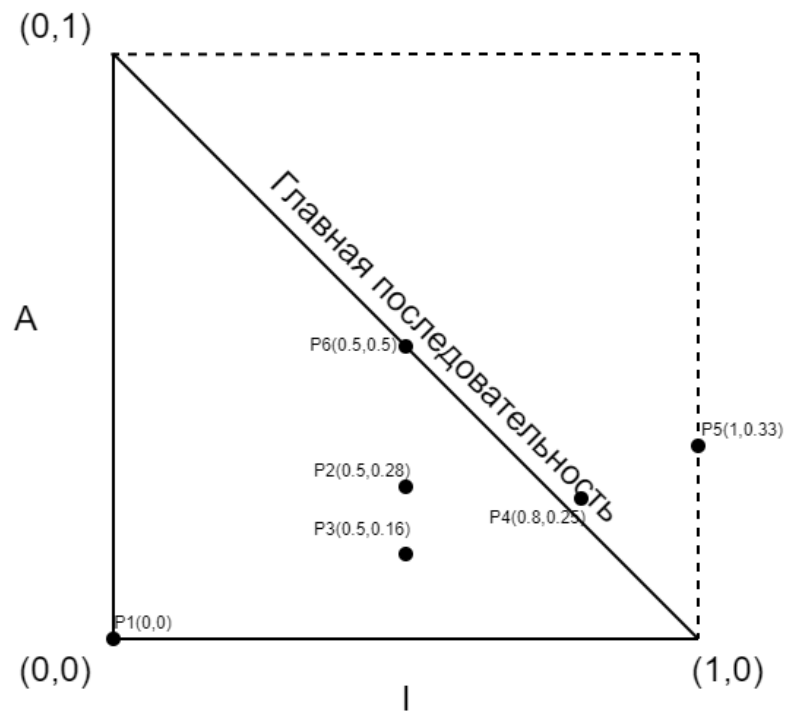


Рисунок 42 – Отображение метрик на графике

Было вычислено расстояние до главной последовательности:

- $D_1 = \left| \frac{0+0-1}{\sqrt{2}} \right| = 0.707;$
- $D_2 = \left| \frac{0.28+0.5-1}{\sqrt{2}} \right| = 0.155;$
- $D_3 = \left| \frac{0.16+0.5-1}{\sqrt{2}} \right| = 0.24;$
- $D_4 = \left| \frac{0.25+0.8-1}{\sqrt{2}} \right| = 0.035;$

$$- D_5 = \left| \frac{0.33+1-1}{\sqrt{2}} \right| = 0.23;$$

$$- D_5 = \left| \frac{0.5+0.5-1}{\sqrt{2}} \right| = 0.$$

Первая компонента достаточно удалена от главной последовательности и находится в зоне неприятностей. Это обусловлено тем, что данная компонента является слоем контроллера и его задача только вызывать методы бизнес логики, поэтому этот компонент имеет устойчивость. Нулевая абстрактность обусловлена малым количеством классов в данной компоненте.

Остальные компоненты лежат достаточно близко к главной последовательности. Это значит, что отношение абстрактности к устойчивости этих компонент является правильным.

4.4 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project [23]. Проект был начат 14 апреля 2021 года.

При планировании разработки было выделено 27 задач. Список задач и их описание представлено на рисунке 43.

	Режим задачи	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	✓	Начало проекта	0 дней	Ср 14.04.21	Ср 14.04.21		
2	✓	Анализ требований	9 дней	Ср 14.04.21	Вт 27.04.21		
3	✓	Анализ предметной области	2 дней	Ср 14.04.21	Пт 16.04.21	1	Интернет[100,00 Р];Аналитик[70%]
4	✓	Сравнительный анализ программ аналогов	2 дней	Пн 19.04.21	Ср 21.04.21	3	Аналитик[30%];Программист 1;Интернет[100,00 Р]
5	✓	Анализ способов визуализации	1 день	Пн 19.04.21	Пн 19.04.21	3	Программист 2
6	✓	Анализ методов понижения размерности данных	2 дней	Вт 20.04.21	Ср 21.04.21	5	Программист 2
7	✓	Постановка задачи	1 день	Чт 22.04.21	Чт 22.04.21	4;6	Аналитик
8	✓	Анализ функциональных требований	2 дней	Пт 23.04.21	Пн 26.04.21	7	Аналитик[70%];Программист 2[30%]
9	✓	Анализ требований и постановка задачи завершена	0 дней	Вт 27.04.21	Вт 27.04.21	8	
10	✓	Проектирование программных средств	9 дней	Вт 27.04.21	Пн 10.05.21		
11	✓	Проектирование структур данных сервиса	4 дней	Вт 27.04.21	Пт 30.04.21	9	Аналитик[50%];Программист 2[50%];Интернет[100,00 Р]
12	✓	Проектирование REST API приложения	4 дней	Пн 03.05.21	Пт 07.05.21	11	Программист 2;Интернет[100,00 Р]
13	✓	Проектирование интерфейса	3 дней	Вт 27.04.21	Пт 30.04.21	9	Программист 1;Интернет[100,00 Р]
14	✓	Проектирование клиентского приложения	4 дней	Пн 03.05.21	Пт 07.05.21	13	Программист 1;Интернет[100,00 Р]
15	✓	Проектирование программных средств завершено	0 дней	Пн 10.05.21	Пн 10.05.21	13;12	
16	✓	Разработка программных средств	19 дней	Пн 10.05.21	Пт 04.06.21		
17	✓	Разработка структур данных	5 дней	Пн 10.05.21	Пт 14.05.21	15	Программист 2[70%];Программист 1[30%];Интернет[100,00 Р]
18	✓	Разработка REST API приложения	10 дней	Пн 17.05.21	Чт 03.06.21	17	Программист 2;Интернет[300,00 Р]
19	✓	Разработка интерфейса	10 дней	Пн 10.05.21	Пт 21.05.21	15	Программист 1[70%]
20	✓	Разработка клиентского приложения	9 дней	Пн 24.05.21	Чт 03.06.21	19	Программист 1;Интернет[250,00 Р]
21	✓	Разработка программных средств завершена	0 дней	Пт 04.06.21	Пт 04.06.21	20;18	
22	✓	Тестирование	4 дней	Пт 04.06.21	Чт 10.06.21		
23	✓	Модульное тестирование	2 дней	Пн 04.06.21	Пн 07.06.21	21	Программист 1[80%];Тестировщик[20%]
24	✓	Функциональное тестирование	2 дней	Пн 04.06.21	Пн 07.06.21	21	Тестировщик[80%]
25	✓	Нагрузочное тестирование	2 дней	Вт 08.06.21	Ср 09.06.21	23;24	Тестировщик;Интернет[100,00 Р]
26	✓	Тестирование завершено	0 дней	Чт 10.06.21	Чт 10.06.21	25	
27	✓	Конец проекта	0 дней	Чт 10.06.21	Чт 10.06.21	26	

Рисунок 43 – Список задач проекта

Для реализации проекта необходимо выделить основные ресурсы, их типа и стоимость. Описание ресурсов представлено на рисунке 44.

	1	Название ресурса	Тип	Единицы измерения	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочн	Затраты на исполн.	Начисление	Базовый календарь	Ко
1		Аналитик	Трудовой		А		100%	000,00 Р/мес	100,00 Р/ч	0,00 Р	Пропорционал	Стандартный	
2		Программист 1	Трудовой		П1		100%	000,00 Р/мес	150,00 Р/ч	0,00 Р	Пропорционал	Стандартный	
3		Программист 2	Трудовой		П2		100%	000,00 Р/мес	200,00 Р/ч	0,00 Р	Пропорционал	Стандартный	
4		Тестировщик	Трудовой		Т		100%	000,00 Р/мес	100,00 Р/ч	0,00 Р	Пропорционал	Стандартный	
5		Интернет	Затраты		И						Пропорционал		

Рисунок 44 – Список ресурсов проекта

В результате выделения задач и ресурсов была построена диаграмма Ганта (рисунок 45). Н ней представлены продолжительность задач и взаимосвязи их друг с другом

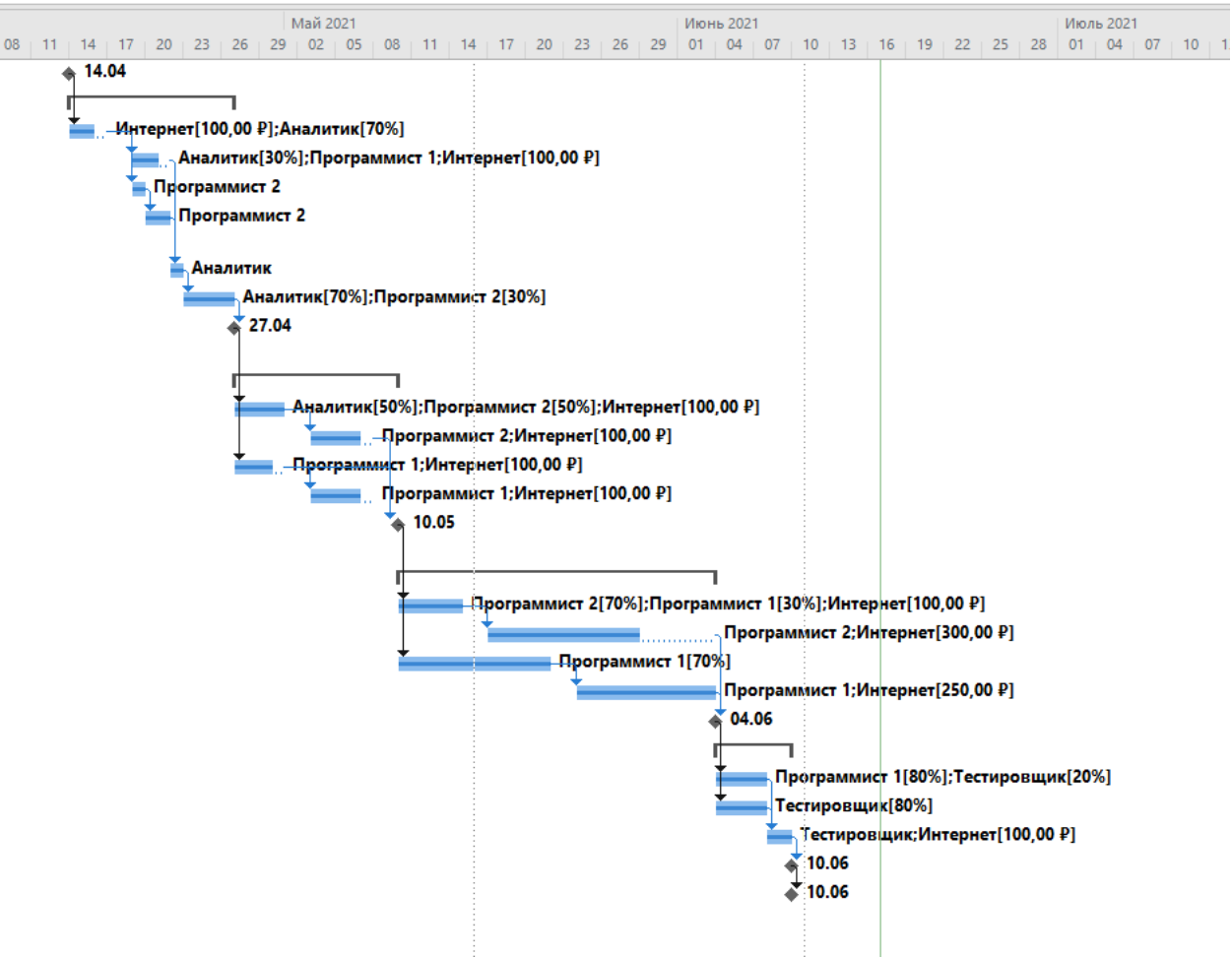


Рисунок 45 – Диаграмма Ганта

На рисунке 46 представлены гистограмма загрузки ресурсов.

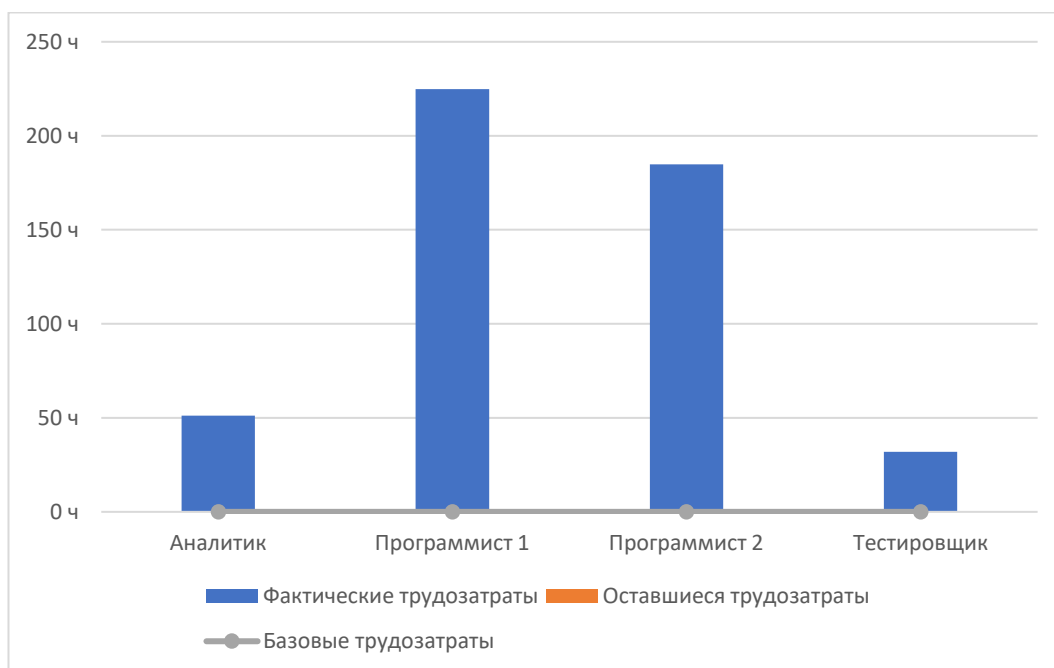


Рисунок 46 – Гистограмма загрузки ресурсов

На рисунке 47 представлена гистограмма затрат на задачи.

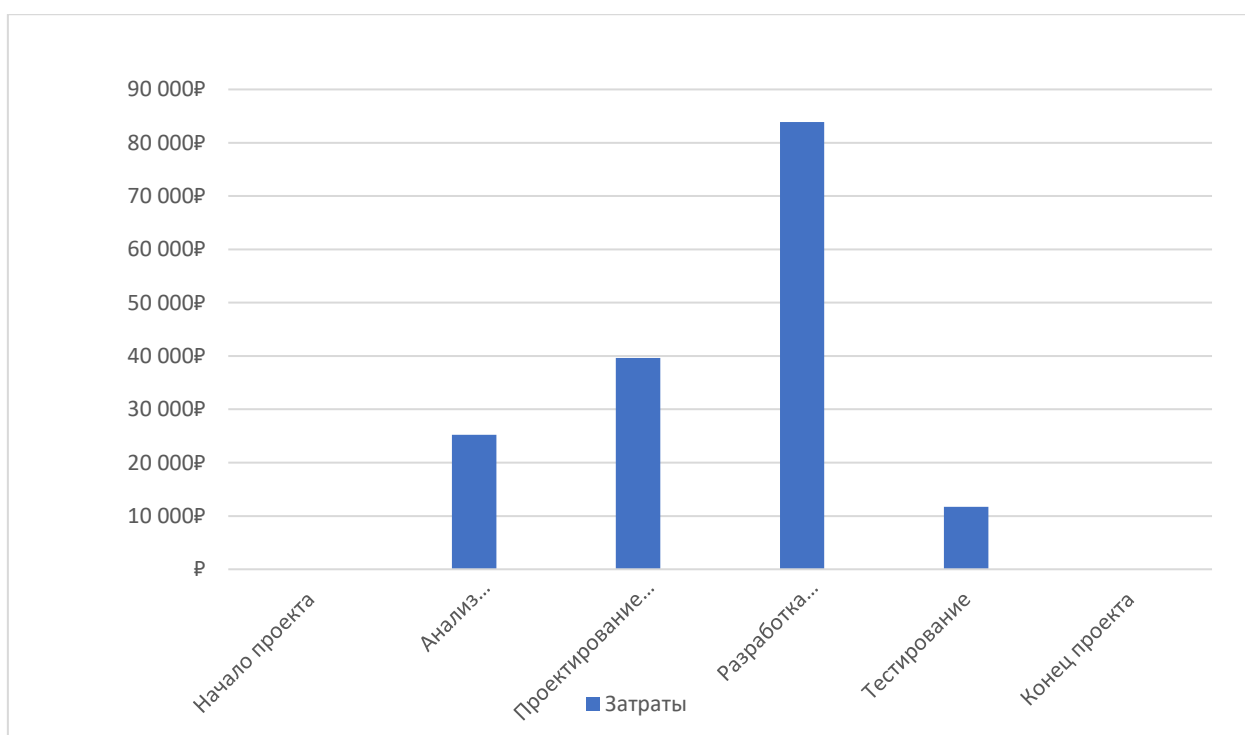


Рисунок 47 – Гистограмма затрат на задачи

Таким образом разработка проекта составит 41 полных рабочих дней с 14 апреля 2021 года по 10 июня 2021 года. Трудозатраты составят 492 человеко-часов. Оценка бюджета проекта составляет 160 450 рублей.

Заключение

В ходе выполнения выпускной квалификационной работы разработан сервис для хранения, поиска и визуализации открытых данных.

Проведен анализ методов визуализации и понижения размерности данных. Данные методы подробно описаны и в каждом из них выделены основные черты и преимущества. Также проведен сравнительный анализ сервисов поиска, хранения и визуализации наборов открытых данных и были выделены основные особенности их реализации.

На данной основе выделены функциональные и нефункциональные требования к системе.

Для решения данной задачи выбрана архитектура системы и основной шаблон проектирования. Также спроектирована логическая и физическая схема БД, отображающая основные сущности и связи в выбранной предметной области. Для основных модулей построены диаграммы классов. А также работа основных сервисов системы представлена на диаграммах UML, таких как диаграмма деятельность, последовательности и состояний.

Разработка программных средств проводилась в среде разработки VS Code на языках программирования Python и JavaScript. Разработанная программа функционирует на любом современном web-браузере, а разработанное API обладает быстрой переносимостью, благодаря использованию docker контейнеров.

Разработанное программное обеспечение протестировано. Выполнено функциональное и модульное тестирование. Все тесты пройдены успешно.

Также проведено планирование и оценка бюджета проекта при помощи программы для управления проектами Microsoft Project. Затраты на проект составили 160 450 рублей. Время разработки 41 рабочих дня.

Таким образом, в результате выполнения выпускной квалификационной работы разработан сервис для хранения, поиска и визуализации открытых данных, отвечающий всем требованиям технического задания.

Список использованных источников

1. Mizuho Bank Ltd. Real-time analytics helps contact center agents serve customers more effectively // IBM [Электронный ресурс]. – URL: <https://www.ibm.com/case-studies/mizuho-bank-ltd> (дата обращения: 24.05.2021).
2. IHS Markit Develops NLP Solution for Financial Professionals in 6 Weeks on AWS // Amazon Web Services [Электронный ресурс]. – URL: <https://aws.amazon.com/ru/solutions/case-studies/ihsmarkit/> (дата обращения: 24.05.2021)
3. Georgia-Pacific Optimizes Processes, Saves Millions of Dollars Yearly Using AWS // Amazon Web Services [Электронный ресурс]. – URL: <https://aws.amazon.com/ru/solutions/case-studies/georgia-pacific/> (дата обращения: 24.05.2021).
4. Jelonek, D. Big Data Analytics in the Management of Business // Jelonek D. – MATEC Web of Conferences, 2017. – 6 с.
5. Amazon Web Services joins NIH’s STRIDES Initiative to harness latest cloud technologies for biomedical researchers // National Institute of Health [Электронный ресурс]. – URL: <https://www.nih.gov/news-events/news-releases/amazon-web-services-joins-nih-strides-initiative-harness-latest-cloud-technologies-biomedical-researchers> (дата обращения: 24.05.2021).
6. Gautam, P. A Machine Learning Approach to Identify Socio-Economic Factors Responsible for Patients Dropping out of Substance Abuse Treatment / P. Gautam, Singh P. – American Journal of Public Health Research, 2020. – С. 140–146.
7. Koshy, R. Enhanced Deep Learning Architectures for Face Liveness Detection for Static and Video Sequence/ R. Koshy, A. Mahmood. – Entropy, 2020. – 28 с.
8. Указ Президента РФ «О развитии искусственного интеллекта в Российской Федерации» (вместе с «Национальной стратегией развития искусственного интеллекта на период до 2030 года») от 10 октября 2019 г.

№ 490. [Электронный ресурс]. – URL: http://www.consultant.ru/document/cons_doc_LAW_335184/8554fa37472e5d49b781f7f04055ef356250d745/ (дата обращения: 24.05.2021).

9. Киссель, Н. Открытые данные в российском правовом поле: обзор лучших практик, рекомендаций и барьеров/ Н. Киссель // Информационная культура [Электронный ресурс]. – URL: <https://opendatareview.infoculture.ru/> (дата обращения: 24.05.2021).

10. International Open Data Charter. Principles. // Open Data Charter [Электронный ресурс]. – URL: <https://opendatacharter.net/principles/> (дата обращения: 24.05.2021).

11. Данные НКО [Электронный ресурс]. – URL: <http://ngodata.ru/> (дата обращения: 25.05.2021)

12. Портал открытых данных Министерства Культуры РФ [Электронный ресурс]. – URL: <https://opendata.mkrf.ru/> (дата обращения: 25.05.2021)

13. Документация портала data.world [Электронный ресурс]. – URL: <https://help.data.world/hc/en-us/> (дата обращения: 25.05.2021)

14. Портал по проведению соревнования по машинному обучению Kaggle [Электронный ресурс]. – URL: <https://www.kaggle.com/> (дата обращения: 25.05.2021)

15. Портал по проведению соревнования по машинному обучению Drivendata [Электронный ресурс]. – URL: <https://www.drivendata.org/> (дата обращения: 25.05.2021)

16. Документация по использованию облачного сервиса визуализации данных Yandex DataLens [Электронный ресурс]. – URL: <https://cloud.yandex.ru/docs/datalens/solutions/data-from-csv-visualization> (дата обращения: 25.05.2021)

17. van der Maaten, L. Dimensionality Reduction: A Comparative Review/ L. van der Maaten, E. Postma, J. van den Herik. – TiCC, Tilburg University, 2009. – 36 с.

18. Friedman, J. The Elements of Statistical Learning / J. Friedman, T. Hastie, R. Tibshirani. – Springer Series in Statistics, 2001. – 764 с.
19. Гудфеллоу, Я. Глубокое обучение/ Я. Гудфеллоу, И. Бенджио, А. Курвилль. – ДМК Пресс, 2018. – 653 с.
20. Арлоу Д. UML 2 и Унифицированный процесс, 2-е издание. Практический объектно-ориентированный анализ и проектирование. / Арлоу Д., Нейштадт А. – СПб: Символ Плюс, 2007. – 624 с.
21. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение Теория и практика. / Коннолли Т., Бегг К. – Санкт Петербург: Питер, 2003. – 1440 с.
22. Официальная документация по фреймворку React.js / Документация по фреймворку React.js [Электронный ресурс] – URL: <https://reactjs.org/> (дата обращения 30.05.2021)
23. Управление проектами в Microsoft Project: учеб.-метод. пособие для студентов по направлению специальности «Информационные системы и технологии (издательско-полиграфический комплекс)» / Т. П. Брусенцова, В. В. Смелов. – Минск: БГТУ, 2011. – 160 с.

Приложение А
ГЛОССАРИЙ
(справочное)

База данных (БД) – это информативная модель, позволяющая организованно поддерживать данные о группе объектов, имеющих один и тот же набор параметров.

Unified Modeling Language (UML) – графический унифицированный язык моделирования, использующий объектно-ориентированный подход для моделирования в сфере разработки программного обеспечения.

Дашборд – инструмент для визуализации и анализа информации, представленный в виде доски с ключевыми графиками.

Открытые данные – общедоступная информация, размещенная в Интернете для неоднократного, свободного и бесплатного использования в виде машиночитаемых систематизированных данных, в формате, позволяющем их автоматизированную обработку.

Уменьшение размерности данных – это процесс уменьшения количества признаков до наиболее оптимального показателя.

ORM – технология, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Метод главных компонент (PCA) – метод снижения размерности данных, который использует собственные вектора и собственные значения ковариационной матрицы.

Линейный дискриминантный анализ (LDA) – метод поиска линейной комбинации признаков, который часто используется для снижения размерности данных перед классификацией.

Приложение Б
ЛИСТИНГ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Б.1 – Файл user_services.py

```

from application import schemas, models
from application.services.validate_service import (
    UserValidateProcess,
    UsernameValidate,
    EmailValidate,
    PasswordValidate,
    UserRegisterValidate
)
from datetime import datetime, timedelta
from application.facades.facades import UserFacade, GroupFacade,
UserGroupFacade, CountDatasetFacade
from application.services.create_records import Creator, UserCreator
from flask_jwt_extended import (
    create_access_token,
    create_refresh_token,
)
from passlib.hash import pbkdf2_sha256 as sha256
import re
from abc import ABC, abstractmethod
import socket

class RegistrationStrategy(ABC):
    @abstractmethod
    def registrate(self):
        pass

class AuthorizationStrategy(ABC):
    @abstractmethod
    def authorize(self):
        pass

class ContextReg:

    def __init__(self, strategy):
        self.strategy = strategy

    def registrate(self):
        return self.strategy.registrate()

class ContextAuth:

    def __init__(self, strategy):
        self.strategy = strategy

    def authorize(self):
        return self.strategy.authorize()

class RegistrationService(RegistrationStrategy):

    def __init__(self, json):
        self.json = json
        self.g_facade = GroupFacade()
        self.u_facade = UserFacade()
        self.ug_facade = UserGroupFacade()

    def registrate(self):
        data = schemas.UserSchema().load(data=self.json)
        validate_user = UserValidateProcess()
        validate_user.register(
            UsernameValidate(data['username']),
            PasswordValidate(data['password']),
            EmailValidate(data['email']),

```

```

        UserRegisterValidate(self.u_facade, data['username'],
data['email'])
    )

    try:
        validate_user.validate()
    except ValueError as ex:
        return 400, {"error" : str(ex)}

    try:
        user = UserCreator(self.g_facade, self.ug_facade, self.u_facade,
data).create()
        a_token, r_token = GenereteJWTService.create_jwt(user)

        return 200, {
            "message"      : f'User {user.username} was registred',
            "access_token" : a_token,
            "refresh_token" : r_token
        }
    except:
        return 500, {"error": "Internal server error"}

class AuthorizationService(AuthorizationStrategy):

    valid_email_regex = "[^@]+@[^@]+\.[^@]+"

    def __init__(self, json):
        self.json = json
        self.u_facade = UserFacade()

    def get_user(self, param):
        if not re.match(self.valid_email_regex, param):
            return self.get_user_by_username(param)
        else:
            return self.get_user_by_email(param)

    def get_user_by_username(self, username):
        user = self.u_facade.get_user_by_username(username)
        if not user:
            raise ValueError(f"Username {username} doesn't exist")
        return user

    def get_user_by_email(self, email):
        user = self.u_facade.get_user_by_email(email)
        if not user:
            raise ValueError(f"Email {email} doesn't exist")
        return user

    def verify_password(self, verified_pass, hash):
        if not sha256.verify(verified_pass, hash):
            raise ValueError("Incorrect password")

    def authorize(self):
        data = schemas.UserSchema().load(data=self.json)
        try:
            user = self.get_user(data["username"])
            self.verify_password(data['password'], user.password)
            a_token, r_token = GenereteJWTService.create_jwt(user)
            user.last_login = datetime.now()
            self.u_facade.change(user)

            return 200, {

```

```

        'message': f'Logged in as {user.username}',
        'id': user.id,
        'username': user.username,
        'roles': [x.group.name for x in user.groups],
        'access_token': a_token,
        'refresh_token': r_token
    }

    except ValueError as ex:
        return 400, {"error" : "The username or password provided is
incorrect."}

class GenereteJWTService:

    def __init__(self, json):
        self.json = json

    def generate_access_token(self):
        expires = timedelta(days=2)
        return create_access_token(identity = self.json,
expires_delta=expires)

    def generate_refresh_token(self):
        return create_refresh_token(identity = self.json)

    @staticmethod
    def create_jwt(user: models.User):
        json = {'id' : user.id, 'username' : user.username}
        jwt = GenereteJWTService(json)

        access_token = jwt.generate_access_token()
        refresh_token = jwt.generate_refresh_token()

        return access_token, refresh_token

```

Листинг Б.2 – Файл dataset_services.py

```

import os
from application.config import Config
from application.facades import facades
from application.services.validate_service import FilenameValidate
from application.services.dataset import upload_files as up
from application.services import create_records as cr

from flask import Response, send_from_directory
from application import schemas
import zipstream
from threading import Thread, Lock
import re
import imghdr

class DataSetDownloadService:
    """
    Service for downloading of datasets
    """

    def download(self, path):

        z = zipstream.ZipFile(mode='w', compression=zipstream.ZIP_DEFLATED)
        pat = os.path.join(Config.DATASETS_ABS_PATH, path)

        if not os.path.exists(pat):
            return 404, {'error': "Path didn't exist"}

```

```

    for root, dirs, files in os.walk(pat):
        for file in files:
            absname = os.path.abspath(os.path.join(root, file))
            arcname = absname[len(pat) + 1:]
            z.write(absname, arcname=arcname)

    response = Response(z, mimetype='application/zip')
    response.headers['Content-Disposition'] = 'attachment;
filename={}'.format('archive.zip')
    return 200, response

class DataSetUploadService:
    """
    Service for uploading of datasets
    """

    types = [
        'csv',
        'json'
    ]

    def __init__(self, j_data, user, file):

        self.data = schemas.DataSetSchema().load(data=j_data)
        self.user = user
        self.file = file

    def get_type(self, f):
        try:
            extension = f.rsplit('.', 1)[1].lower()
        except ValueError:
            extension = ''

        if extension not in self.types:
            return 'other'
        else:
            return extension

    def find_types_and_size(self, dir):
        t = set()

        sizes = ['B', 'KB', 'MB', 'GB']
        total_size = {'B': 0, 'KB': 0, 'MB': 0, 'GB': 0}

        for dirpath, _, filenames in os.walk(dir):
            for f in filenames:
                fp = os.path.join(dirpath, f)

                size = os.path.getsize(fp)
                i = 0

                while size > 0 and i < len(sizes):
                    total_size[sizes[i]] += size % 1024
                    size = size // 1024
                    i += 1

                for s in range(len(sizes) - 1):
                    if total_size[sizes[s]] // 1024 > 0:
                        total_size[sizes[s + 1]] += total_size[sizes[s]] //
1024

                        total_size[sizes[s]] = total_size[sizes[s]] % 1024

                t.add(self.get_type(f))

```

```

        return t, total_size

    def __check_error_dataset(self, name):
        if facades.DataSetFacade().get_dataset_by_name(name):
            raise ValueError('Data set with this name was already upload')

    def upload(self):

        def extract(arch: up.Extractor):
            mutex.acquire()
            arch.extract()
            mutex.release()

        def write_dataset(upload_dir: str, dataset_id: int):

            dataset = facades.DataSetFacade().get_entity(dataset_id)

            mutex.acquire()
            types, size = self.find_types_and_size(upload_dir)
            mutex.release()

            d_creator.set_size(dataset, size)
            d_type = cr.DataSetTypeCreator(types, dataset)
            d_type.create()

            dataset.is_loaded = True
            facades.DataSetFacade().change(dataset)

        try:
            f_validate = FilenameValidate(self.file.filename)
            f_validate.validate()

            self.__check_error_dataset(self.data['name'])

            upload_dir = os.path.join(Config.DATASETS_ABS_PATH,
self.user['username'], self.data['name'])
            upload_creator = up.UploadCreator(upload_dir, self.file)

            uploader = upload_creator.create()
            uploader.upload()
            mutex = Lock()

            extr = upload_creator.get_extractor()

            if extr:
                thread_2 = Thread(target=extract, kwargs={'arch': extr})
                thread_2.start()

            d_creator = cr.DataSetCreator(self.data, self.user)
            dataset = d_creator.create()

            thread_1 = Thread(target=write_dataset, kwargs={'upload_dir':
upload_dir, 'dataset_id': dataset.id})
            thread_1.start()

            return 200, {'message': 'file saved'}

        except ValueError as err:
            return 400, {"error": str(err)}

        except Exception as err:
            return 500, {"error": "Can't upload dataset"}

```

Листинг Б.3 – Файл data_chart_services.py

```

from dask import dataframe as dd
from application.config import Config
import os
import application.facades.facades as fc
from sklearn.decomposition import KernelPCA
from sklearn.manifold import TSNE
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd

class CalcData:

    def __init__(self, path, type_id, col1_id = None, col2_id = None):
        self.path = path
        self.abs_path = os.path.join(Config.DATASETS_ABS_PATH, self.path)
        self.df: dd.DataFrame = dd.read_csv(self.abs_path)
        self.chart_type = type_id

        self.col1 = col1_id
        self.col2 = col2_id

        self.types = {
            'integer': 'int64',
            'string': 'string',
            'float': 'float64'
        }

        self.default_agg = {
            'string': 'count',
            'integer': 'sum',
            'float': 'sum'
        }

        self.aggregations = {
            'count': lambda x: x.count(),
            'countunique': lambda x: x.nunique(),
            'sum': lambda x: x.sum(),
            'avg': lambda x: x.mean(),
            'min': lambda x: x.min(),
            'max': lambda x: x.max(),
        }

    def get_chart_type(self):
        return fc.ChartTypeFacade().get_entity(self.chart_type)

    def get_col(self, col_id):
        return fc.DataSetColumnVersionedFacade().get_entity(col_id)

    def calc_pca(self, df):

        df = df.compute()
        df = df.dropna(axis=0)

        for col in df.columns:
            try:
                if df[col].dtype == "string":
                    df[col] = pd.factorize(df[col])[0]
            except TypeError:
                if df[col].dtype == "object":
                    df[col] = pd.factorize(df[col])[0]

        scaler = StandardScaler()

```

```

# y = df['output']

scaler.fit(df)
df = scaler.transform(df)
pca = KernelPCA(n_components=2, kernel='rbf', max_iter=100)

df = pca.fit_transform(df)
df = df.T

pdf = pd.DataFrame()
pdf["x"] = df[0]
pdf["y"] = df[1]

return pdf

def calc(self):

    df: dd.DataFrame = self.df.persist()
    fraction = 1000 / len(df)

    if fraction > 1:
        df = df.sample(frac = 1, random_state=10)
    else:
        df = df.sample(frac = fraction, random_state=10)

    chart_type = self.get_chart_type()

    if chart_type.id != 5 and chart_type.id != 6:

        col_x = self.get_col(self.col1)
        col_y = self.get_col(self.col2)

        x_name = col_x.dataset_column_sources.tittle
        y_name = col_y.dataset_column_sources.tittle

        df[x_name] =
df[x_name].astype(self.types[col_x.data_type_aggregation.data_type.name])
        df[y_name] =
df[y_name].astype(self.types[col_y.data_type_aggregation.data_type.name])

        if chart_type.id != 2 or
col_y.data_type_aggregation.aggregation.name != "none":
            aggregation = col_y.data_type_aggregation.aggregation.name

            if aggregation == "none":
                aggregation =
self.default_agg[col_y.data_type_aggregation.data_type.name]

        result =
self.aggregations[aggregation](df.groupby(x_name)[y_name]).compute()

        pdf = pd.DataFrame()

        pdf["x"] = result.keys()
        pdf["y"] = list(result)

    if chart_type.id == 2:
        df = df[(df[x_name].notnull()) & (df[y_name].notnull())]
        d = {
            "x": df[x_name].compute(),

```

```

        "y": df[y_name].compute()
    }
    pdf = pd.DataFrame(d)

    aggregation = col_y.data_type_aggregation.aggregation.name

elif chart_type.id == 5:
    x_name = "PCA 1"
    y_name = "PCA 2"
    pdf = self.calc_pca(df)
    aggregation = "none"

elif chart_type.id == 6:

    col_x = self.get_col(self.col1)
    classes = col_x.dataset_column_sources.tittle

    x_name = "LDA 1"
    y_name = "LDA 2"
    pdf = self.calc_lda(df, classes)
    aggregation = "none"

pdf["xlabel"] = pd.factorize(pdf["x"])[0]
pdf["ylabel"] = pd.factorize(pdf["y"])[0]

try:
    categorical_x = pdf.dtypes["x"] == "object" or pdf.dtypes["x"] ==
"string"
except TypeError:
    categorical_x = False

try:
    categorical_y = pdf.dtypes["y"] == "object" or pdf.dtypes["y"] ==
"string"
except TypeError:
    categorical_y = False

return {
    "type" : chart_type.name,
    "agg" : aggregation,
    "x" : {
        "data" : list(pdf["xlabel"]),
        "label" : list(pdf["x"]),
        "max" : int(pdf["xlabel"].max()) if categorical_x else
int(pdf["x"].max()),
        "min" : int(pdf["xlabel"].min()) if categorical_x else
int(pdf["x"].min()),
        "field" : x_name,
        "categorical" : categorical_x
    },
    "y" : {
        "data" : list(pdf["ylabel"]),
        "label" : list(pdf["y"]),
        "field" : y_name,
        "categorical" : categorical_y
    }
}

```

Листинг Б.4 – Файл DataSet.jsx

```

import React, { Component } from "react";
import DataSetService from "../../services/DataSetService";
import DataSetDirList from "../Folders/DataSetDirList";
import DataSetFile from "../Preview/DataSetFile";
import DataHead from "../Header/DataHead";
import DataSetInfo from "../Header/DataSetInfo";
import DataSetRating from "../DataSetRating";
import ModalDatasetCreate from "../Header/ModalDatasetCreate";

import "../DataSet.css";

const API_URL = "http://localhost:5000/api/v1/dataset/download/"

export default class DataSet extends Component {

  constructor(props) {
    super(props);
    const args = this.props.match.params;
    this.myRef = React.createRef();

    this.onCloseModal = this.onCloseModal.bind(this);
    this.onOpenModal = this.onOpenModal.bind(this);

    this.state = {
      username: args.user,
      datasetName: args.dataset,
      pathToFile: "",
      filename: "",
      dataset: null,
      error: null,
      showModal: false
    };
  }

  onOpenModal() {
    this.setState({
      showModal: true,
    });
  }

  onCloseModal() {
    this.setState({
      showModal: false,
    });
  }

  componentDidMount() {
    const args = this.props.match.params;

    DataSetService.getDataset(args.user, args.dataset).then(
      response => {
        if (response.data) {
          this.setState({
            dataset: response.data,
          });
          console.log(response.data);
        }
      },
      error => {
        this.setState({
          error:

```

```

        (error.response && error.response.data) ||
        error.message ||
        error.toString()
    });
    }
    );
}

clickOnFile = (filename, pathToFile) => {
    this.setState({
        pathToFile: pathToFile,
        filename: filename,
    })
}

render() {
    const{ dataset, datasetName, username, pathToFile, filename, showModal }
= this.state;
    return (
        <>
            { dataset ? (
                <div className="container">

                    <ModalDatasetCreate show={showModal} onClose={() =>
this.onCloseModal()} path={`/${username}/${datasetName}`} />

                    <DataHead
                        tittle={dataset.title}
                        helpInfo={[username, dataset.dataset_meta.size + " " +
dataset.dataset_meta.size_name]}

                        date_load={dataset.date_load}
                        since_create={dataset.since_created}

                        hrefs={{
                            'Data': {
                                'active' : true,
                                'href' : "/"
                            },

                            'Data tables': {
                                'active' : false,
                                'href' : "/"
                            },

                            'Charts': {
                                'active' : false,
                                'href' : "/"
                            }
                        }}

                        buttons={{
                            'Download':
                                <a href={API_URL + dataset.user.username + "/" +
dataset.name}>
                                    <button className="white-apply-button "
style={{marginRight: 8}}>
                                        <span className="text-in-white-apply-button">
                                            Download
                                        </span>
                                    </button>
                                </a>,
                            'New dataset':
                                <button className="apply-button" onClick={this.onOpenModal}>

```

```

        <span className="text-in-apply-button">
          New dataset
        </span>
      </button>
    )
  }
}
</div>

<div> {dataset.description} </div>

<DataSetInfo rating={dataset.rating ? dataset.rating : "None"}
  tags={dataset.tags}
  size={dataset.dataset_meta.size}
  size_name={dataset.dataset_meta.size_name} dId={dataset.id}
  duId={dataset.user.id} />

  <div style={{marginTop: "20px", position: "relative", overflowX:
    "hidden"}}>
    <div style={{display: "flex"}}>
      <div className="block">
        <DataSetDirList path= `${username}/${datasetName}`
        className="first-ul" clickOnFile={this.clickOnFile} filter="none"/>
      </div>

      <div className="sub-block">
        <div className="file-reader-block">
          <div className="header-block">
            <h5>
              { pathToFile ? (<{filename}</>) : (<>Select file to
preview</>)}
            </h5>
          </div>

          { pathToFile &&
            (<DataSetFile pathToFile={pathToFile} />)
          }
        </div>
      </div>
    </div>

    <div>
      <DataSetRating ratings={dataset.user_ratings}
        duId={dataset.user.id} dId={dataset.id}/>
      <div ref={this.myRef}>
        </div>
      </div>
    </div>
  ) : (<div/>)}
</>
)
}
}

```

Листинг Б.5 – Файл DataLineAreaChart.jsx

```

import React, { Component } from "react";
import { AreaChart, Area, XAxis, YAxis, CartesianGrid, Tooltip,
ResponsiveContainer } from 'recharts';

export default class DataLineAreaChart extends Component {

  constructor(props) {
    super(props);
    this.state = {
      data: this.props.data.x.data.map((el, i) => ({
        'x': el,
        'x_label' : this.props.data.x.label[i],

```

```

        'y': this.props.data.y.data[i],
        'y_label' : this.props.data.y.label[i],
      )),
    }
    this.ref = React.createRef();
  }

  render() {
    const {data} = this.state;

    let getValX = (x) => {
      return this.props.data.x.categorical ? x.x : x.x_label;
    }

    let getValY = (y) => {
      return this.props.data.y.categorical ? y.y : y.y_label;
    }

    let lenT = (this.props.data.x.max - this.props.data.x.min) > 12 ? 12 :
    (this.props.data.x.max - this.props.data.x.min);

    let lenInt = (this.props.data.x.max - this.props.data.x.min) / lenT;

    let xTicks = Array(lenT + 1).fill().map((i, index) => {
      return index !== (lenT) ? this.props.data.x.min + parseInt(lenInt) *
index : this.props.data.x.max
    }));

    return (
      <ResponsiveContainer width="100%" height="100%">
        <AreaChart
          width={1200}
          height={800}
          data={data}
        >
          <CartesianGrid />

          <XAxis
            style={{fontSize: 12, fill: "#666"}}
            padding={{left: 12}}
            height={70}
            tickCount={lenT + 1}
            ticks={xTicks}
            type="number"
            label={<CustomXLabel
              label={this.props.data.x.field}
            />}
            dataKey={getValX}

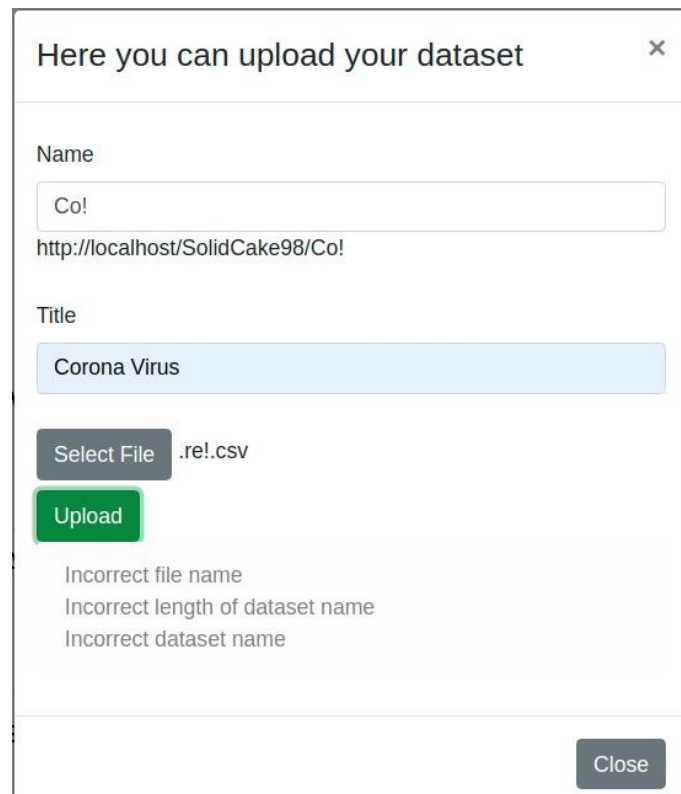
            domain={['dataMin - 1', 'dataMax + 1']}
          />

          <YAxis/>

          <Area dataKey={getValY} fill="rgb(77, 162, 241)"
isAnimationActive={false}/>
        </AreaChart>
      </ResponsiveContainer>
    )
  }
}

```

Приложение В
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)



Here you can upload your dataset

Name

Co!

http://localhost/SolidCake98/Co!

Title

Corona Virus

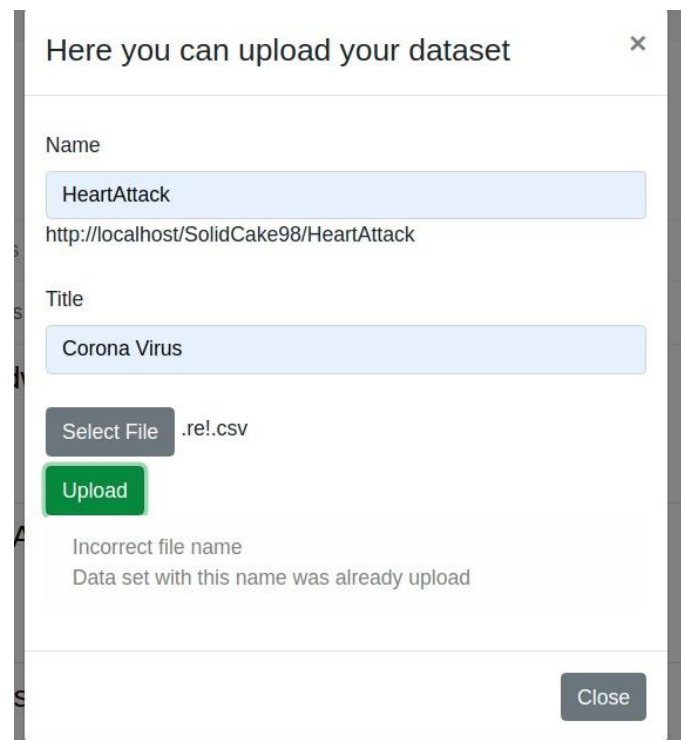
Select File .re!.csv

Upload

Incorrect file name
Incorrect length of dataset name
Incorrect dataset name

Close

Рисунок В.1 – Тест 1, покрывающий таблицу решений



Here you can upload your dataset

Name

HeartAttack

http://localhost/SolidCake98/HeartAttack

Title

Corona Virus

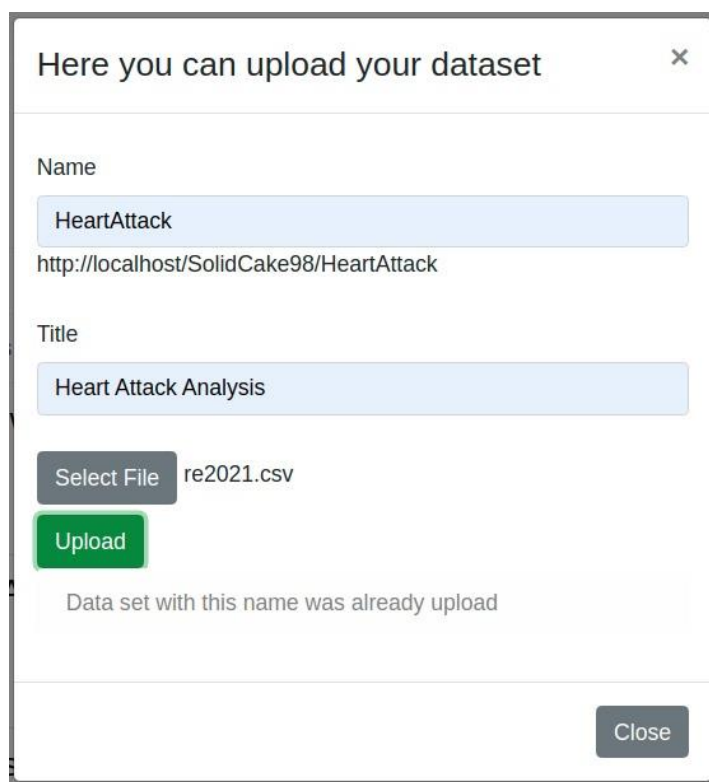
Select File .re!.csv

Upload

Incorrect file name
Data set with this name was already upload

Close

Рисунок В.2 – Тест 2, покрывающий таблицу решений



Here you can upload your dataset ×

Name
HeartAttack
http://localhost/SolidCake98/HeartAttack

Title
Heart Attack Analysis

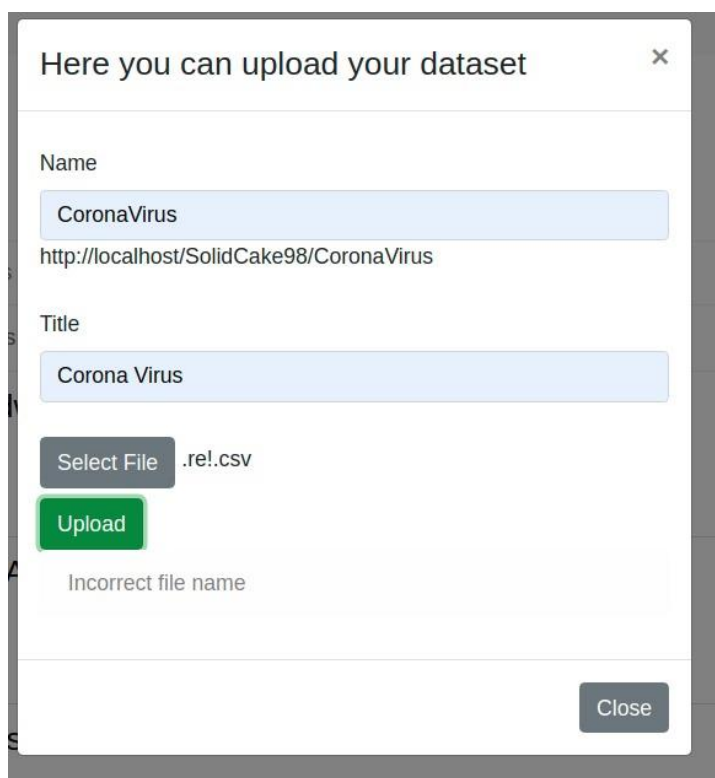
Select File re2021.csv

Upload

Data set with this name was already upload

Close

Рисунок В.3 – Тест 3, покрывающий таблицу решений



Here you can upload your dataset ×

Name
CoronaVirus
http://localhost/SolidCake98/CoronaVirus

Title
Corona Virus

Select File .re!.csv

Upload

Incorrect file name

Close

Рисунок В.4 – Тест 4, покрывающий таблицу решений

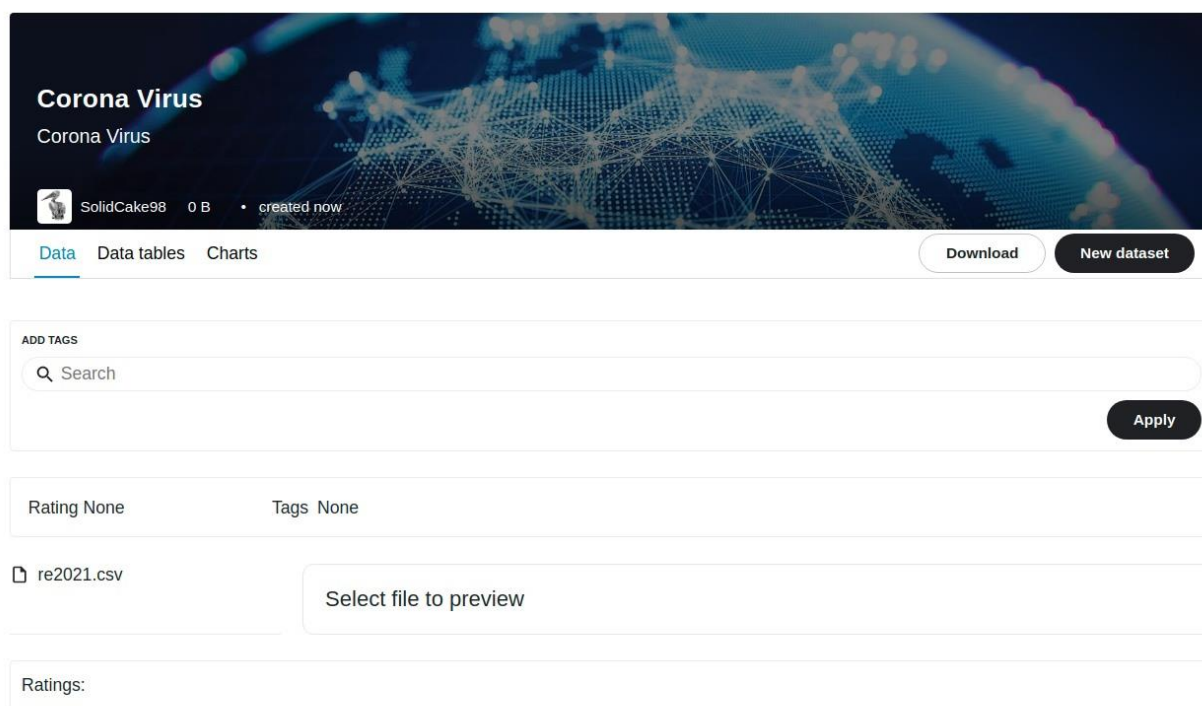


Рисунок В.5 – Тест 5, покрывающий таблицу решений

```

1  ===== test session starts =====
2  platform linux -- Python 3.8.2, pytest-6.0.1, py-1.9.0, pluggy-0.13.1 -- /home/danil/Project/MLService/backend/venv/bin/python3
3  cachedir: .pytest_cache
4  rootdir: /home/danil/Project/MLService/backend
5  plugins: cov-2.10.1, flask-1.0.0
6  collecting ... collected 14 items
7
8  tests/test_upload.py::test_read PASSED
9  tests/test_upload.py::test_read_with_dirs PASSED
10 tests/test_upload.py::test_not_valid_read PASSED
11 tests/test_upload.py::test_download PASSED
12 tests/test_upload.py::test_download_not_found PASSED
13 tests/test_user.py::test_register_user PASSED
14 tests/test_user.py::test_register_not_valid_password_user 400 {'error': 'Password must be at least 8 characters, contain 1 number and 1 letter '}
15 PASSED
16 tests/test_user.py::test_register_not_valid_username_user 400 {'error': 'Username must be at least 5 characters and contain 1 letter'}
17 PASSED
18 tests/test_user.py::test_register_user_is_already_exist 400 {'error': 'User danil already exist'}
19 PASSED
20 tests/test_user.py::test_register_user_email_is_already_exist 400 {'error': 'Email some.email@server.com already registered'}
21 PASSED
22 tests/test_user.py::test_auth_user PASSED
23 tests/test_user.py::test_auth_user_not_valid_email 400 {'error': 'The username or password provided is incorrect.'}
24 PASSED
25 tests/test_user.py::test_auth_not_valid_name 400 {'error': 'The username or password provided is incorrect.'}
26 PASSED
27 tests/test_user.py::test_auth_not_valid_password 400 {'error': 'The username or password provided is incorrect.'}
28 PASSED
29

```

Рисунок В.6 – Результаты модульного тестирования

30	----- coverage: platform linux, python 3.8.2-final-0 -----			
31	Name	Stmts	Miss	Cover
32	-----			
33	application/__init__.py	0	0	100%
34	application/api/__init__.py	0	0	100%
35	application/api/dataset_controller.py	90	52	42%
36	application/api/media_controller.py	12	3	75%
37	application/api/user_controller.py	59	31	47%
38	application/app.py	41	0	100%
39	application/config.py	21	0	100%
40	application/database.py	86	33	62%
41	application/facades/__init__.py	0	0	100%
42	application/facades/abstract_facade.py	23	5	78%
43	application/facades/facades.py	78	32	59%
44	application/models.py	111	2	98%
45	application/schemas.py	69	16	77%
46	application/services/__init__.py	0	0	100%
47	application/services/create_records.py	94	53	44%
48	application/services/dataset/__init__.py	0	0	100%
49	application/services/dataset/file_reader.py	62	39	37%
50	application/services/dataset/read_csv.py	41	34	17%
51	application/services/dataset/upload_files.py	82	50	39%
52	application/services/dataset_services.py	163	80	51%
53	application/services/user_services.py	123	18	85%
54	application/services/validate_service.py	61	8	87%
55	-----			
56	TOTAL	1216	456	62%

Рисунок В.7 – Покрытие кода при модульном тестировании