МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательно учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
Заведующий кафедрой, к.т.н., доцент
_____ М. С. Воробьева
02.07. 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистерская диссертация

РАЗРАБОТКА ХРАНИЛИЩА И ETL-ПРОЦЕССОВ ДЛЯ СИСТЕМЫ СОПРОВОЖДЕНИЯ ИНДИВИДУАЛЬНЫХ ОБРАЗОВАТЕЛЬНЫХ ТРАЕКТОРИЙ СТУДЕНТОВ ВУЗА

02.04.03 Математическое обеспечение и администрирование
информационных систем
Магистерская программа «Разработка технологий
Интернета вещей и больших данных»

Выполнили работу
(групповой проект)
студенты 2 курса
очной формы обучения

Бакланов Иван Андреевич

Иваненко Ольга Александровна

Научный руководитель
к.т.н., доцент

Воробьева Марина Сергеевна

Рецензент
ведущий инженер-программист
отдела информатизации
Отделения по Тюменской
области Уральского главного
управления Центрального банка
Российской Федерации

Ялдыгин Валерий Борисович

Тюмень
2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4

ГЛАВА 1. МЕТОДОЛОГИЯ ПОСТРОЕНИЯ МНОГОУРОВНЕВОГО ХРАНИЛИЩА ДАННЫХ 6

1.1. ПРИНЦИП РАЗБИЕНИЯ ХРАНИЛИЩА НА СЛОИ 6

1.2. ИСТОЧНИКИ И ПЕРИОДИЧНОСТЬ СБОРА ДАННЫХ 8

1.3. ПРОЦЕСС ПОПОЛНЕНИЯ ХРАНИЛИЩА ДАННЫМИ И ПРЕОБРАЗОВАНИЕ ДАННЫХ 11

ГЛАВА 2. РЕАЛИЗАЦИЯ ХРАНИЛИЩА ДАННЫХ 14

2.1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 14

2.2. АРХИТЕКТУРА ХРАНИЛИЩА 14

ГЛАВА 3. РЕАЛИЗАЦИЯ ИНФРАСТРУКТУРЫ РЕШЕНИЯ 31

3.1. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ И ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ 31

3.2. КОМПОНЕНТЫ СИСТЕМЫ 32

3.3. РАЗВЕРТЫВАНИЕ КОМПОНЕНТОВ ЭКОСИСТЕМЫ HADOOP 35

3.4. ПРИМЕНЕНИЕ APACHE NIFI ДЛЯ ПОПОЛНЕНИЯ ХРАНИЛИЩА ДАННЫМИ 37

3.5. АВТОМАТИЗАЦИЯ ЗАДАЧ РАБОТЫ С ХРАНИЛИЩЕМ 48

3.6. ТЕСТИРОВАНИЕ РЕЗУЛЬТАТОВ 52

ЗАКЛЮЧЕНИЕ 59

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 61

ПРИЛОЖЕНИЕ 1. ФАЙЛ DOCKER-COMPOSE ДЛЯ ИНФРАСТРУКТУРЫ СИСТЕМЫ 64

ПРИЛОЖЕНИЕ 2. DOCKER-ФАЙЛ
ДЛЯ ТЕСТОВОГО ОДНОНОДОВОГО КЛАСТЕРА HADOOP.....67

ПРИЛОЖЕНИЕ 3. КОД СКРИПТА ГЕНЕРАЦИИ
КОНФИГУРАЦИОННЫХ ФАЙЛОВ.....71

ВВЕДЕНИЕ

Одним из важных направлений модернизации образования в вузах в последнее время стала индивидуализация обучения студентов. Поэтому задача интерпретации цифрового следа студентов с целью достижения более эффективного результата обучения приобрела особую актуальность [1–3].

В процессе обучения студенты генерируют множество данных, участвуя в различных видах деятельности. Они выбирают элективные курсы, получают оценки, выполняют работы различного вида: пишут программы, курсовые по предмету и по дисциплине, рефераты, эссе, отчеты по учебной, производственной, технологической, преддипломной практикам, и в завершении обучения — выпускные квалификационные работы.

Применение интеллектуального анализа данных к исследованию этой информации (Educational Data Mining) позволяет получить более детальное представление о текущей учебной ситуации и перспективах обучения каждого студента [4–6].

Внедрение индивидуальных образовательных траекторий в Тюменском государственном университете (ТюмГУ) стало отправной точкой для начала работ по анализу данных, генерируемых студентами в процессе обучения. В Институте математики и компьютерных наук были инициированы проекты, посвященные анализу данных об учебном опыте студентов [7, 8]. Для осуществления такого анализа необходимы инструменты сбора и единое пространство, содержащее исходные данные и результаты аналитической обработки данных цифрового следа студентов, собранного за период их обучения [9].

Цель работы — разработать и апробировать подход к построению и наполнению данными единого хранилища информации, которое позволит реализовать конвейер интеллектуального анализа больших данных от интерпретации до выработки рекомендаций.

Задачи:

1. Описать исходные данные и их источники.

2. Разработать структуру хранилища информации.
3. Развернуть хранилище данных.
4. Разработать ETL-процессы пополнения хранилища данными.
5. Организовать доступ к данным в хранилище.

Для подготовки и защиты выпускной квалификационной работы использовались поиск, анализ информации, системный подход для решения поставленных задач; приемы критического анализа проблемных ситуаций, а также средства и методы саморазвития и самореализации; методики межкультурного взаимодействия; умение расставлять приоритеты собственной деятельности при работе в общем проекте в соответствии с командной стратегией для достижения поставленной цели.

Формулирование выводов по итогам проведенной работы осуществлялись с учетом применения современных коммуникативных технологий (в том числе на иностранном языке) для представления результатов на академических, профессиональных, экспертных ИТ-мероприятиях.

ГЛАВА 1. МЕТОДОЛОГИЯ ПОСТРОЕНИЯ МНОГОУРОВНЕВОГО ХРАНИЛИЩА ДАННЫХ

1.1. ПРИНЦИП РАЗБИЕНИЯ ХРАНИЛИЩА НА СЛОИ

При планировании и принятии решений используется анализ больших объемов данных. В их числе как структурированные, так и неструктурированные данные, собранные из функционирующих информационных систем и других источников в образовательном пространстве ТюмГУ.

Отсюда вытекает задача разработки информационного хранилища, консолидирующего все эти данные. Одним из важнейших мест в этих данных являются данные цифрового следа студента [10].

Для создания распределенного информационного хранилища больших данных разработан и реализован оригинальный подход к разработке архитектуры — «Big Data to Smart Data» (BD2SD), опирающийся на мультислойность хранилища и поэтапное преобразование данных цифрового следа.

Такой подход позволит сохранять как сырые данные, так и данные, прошедшие разные уровни обработки, что позволит реализовать информационное хранилище на основе иерархической структуры.

Важным критерием оценки работоспособности хранилища с точки зрения пользователя является скорость, с которой возвращается запрошенный результат. С другой стороны, интеллектуальная обработка данных может занимать продолжительное время. Поэтому необходимо предусмотреть процесс сохранения обработки результатов, что позволит достаточно быстро получать требуемые данные и не нагружать систему многочисленными и ресурсоемкими перерасчетами.

Для создания первого слоя (RD, Raw Data — сырые данные) информационного хранилища, обеспечивающего сбор данных в исходном виде, разработаны методы извлечения ключевых метаданных и выбраны подходы работы с персональной информацией.

Для создания второго слоя (PD, Preprocessed Data — предобработанные данные) использованы преимущества подхода NoSQL (Not only SQL), который упрощает решение задач масштабируемости хранилища и предоставляет возможности согласованного распределенного хранения ключевых метаданных отдельных записей (дата создания записи, структура, ID студента и т.п.). Для выбора оптимального способа хранения данных этого слоя проведено сравнение методов организации данных «Ключ-значение» и «Хранилище документов» в режиме имитационного моделирования запросов на доступ к данным.

Для создания третьего слоя (SD, Smart Data — «умные» данные) используются методы извлечения валидной проблемно-ориентированной информации. Отдельные данные, которые необходимы для решения конкретной задачи, извлекаются из данных слоя PD и передаются в табличном виде гибкому Smart-слою хранилища – для проверки корректности, предварительного анализа и визуализации, а также формирования наборов данных для исследования конкретных проблем с помощью методов машинного и глубокого обучения. На этом же слое хранятся проблемно-ориентированные базы знаний (критерии, правила, образовательные стандарты), пополняемые и/или корректируемые с помощью метода NoSQL запросов при решении определенных задач. Многослойная структура хранилища приведена на рисунке 1.

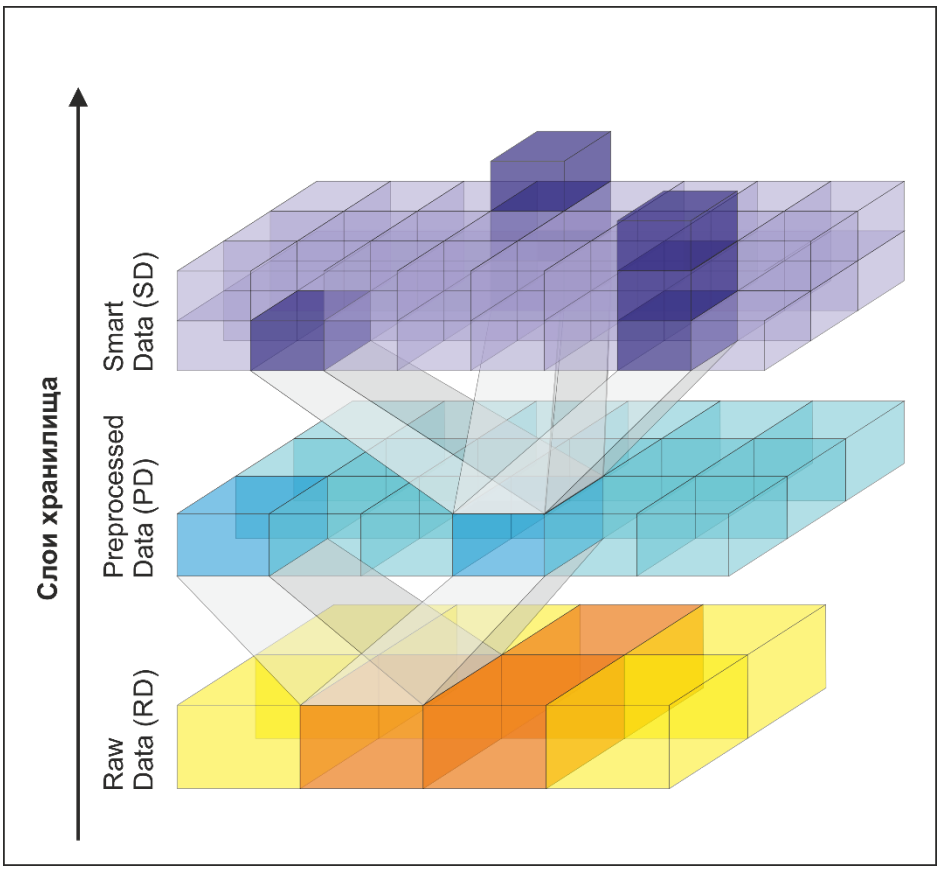

Рис. 1. Многослойная структура хранилища

1.2. ИСТОЧНИКИ И ПЕРИОДИЧНОСТЬ СБОРА ДАННЫХ

При проектировании мультислойного распределенного хранилища данных, аккумулирующего в себе разнородные данные по студентам, собранные за период их обучения, необходимо было учесть, что данные, необходимые для сбора и проведения анализа, распределены по разным источникам, ресурсам, системам, часть из них хранится в виде отдельных файлов, объектов, реестров, методических ресурсов, опросов студентов, файловых хранилищ (Рисунок 2) и собираются с разной периодичностью.

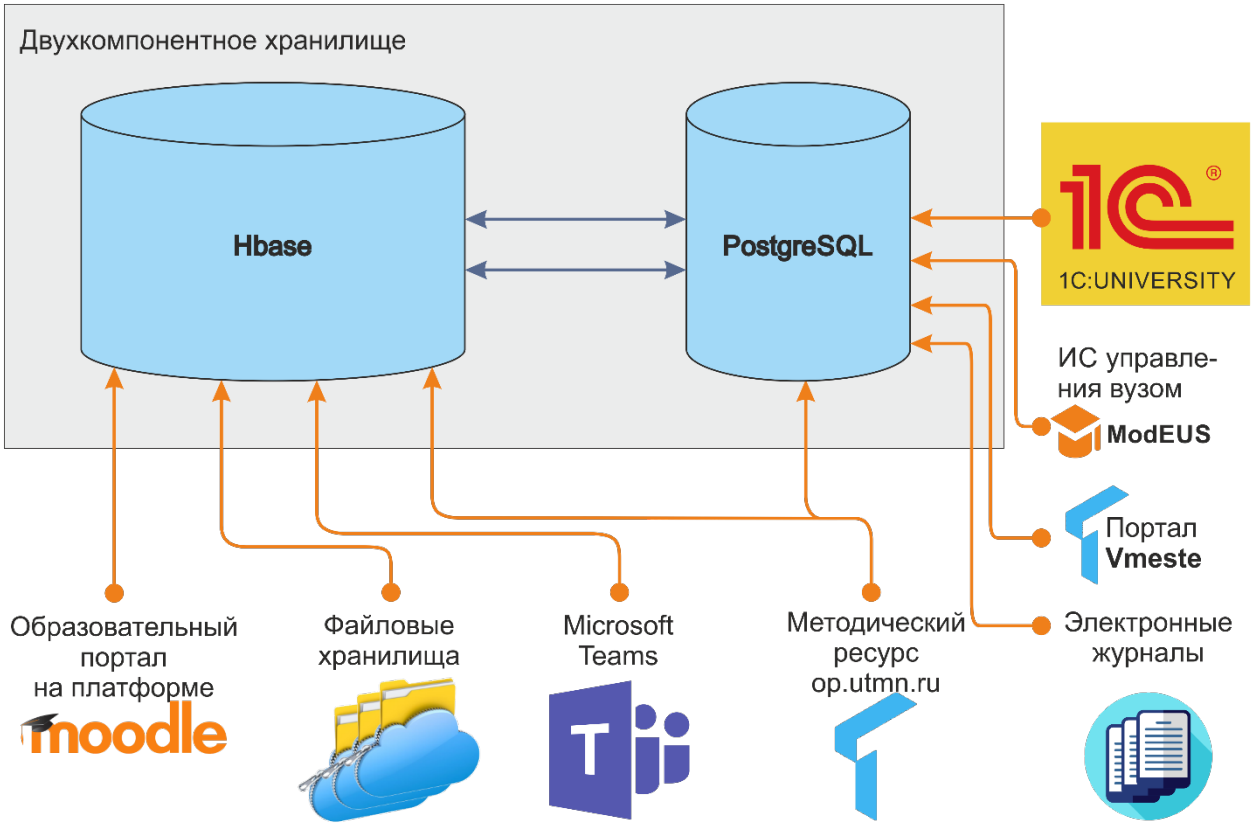

Рис. 2. Источники данных хранилища

Эти данные могут быть в форме ответов большого числа студентов на различные вопросы или в виде достаточно большого пула текстов учащихся, полученных от учебных команд, и другие общие описательные данные. Такую информацию, полученную в образовательном процессе, можно фактически разделить на два типа — структурированные данные и неструктурированные данные.

Структурированные данные уже организованы, поэтому вероятность того, что они будут слишком обширными и расплывчатыми, меньше. Такие данные не требуют пояснений и проще анализируются по сравнению с неструктурированными данными [11].

Неструктурированные данные поступают из разрозненных источников, не имеют предопределенной модели данных и требуют предварительной обработки перед выполнением анализа. Например, к таким данным можно отнести тексты работ и программный код.

Необходимые данные распределены по нескольким системам, часть из них хранится в виде отдельных файлов. Сбор данных осуществляется из источников, ресурсов и систем, которые используются в университете: 1С:Университет, Modeus, образовательный портал на платформе Moodle (elearning.utmn.ru), Microsoft Teams, портал «Вместе» (vmeste.utmn.ru), методический ресурс (op.utmn.ru), результаты опросов студентов, разные файловые хранилища.

Данные собираются с различной периодичностью:

- рабочие программы дисциплин (РПД) — в начале учебного года;
- тексты студенческих работ (курсовые, отчеты по практике, выпускные квалификационные работы) — раз в семестр;
- данные о текущей и итоговой успеваемости студентов — раз в месяц и раз в семестр;
- исходные коды программ, написанные студентами в рамках преподаваемых дисциплин и курсовых работ — раз в неделю.

Тексты студенческих работ

Формат файлов — документы Word или pdf-файлы. В имени файла содержатся данные об студенте, выполнившем работу. Метаданные генерируются на основе имени файла. При загрузке требуется определить идентификатор студента по входной текстовой информации, извлеченной из неструктурированного текста.

Рабочие программы дисциплин

Формат файлов — документы Word или pdf-файлы. Имя файла содержит полное или сокращенное название дисциплины. Метаданные создаются на основе названия файла и содержат сокращенное название дисциплины.

Данные об успеваемости студентов

Выгрузка из систем «1С: Университет» и «Модеус» в виде файлов формата Excel.

Исходные коды программ

Данные по кодам поступают из двух источников:

1. Выгрузка из образовательного портала Moodle (xml-формат).

2. Архивы исходных кодов, которые присылают студенты преподавателям на проверку.

1.3. ПРОЦЕСС ПОПОЛНЕНИЯ ХРАНИЛИЩА ДАННЫМИ И ПРЕОБРАЗОВАНИЕ ДАННЫХ

Весь процесс пополнения хранилища данными можно разделить на три этапа: получение данных, их преобразование и затем сохранение результата.

Данные могут загружаться из файлов в локальной или внешней папке, с помощью GET-запросов с web-ресурсов, посредством SQL-запросов из баз данных. При преобразовании данных может измениться их формат и структура. Результат преобразования данных сохраняется в реляционной или нереляционной части хранилища в зависимости от типа данных.

Процесс преобразования данных заключается в изменении формата или структуры данных. Этот процесс является основным в задачах интеграции и управления данными [12].

Сложность преобразования данных зависит от изменений, которые необходимо выполнить над данными. Преобразование может выполняться вручную или автоматически. Выбор инструментов и технологий зависит от формата, структуры и объема преобразовываемых данных.

Данные в систему могут поступать в различных форматах:

- простой текст (файлы формата *.txt);
- файлы формата PDF, содержащие текст;
- файлы формата PDF, содержащие отсканированные изображения;
- файлы формата MS Word;
- файлы формата MS Excel;
- данные формата JSON.

Далее данные, в зависимости от их назначения, необходимо конвертировать в требуемый формат.

Далее под текстами студенческих работ будем понимать следующие виды документов:

- выпускная квалификационная работа;
- курсовая по дисциплине;
- курсовая по направлению;
- курсовая по специальности;
- научно-исследовательская работа;
- отчет по преддипломной практике;
- отчет по производственной практике;
- программный код.

При поступлении данных в систему происходит распознавание формата входных данных и дальнейшее преобразование:

- данные формата MS Word сохраняются в исходном виде в файловом хранилище. Далее из файла формата MS Word извлекается плоский текст и помещается в таблицу NoSQL хранилища. Из текста извлекаются метаданные и помещаются в соответствующую таблицу реляционной СУБД;
- данные формата PDF, содержащие отсканированные изображения, проходят через модуль распознавания текста. Распознанный текст помещается в таблицу NoSQL хранилища. Из текста извлекаются метаданные и помещаются в соответствующую таблицу реляционной СУБД;
- плоский текст не подвергается преобразованиям, и просто переносится в соответствующую таблицу хранилища. Метаданные извлекаются аналогично предыдущим пунктам.

Извлечение метаданных

Метаданные — это структурированные данные, которые описывают характеристики некоторых других данных. Метаданные позволяют отбирать описываемые ими данные без анализа самих данных. При работе с текстами такой подход позволяет получать набор текстов только по их характеристикам.

При поступлении исходных текстов как студенческих работ, так и текстов учебных программ, необходимо извлечь определенные метаданные об этих

текстах, то есть получить некую структурированную информацию из неструктурированной, и сохранить их в реляционной СУБД.

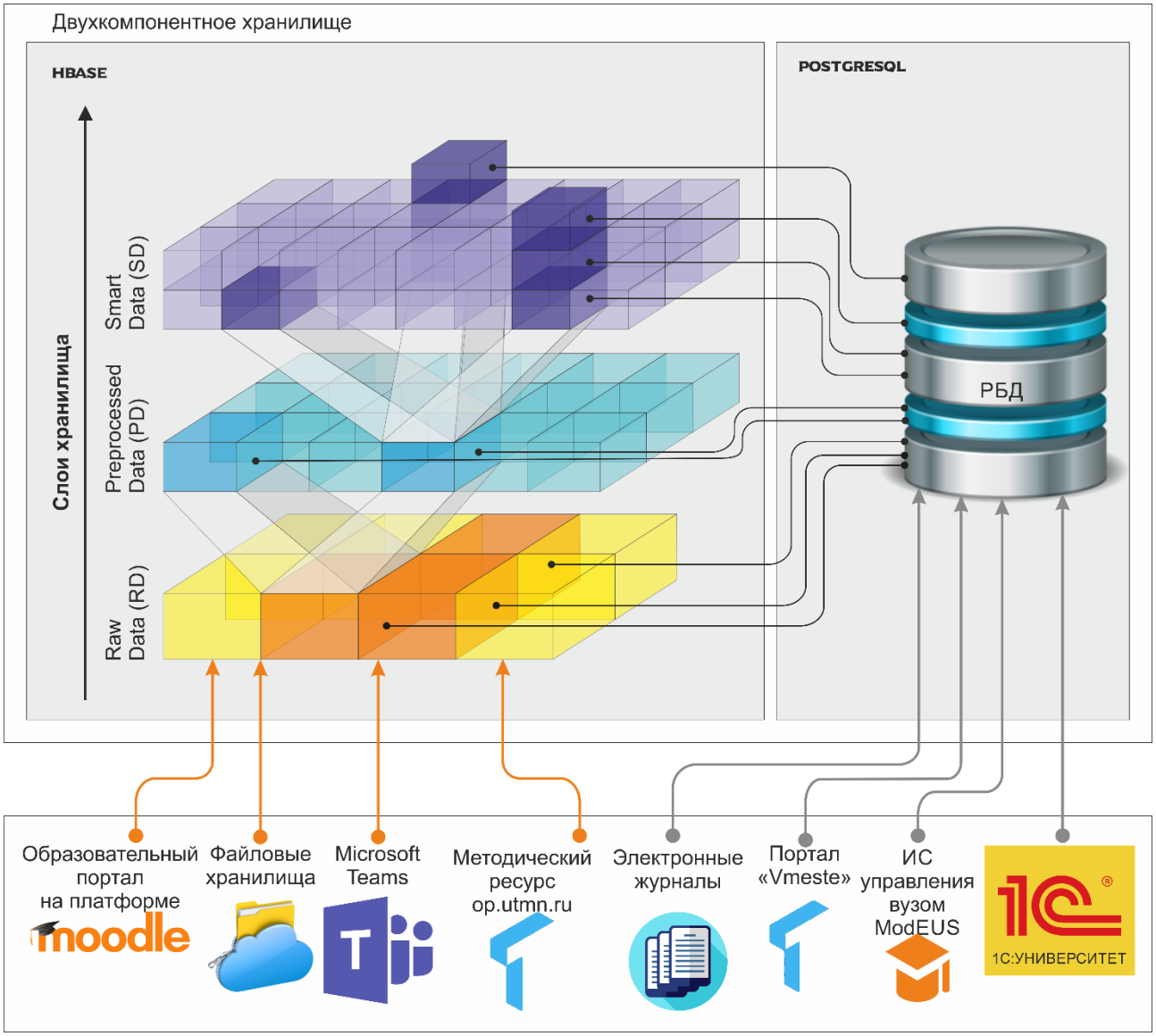

Рис. 3. Обобщенная схема хранилища

ГЛАВА 2. РЕАЛИЗАЦИЯ ХРАНИЛИЩА ДАННЫХ

2.1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В результате анализа предметной области, были выявлены сущности, которые необходимо сохранять в системе, а также взаимосвязи между ними (Рисунок 4).

Хранилище можно разделить на два различных по смыслу и по характеристикам блока. В итоге, имеется двухкомпонентное хранилище, состоящее из следующих частей (Рисунок 3):

- данные, имеющие четкую и редко изменяющуюся структуру, не превосходящие по объему 100 тыс. записей, размещены в реляционной СУБД;
- неструктурированные данные большого объема, которыми нужно довольно быстро оперировать, превосходящие по хранимому объему (в том числе и в перспективе) 100 тыс. записей, либо данные, имеющие изменчивую структуру, которую невозможно предугадать заранее, размещены в NoSQL базе данных.

Структурированные данные и метаданные будут хранить в реляционной базе данных. Для хранения этого типа данных была выбрана РСУБД PostgreSQL. Для хранения неструктурированных данных и данных с изменчивой структурой была выбрана Apache HBase.

2.2. АРХИТЕКТУРА ХРАНИЛИЩА

Диаграмма «Сущность-связь» реляционной части хранилища представлена на рисунке 5 и насчитывает 36 таблиц.

Рассмотрим реляционную часть хранилища (PostgreSQL). Далее приведено описание таблиц, размещенных в схеме public.

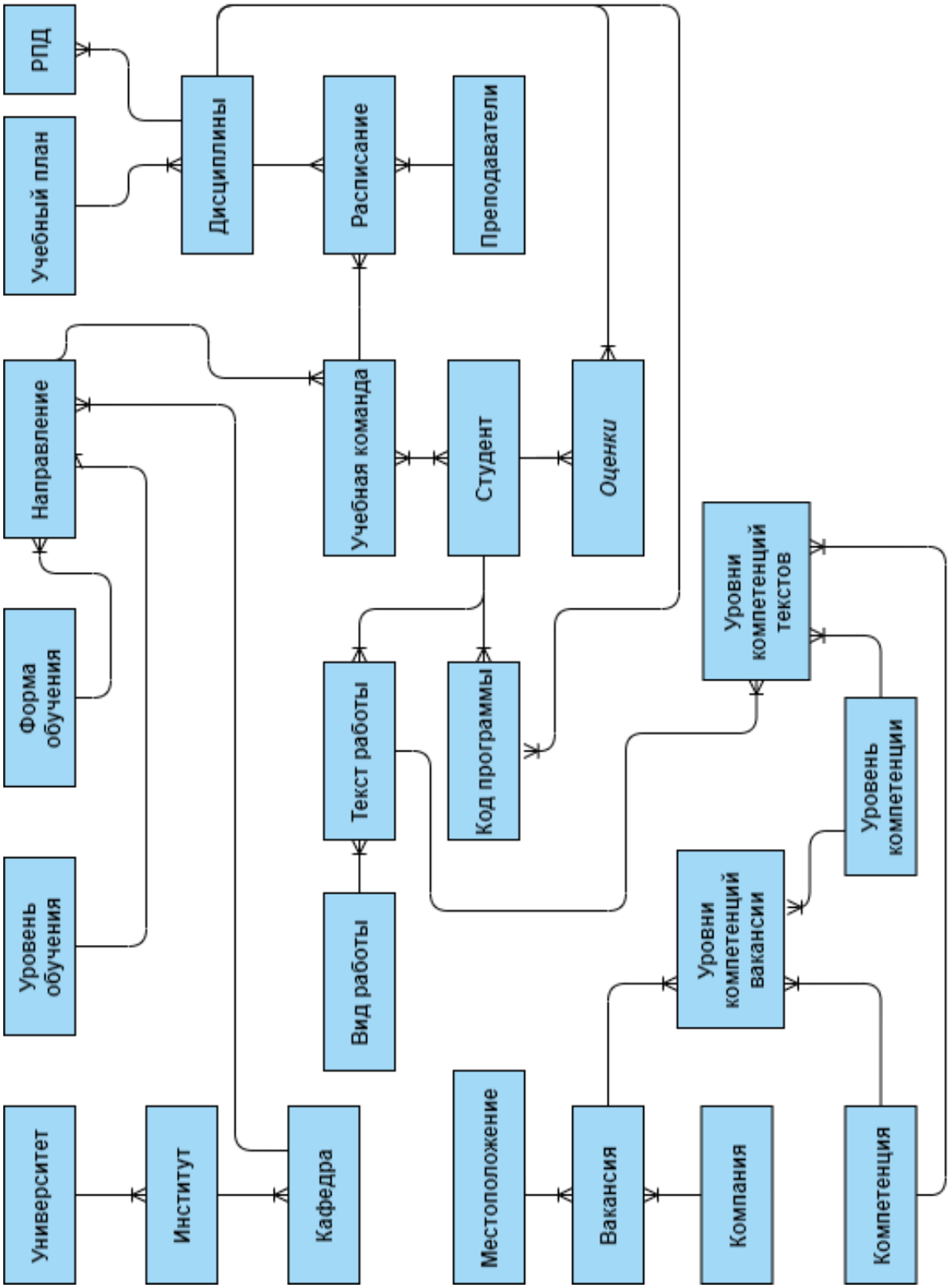

Рис. 4. Основные сущности модели данных и взаимосвязи между ними

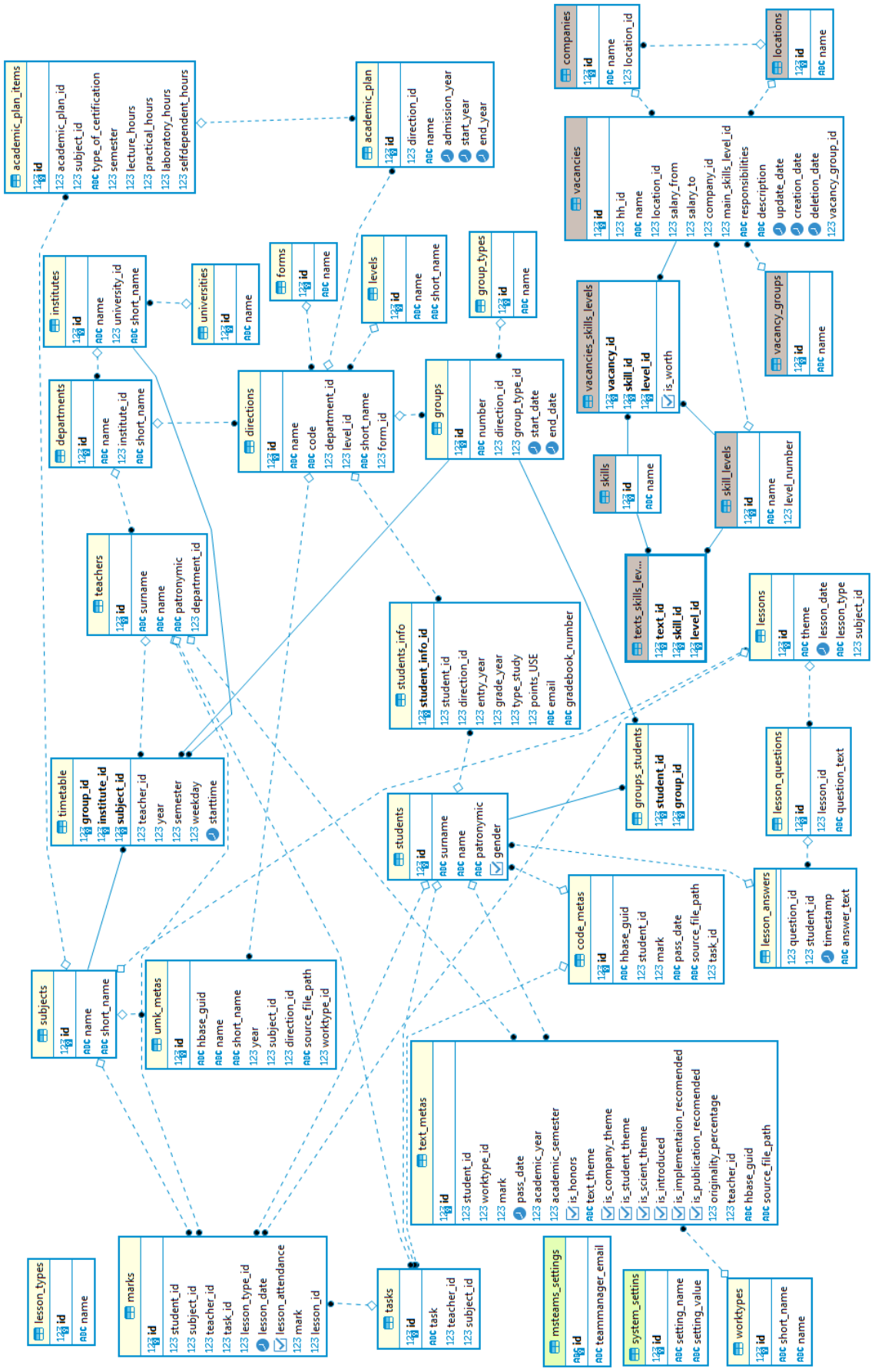

Рис. 5. ER-диаграмма реляционной части хранилища

Таблица 1

Таблица universities (Университеты)

Имя поля	Описание	Тип поля
id	id Университета	smallint NOT NULL
name	Наименование	character varying(255)

Таблица 2

Таблица institutes (Институты)

Имя поля	Описание	Тип поля
id	id Института	smallint NOT NULL
name	Наименование	character varying(255)
short_name	Краткое наименование	character varying(20)
university_id	id Университета	smallint

Таблица 3

Таблица departments (Кафедры)

Имя поля	Описание	Тип поля
id	id кафедры	integer NOT NULL
name	Наименование	character varying(255)
short_name	Краткое наименование	character varying(20)
institute_id	id института	smallint

Таблица 4

Таблица levels (Уровни обучения)

Имя поля	Описание	Тип поля
id	id уровня	smallint NOT NULL
name	Наименование	character varying(50)
short_name	Краткое наименование	character varying(10)

Таблица 5

Таблица directions (Направления)

Имя поля	Описание	Тип поля

id	id направления	integer NOT NULL
name	Наименование	character varying(255)
short_name	Краткое наименование	character varying(20)
code	Код направления	character varying(20)
department_id	id кафедры	integer
level_id	id уровня	smallint
form_id	id формы обучения	smallint

Таблица 6

Таблица forms (Форма обучения)

Имя поля	Описание	Тип поля
id	id формы обучения	smallint NOT NULL
name	Наименование	character varying(50)

Таблица 7

Таблица groups (Группы/Команды)

Имя поля	Описание	Тип поля
id	id группы	id integer NOT NULL
number	Номер группы	character varying(255)
direction_id	id направления	integer
group_type_id	id типа группы	smallint
start_date	дата начала обучения в команде	date
end_date	дата окончания обучения в команде	date

Таблица 8

Таблица group_types (Типы групп)

Имя поля	Описание	Тип поля
id	id типа группы	smallint NOT NULL
name	Наименование	character varying(25)

Таблица 9

Таблица groups_students (Студенты групп)

Имя поля	Описание	Тип поля
student_id	id студента	integer NOT NULL
group_id	id группы/команды	integer NOT NULL

Таблица 10

Таблица students (Студенты)

Имя поля	Описание	Тип поля
id	id студента	integer NOT NULL
gender	Пол	boolean
surname	Фамилия	character varying(100)
name	Имя	character varying(100)
patronymic	Отчетство	character varying(100)

Таблица 11

Таблица students_info (Информация о студенте)

Имя поля	Описание	Тип поля
student_info_id	id записи о студенте	integer NOT NULL
student_id	id студента	integer
direction_id	id направления	integer
entry_year	Год поступления	numeric(4,0)
grade_year	Год выпуска	numeric(4,0)
type_study	тип обучения	smallint
gradebook_number	Номер зачетной книжки студента	character varying(255)
email	адрес эл почты студента	character varying(255)
points_USE	баллы за ЕГЭ	smallint

Таблица 12

Таблица teachers (Преподаватели)

Имя поля	Описание	Тип поля

id	Id преподавателя	integer NOT NULL
surname	Фамилия	character varying(100)
name	Имя	character varying(100)
patronymic	Отчетство	character varying(100)
department_id	id кафедры	integer

Таблица 13

Таблица subjects (Дисциплины)

Имя поля	Описание	Тип поля
id	id дисциплины	integer NOT NULL
name	Наименование	character varying(300)
short_name	Краткое наименование	character varying(50)

Таблица 14

Таблица timetable (Расписание)

Имя поля	Описание	Тип поля
group_id	Группа	integer NOT NULL
institute_id	id института	integer NOT NULL
subject_id	Дисциплина	integer NOT NULL
teacher_id	Преподаватель	integer
year	Учебный год	numeric(4,0)
semester	Семестр	numeric(1,0)
weekday	день недели	numeric(1,0)
starttime	Время начала пары	timestamp

Таблица 15

Таблица tasks (Задания по предметам)

Имя поля	Описание	Тип поля
id	id задания	integer NOT NULL
task	Текст задания	text
teacher_id	Преподаватель	integer
subject_id	Дисциплина	integer

Таблица 16

Таблица academic_plan (Учебные планы)

Имя поля	Описание	Тип поля
id	Id плана	integer NOT NULL
direction_id	Id направления	integer
name	Наименование плана	character varying(255)
start_year	Год начала действия учебного плана	numeric(4,0)
end_year	Год окончания действия учебного плана	numeric(4,0)

Таблица 17

Таблица academic_plan_items (Состав учебного плана)

Имя поля	Описание	Тип поля
id	Id элемента плана	integer NOT NULL
academic_plan_id	id учебного плана	integer
subject_id	Id дисциплины	integer
semester	Семестр	numeric(1,0)
lecture_hours	Лекции, часы	smallint
practical_hours	Практики, часы	smallint
laboratory_hours	Лабораторные, часы	smallint
selfdependent_hours	Самостоятельная работа, часы	smallint
type_of_certification	Вид контроля	character varying(20)

Таблица 18

Таблица worktypes (Тип работы)

Имя поля	Описание	Тип поля
id	id типа работы	integer NOT NULL
name	Наименование полное	character varying(100)
short_name	Наименование краткое	character varying(20)

Таблица 19

Таблица text_metas (Метаданные текста)

Имя поля	Описание	Тип поля
id	id метаданных текста	integer NOT NULL
hbase_guid	GUID текста	character varying(255)
student_id	id студента	integer
worktype_id	id типа работы	smallint
mark	Оценка	numeric(1,0)
pass_date	Дата сдачи работы	date
academic_year	Учебный год выполнения работы	numeric(4,0)
academic_semester	Семестр выполнения работы	numeric(1,0)
is_honors	Диплом с отличием	boolean
text_theme	Тема работы	text
is_company_theme	Тема по заявке предприятия	boolean
is_student_theme	Тема по теме выпускника	boolean
is_scient_theme	Тема в области науч. иссл.	boolean
is_introduced	Внедрено	boolean
is_implementaion_recomended	Рекомендовано к внедрению	boolean
is_publication_recomended	Рекомендовано к опубликованию	boolean
originality_percentage	Процент оригинальности	double
teacher_id	id руководителя	integer
source_file_path	Путь к исходному файлу	text

Таблица 20

Таблица umk_metas (Метаданные рабочих программ дисциплин)

Имя поля	Описание	Тип поля
id	id РПД	integer NOT NULL
hbase_guid	GUID текста	character varying(255)
name	название РПД	character varying(255)
short_name	Краткое наименование	character varying(50)

year	год	numeric(4,0)
subject_id	дисциплина	integer
direction_id	направление	integer
source_file_path	Путь к исходному файлу	text
worktype_id	id типа РПД	smallint

Таблица 21

Таблица code_metas (Метаданные кода программы)

Имя поля	Описание	Тип поля
id	id кода	integer NOT NULL
hbase_guid	GUID текста	character varying(255)
student_id	id студента	integer
mark	Оценка/баллы	numeric(1,0)
task_id	id задания	integer
pass_date	Дата сдачи работы	Дата сдачи работы
source_file_path	Путь к исходному файлу	text

Таблица 22

Таблица marks (Оценки студентов)

Имя поля	Описание	Тип поля
id	id оценки	integer NOT NULL
student_id	id студента	integer
subject_id	id предмета	integer
teacher_id	id преподавателя	integer
group_id	id команды	integer
task_id	id задания	integer
lesson_date	дата оценки	date
lesson_id	id занятия	integer
lesson_type_id	тип занятия	smallint
lesson_mark	Оценка (кол-во заработанных баллов)	smallint
lesson_attendance	Присутствие на паре	boolean

Таблица 23

Таблица lesson_types (Тип занятия)

Имя поля	Описание	Тип поля
id	id типа занятия	smallint NOT NULL
name	Наименование	character varying(50)

Возможны следующие значения типов занятий:

1) Лекция
2) Практика
3) Контрольная неделя
4) Экзамен
5) Зачет
6) Число посещенных лекций
7) Число пропущенных лекций

Таблица 24

Таблица lessons (Занятия)

Имя поля	Описание	Тип поля
id	id занятия	integer NOT NULL
theme	Тема занятия	text
subject_id	id дисциплины	integer
lesson_date	Дата проведения занятия	date
lesson_type_id	id типа занятия	smallint

Таблица 25

Таблица lesson_questions (Вопросы теста занятия)

Имя поля	Описание	Тип поля
id	id вопроса	integer NOT NULL
lesson_id	id занятия	integer
question_text	текст вопроса	text

Таблица 26

Таблица lesson_answers (Ответы студентов на вопросы теста)

Имя поля	Описание	Тип поля
question_id	id вопроса	integer NOT NULL
student_id	id студента	integer NOT NULL
timestamp	дата и время ответа	timestamp
answer_text	текст ответа	text

Далее приведено описание таблиц, размещенных в схеме vacancies.

Таблица 27

Таблица companies (Компании)

Имя поля	Описание	Тип поля
id	id компании	integer NOT NULL
name	Наименование компании	character varying(255)
location_id	Основное местоположение (id)	integer

Таблица 28

Таблица skill_levels (Уровни компетенций)

Имя поля	Описание	Тип поля
id	id уровня компетенции	smallint NOT NULL
name	Наименование уровня компетенции	character varying(255)
level_number	Номер уровня	smallint

Таблица 29

Таблица skills (Компетенции)

Имя поля	Описание	Тип поля
id	id компетенции	integer NOT NULL
name	наименование компетенции	character varying(255)

Таблица 30

Таблица vacancies_skills_levels (Уровни компетенций вакансий)

Имя поля	Описание	Тип поля
vacancy_id	id вакансии	integer NOT NULL
skill_id	id компетенции	integer NOT NULL
level_id	id уровня компетенции	smallint NOT NULL
is_worth	обязательна ли данная компетенция для вакансии	boolean

Таблица 31

Таблица texts_skills_levels (Уровни компетенций текстов)

Имя поля	Описание	Тип поля
text_id	id текста	integer NOT NULL
skill_id	id компетенции	integer NOT NULL
level_id	id уровня компетенции	smallint NOT NULL

Таблица 32

Таблица locations (Местоположение)

Имя поля	Описание	Тип поля
id	id местоположения	integer NOT NULL
name	наименование местоположения	character varying(255)

Таблица 33

Таблица vacancy_groups (Группы вакансий)

Имя поля	Описание	Тип поля
id	id группы вакансий	integer NOT NULL
name	наименование группы вакансий	character varying(255)

Таблица 34

Таблица vacancies (Вакансии)

Имя поля	Описание	Тип поля
id	id вакансии	integer NOT NULL
hh_id	id вакансии в hh	integer
name	заголовок	character varying(255)
location_id	id местоположения	integer
salary_from	Зарплата от	numeric(10,2)
salary_to	Зарплата до	numeric(10,2)
company_id	id компании	integer
main_skillls_level_id	основной уровень вакансии	smallint
responsibilities	обязанности	text
description	описание	text
update_date	дата обновления	timestamp
creation_date	дата создания	timestamp
deletion_date	дата удаления	timestamp
vacancy_group_id	id группы вакансий	smallint

В базе данных существует 2 таблицы, предназначенные для хранения параметров системы, необходимых для функционирования процессов загрузки данных в хранилище (таблицы 35-36).

Таблица 35

Таблица system_settings (Таблица для хранения настроек системы)

Имя поля	Описание	Тип поля
id	id настройки	integer NOT NULL
setting_name	наименование параметра	character varying(20)
setting_value	значение параметра	character varying(255)

Таблица 36

Таблица msteams_settings (Команды MS Teams для сбора данных)

Имя поля	Описание	Тип поля
id	Уникальный идентификатор команды MS Teams	character (36)
teammanager_email	наименование параметра	character varying(255)

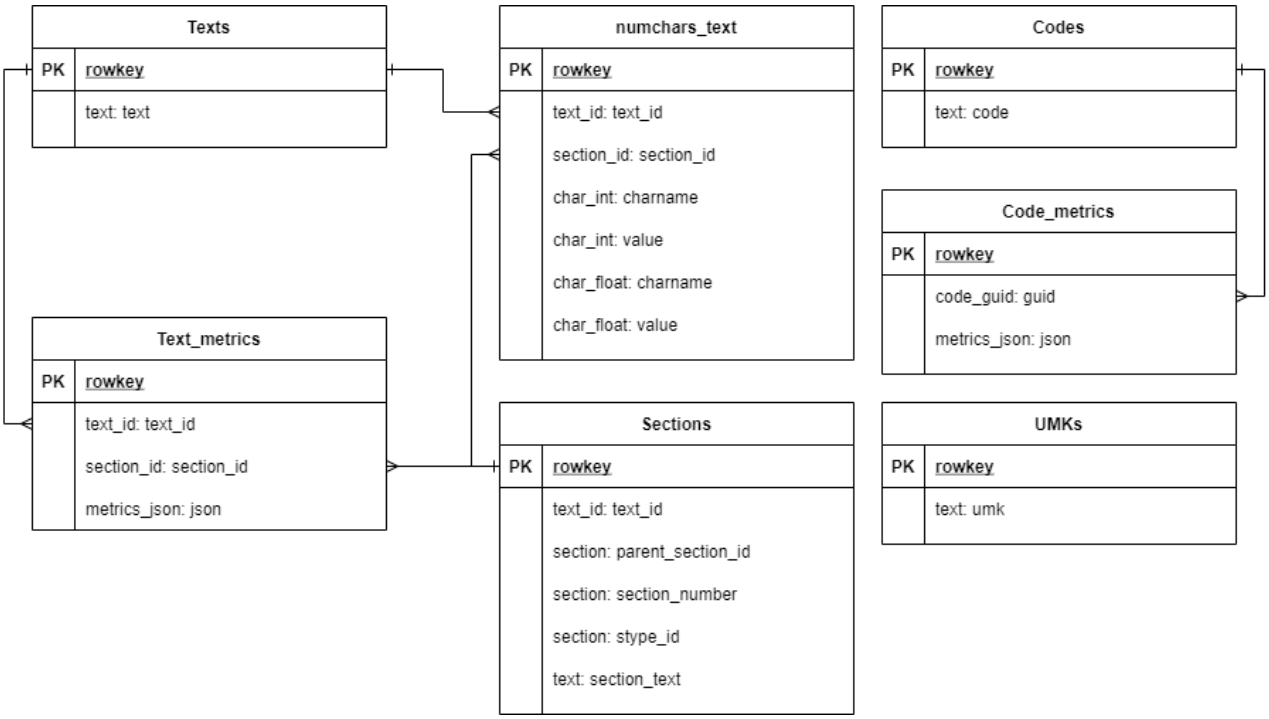

Рис. 6. ER- диаграмма нереляционной части хранилища

Рассмотрим нереляционную часть хранилища, реализованную в HBase (Рисунок 6). В этой СУБД содержатся таблицы, имеющие структуру, приведенную в таблицах 35 – 42. Так как любое поле таблицы в HBase является последовательностью байт и может быть интерпретировано как угодно, типы полей не указываются. Каждое поле таблицы (кроме rowkey) принадлежит семейству столбцов.

При создании таблицы в HBase создаются только семейства столбцов, колонки создаются в процессе вставки данных в таблицу, благодаря чему, таблица может быть дополнена нужными колонками без необходимости пересоздания структуры таблицы.

Таблица 35

Таблица texts (Тексты студенческих работ)

Семейство столбцов	Имя поля	Описание
	rowkey	id текста
text	text	Текст работы

Таблица 36

Таблица sections (Структура текста, его разделы)

Семейство столбцов	Имя поля	Описание
	rowkey	id секции
text_id	text_id	id текста
section	parent_section_id	id родительской секции
section	section_number	Номер секции
section	stype_id	Тип секции
text	section_text	Содержимое секции

Таблица 37

Таблица umks (Тексты рабочих программ дисциплин)

Семейство столбцов	Имя поля	Описание
	rowkey	id РПД
text	umk	Текст РПД

Таблица 38

Таблица codes (Коды студенческих программ)

Семейство столбцов	Имя поля	Описание
	rowkey	id кода
text	code	Программный код

Таблица 39

Таблица numchars_text (Числовые характеристики текста)

Семейство столбцов	Имя поля	Описание
	rowkey	id записи
text_id	text_id	id текста
text_id	section_id	id секции
char_int	charname	Наименование характеристики
char_int	value	Значение характеристики
char_float	charname	Наименование характеристики
char_float	value	Значение характеристики

Таблица 40

Таблица text_metrics (Метрики текста)

Семейство столбцов	Имя поля	Описание
	rowkey	id метрики
text_id	text_id	id текста
section_id	section_id	Id секции
metrics_json	metrics_json	Рассчитанные метрики текста в формате json

Таблица 41

Таблица code_metrics (Метрики программного кода)

Семейство столбцов	Имя поля	Описание
	rowkey	id кода
metrics_json	metrics_json	Рассчитанные метрики кода в формате json

Версионность полей таблиц настроена таким образом, что для каждого поля, кроме guid, в таблице HBase хранится 3 последних версии его значения, и дата изменения каждой версии.

Важной особенностью хранилища является взаимосвязь реляционной и NoSQL частей хранилища посредством ключевых полей типа guid: зная guid текста студенческой работы или программного кода из HBase можно найти метаданные этого текста и кода в реляционной части — PostgreSQL.

ГЛАВА 3. РЕАЛИЗАЦИЯ ИНФРАСТРУКТУРЫ РЕШЕНИЯ

3.1. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ И ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

Для развертывания системы была предоставлена виртуальная машина (сервер) на платформе VMware со следующими характеристиками:

- 10 ядер Intel Xeon X5670 с частотой 2.93GHz;
- 20GB оперативной памяти;
- 500GB жесткий диск.

На сервере установлена система из семейства Linux — CentOS 7. Доступ к серверу выполняется с помощью ssh-подключения.

Для управления контейнерами с приложениями используются Docker и Docker-compose.

Docker — это открытая платформа для разработки, доставки и запуска приложений.

Docker-compose — это инструмент для описания и запуска многоконтейнерных приложений Docker.

Используется OpenVPN и iptables для обеспечения доступа разработчиков к инфраструктуре.

OpenVPN — это система виртуальной частной сети (VPN), которая реализует методы для создания безопасных подключений к сети.

Iptables используется для настройки, обслуживания и проверки таблиц правил фильтрации пакетов IPv4 в ядре Linux.

OpenVPN позволяет получать доступ к инфраструктуре на сервере, а с помощью iptables определяются ресурсы, к которым предоставляется доступ. OpenVPN предоставляет возможность организовать доступ по персональным сертификатам, что позволяет при необходимости отозвать сертификат и прекратить доступ к серверу определенному пользователю. Iptables задает правила, которые разрешают доступ из vpn-сети только к подсети, используемой docker-контейнерами.

3.2. КОМПОНЕНТЫ СИСТЕМЫ

Обобщенно система состоит из следующих компонентов:

- backend;
- frontend;
- хранилище данных;
- процедуры ETL;
- сервис аутентификации.

Рассмотрим каждый из элементов системы и его техническую реализацию подробнее.

Backend

Серверное приложение, которое выполняет запросы от frontend. Оно получает необходимые данные из хранилища, выполняет их обработку и возвращает ответ.

Frontend

Клиентское веб-приложение, которое предоставляет возможность взаимодействия с системой. Отображает данные в интерфейсе, оправляет запросы на backend, получает ответы.

Хранилище данных

Многоуровневое хранилище данных, реализующее возможности чтения и записи данных.

Процедуры ETL

Процессы, служащие для пополнения хранилища данными, включающие загрузку сырых данных, преобразование к различным форматам, извлечение метаданных, запись данных в таблицы реляционной и NoSQL части хранилища [13].

Сервис аутентификации

Сервис управления доступом, позволяет определять права доступа на просмотр и запрос данных.

Для развертывания системы на сервере применяются контейнеры. Контейнер — это единица программного обеспечения, которая упаковывает код

и все его зависимости, поэтому приложение быстро и надежно перемещается из одной вычислительной среды в другую.

Каждому контейнеру задан ip-адрес из одной подсети, что позволяет разработчикам получать доступ к контейнерам через vpn-сеть.

На данный момент система состоит из следующих элементов (каждый из которых является контейнером), взаимосвязанных между собой:

- PostgreSQL;
- Hadoop + HBase;
- Python backend;
- C# backend;
- Nginx + Vue.js frontend;
- Keycloak;
- Hue;
- Apache NiFi.

Код файла docker-compose для развертывания контейнеров приведен в приложении 1. Схема взаимодействия контейнеров представлена на рисунке 7.

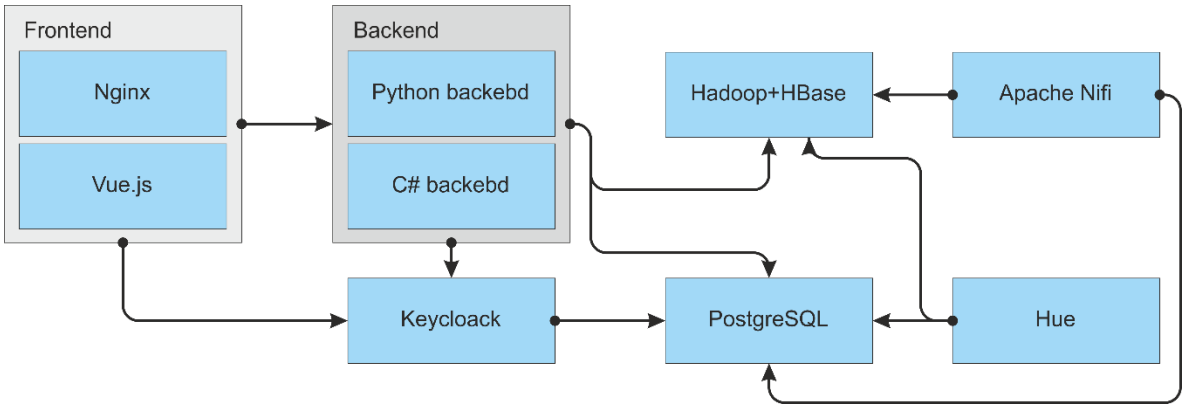

Рис. 7. Схема взаимодействия контейнеров

PostgreSQL

PostgreSQL — это мощная система объектно-реляционной базы данных с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании с множеством функций, обеспечивающих безопасное хранение и

масштабирование самых сложных рабочих нагрузок с данными. Используется для хранения структурированных данных и метаданных.

Hadoop и HBase

Hadoop — это программная среда с открытым исходным кодом для хранения данных и запуска приложений. Она обеспечивает хранение любых типов данных в HDFS и способность обрабатывать практически неограниченное количество одновременных задач. Позволяет выполнять масштабирование за счет добавления новых узлов в кластер [14].

HBase — это система управления нереляционными базами данных, ориентированная на столбцы, которая работает поверх распределенной файловой системы HDFS. HBase обеспечивает отказоустойчивый способ хранения разреженных наборов данных. Используется для хранения неструктурированных данных и результатов их анализа.

Python backend

Приложение разработано с использованием библиотеки FastAPI, которая позволяет создавать API-интерфейсы на Python. Для взаимодействия с PostgreSQL применяется библиотека SQLAlchemy. Библиотека happybase позволяет работать с HBase. Для получения спецификации API применяется Swagger.

C# backend

Приложение, реализованной с применением технологии ASP.NET. ASP.NET — это серверная платформа с открытым исходным кодом, предназначенная для разработки приложений и сервисов. Swagger используется для получения спецификации API.

Nginx

Nginx — это бесплатный высокопроизводительный HTTP-сервер с открытым исходным кодом. Он известен своей высокой производительностью, стабильностью, богатым набором функций, простой конфигурацией и низким потреблением ресурсов.

Vue.js frontend

Для реализации клиентской части приложения используется vue.js. Vue.js — это JavaScript-фреймворк с открытым исходным кодом, используемый для разработки интерактивных веб-интерфейсов. Используется язык программирования TypeScript, html-шаблонизатор pug и css-препроцессор stylus. Для генерации запросов к методам API применяется Swagger codegen.

Keycloak

Keycloak — это приложение с открытым исходным кодом, предназначенное для управления учетными записями и правами доступа в приложениях и сервисах. Отвечает за возможность получения доступа к клиентской части приложения. Проверяет права на выполнение запросов к api.

Hue

Hue — это приложение с открытым исходным кодом для запросов к базам данных и хранилищам данных. Hue является частью платформы обработки данных Cloudera и сервисов инфраструктуры Hadoop. Предоставляет возможность просмотра списка таблиц в HBase, а также просмотра и редактирования данных в этих таблицах. Позволяет взаимодействовать с файловой системой HDFS.

Apache NiFi

Apache NiFi — это система управления потоками данных, которая позволяет выполнять конфигурирование с помощью веб-интерфейса.

Apache Spark

Apache Spark — фреймворк с открытым исходным кодом для реализации распределенной обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop, который имеет API на Java, Scala, Python и R. В системе используется для выполнения задач, поступающих из обработчиков в NiFi.

3.3. РАЗВЕРТЫВАНИЕ КОМПОНЕНТОВ ЭКОСИСТЕМЫ HADOOP

Однонодовый тестовый кластер Hadoop был развернут с помощью docker-контейнера [15–17]. Код docker-файла приведен в приложении 2. В качестве

базового образа используется CentOS 7. При создании docker-контейнера в нем устанавливается Java 8, пакеты для работы с ssh. Затем скачиваются архивы с бинарными файлами Hadoop, HBase и Spark, и выполняется их разархивирование в советующие папки. После выполняется копирование файлов настроек и форматирование HDFS. Создаются необходимые пользователи, папки для логов и задаются права пользователей на эти папки.

При запуске выполняется скрипт, конфигурирующий ssh-подключение между созданными пользователями с помощью ключей. Затем последовательно выполняются скрипты, запускающие HDFS, HBase, Spark-master и Spark-worker.

За конфигурирование компонентов кластера отвечают следующие файлы:

- core-site.xml — основной настроек Hadoop. Содержит путь до HDFS и настройки прав доступа.
- hdfs-site.xml — файл настроек HDFS. В нем задается количество копий HDFS, отслеживание прав доступа к файлам, пути к папкам с данными DataNode и NameNode.
- hbase-site.xml — файл конфигурации HBase. В нем задается имя хоста, путь до HDFS, распределенный режим работы.
- spark-defaults.conf содержит настройки Spark. В файле определяется адрес spark-master.

Статус запущенных приложений можно проверить с помощью веб-страниц, которые доступны в контейнере по следующим портам:

1. 9870 — порт для доступа к странице со статусом NameNode;
2. 9864 — порт для доступа к странице со статусом DataNode;
3. 16010 — порт для доступа к странице со статусом HBase Master;
4. 16030 — порт для доступа к странице со статусом HBase Region server;
5. 8082 — порт для доступа к странице со статусом Spark-master;
6. 8083 — порт для доступа к странице со статусом Spark-worker.

Доступ к приложениям в контейнере можно получить по следующим портам:

- 8020 — порт для доступа к HDFS;
- 9870 — порт для доступа к WebHDFS;
- 9090 — порт для доступа к Thrift API HBase;
- 7077 — порт для доступа к Spark-master.

3.4. ПРИМЕНЕНИЕ APACHE NIFI ДЛЯ ПОПОЛНЕНИЯ ХРАНИЛИЩА ДАННЫМИ

Для решения задач реализации и мониторинга ETL-процессов, необходимых для функционирования системы, был выбран инструмент Apache NiFi. Это простая платформа обработки событий (сообщений), предоставляющая возможности управления потоками данных из разнообразных источников в режиме реального времени с использованием графического интерфейса.

Apache NiFi имеет богатый графический веб-интерфейс для создания и управления потоками данных, более 260 процессоров и коннекторов из коробки, что позволяет его использовать для сопряжения практически с любыми типами источников и потребителей данных.

В данном инструменте можно выделить четыре основных компонента: файлы потока, обработчики, соединения и контроллеры [18].

Файл потока — это один фрагмент данных, который состоит из двух компонентов: атрибутов и содержимого. Каждый файл потока имеет стандартные атрибуты: универсальный идентификатор (uuid), имя файла и путь до файла на диске или внешней службе.

Обработчик отвечает за прослушивание входящих данных, извлечение данных из внешних источников, публикацию данных во внешних источниках, а также маршрутизацию, преобразование или извлечение информации из файлов потока.

Соединения выполняют передачу данных от обработчика к обработчику.

Контроллеры позволяют создавать конфигурации, которые могут использоваться в обработчиках.

Рассмотрим основные используемые контроллеры и их параметры.

CSVReader — выполняет чтение csv-файла и возвращает каждую строку как отдельную запись. Имеет следующие основные параметры:

- CSV Parser — используемый парсер для получения данных из файла (Apache Commons CSV, Jackson CSV);
- CSV Format — csv-формат файла;
- Value Separator — разделитель в csv-файле;
- Record Separator — разделитель получаемых записей;
- Treat First Line as Header — использовать первую строку как заголовок;
- Quote Character — символ кавычки;
- Escape Character — символ экранирования.

DBCPConnectionPool — пул подключений к базе данных. Имеет следующие основные параметры:

- Database Connection URL — строка подключения к БД;
- Database Driver Class Name — название класса драйвера подключения.

JsonPathReader — выполняет чтение json-файла и возвращает указанное поле из него. Имеет следующие основные параметры:

- Schema Access Strategy — схема, используемая для интерпретации данных;
- The field name for the record — поле json-объекта, которое необходимо извлечь.

HBase_2_ClientService — задает подключение к HBase. Основной параметр: Hadoop Configuration Files — путь до конфигурационных файлов HBase.

ScriptedReader — выполняет чтение данных с помощью скрипта на заданном языке программирования. Имеет следующие основные параметры:

- Script Engine — движок выполнения скрипта (Clojure, ECMAcript, Groovy, lua, python, ruby);
- Script File — путь до файла со скриптом;
- Script body — код скрипта;
- Module Directory — путь до папки, содержащей необходимые модули для выполнения скрипта.

XMLReader — выполняет чтение данных из xml-файла. Имеет следующие основные параметры:

- Schema Access Strategy — схема, используемая для интерпретации данных;
- Expect Records as Array — читать файл как массив записей (истина, ложь, использовать атрибут файла).

Рассмотрим основные используемые обработчики и их параметры.

ConvertRecord — выполняет преобразование данных файла потока из одного формата в другой. Имеет следующие основные параметры:

- Record Writer — контроллер записи результата в файл потока;
- Record Reader — контроллер для чтения данных из файла потока;
- Include Zero Record FlowFiles — необходимо ли выполнять обработку пустых файлов потока и передавать их далее (истина, ложь).

DeleteHBaseRow — удаление строки в HBas. Имеет следующие основные параметры:

- HBase Client Service — контроллер подключения к HBase;
- Table Name — название таблицы;
- Row Identifier — идентификатор строки;
- Row ID Location — место хранения идентификатора строки в файле потока (содержимое файла потока, атрибуты файла потока).

ExecuteScript — выполняет обработку файла потока с помощью заданного скрипта. Имеет следующие основные параметры:

- Script Engine — движок выполнения скрипта (Clojure, ECMAcript, Groovy, lua, python, ruby);
- Script File — путь до файла со скриптом;
- Script body — код скрипта;
- Module Directory — путь до папки, содержащей необходимые модули для выполнения скрипта.

ExecuteSQL — выполнение select-запроса из базы данных. Имеет следующие основные параметры:

- Database Connection Pooling Service — контроллер пула подключений к БД;
- SQL select query — sql запрос.

ExecuteSQLRecord — выполнение select-запроса из базы данных и преобразование результата запроса к заданному формату с помощью контроллера. Имеет следующие основные параметры:

- Database Connection Pooling Service — контроллер пула подключений к БД;
- SQL select query — sql запрос;
- Record Writer — контроллер записи результата в файл потока.

FetchFile — чтение файла с диска или потока. Имеет следующие основные параметры:

- File to Fetch — путь до файла;
- Completion Strategy — действие после обработки файла (нет дейтсвия, удалить, переместить);
- Move Destination Directory — папка для перемещения файлов;
- Move Conflict Strategy — действие в случае, если при перемещении файла в папке файл с таким именем уже существует (переименовать, заменить, сохранить имеющийся, ошибка).

FetchHBaseRow — извлечение строки из таблицы в HBase. Имеет следующие основные параметры:

- HBase Client Service — контроллер подключения к HBase;
- Table Name — название таблицы;
- Row Identifier — идентификатор строки.

GetFile — создание файлов потока из файлов в папке. Имеет следующие основные параметры:

- Input Directory — путь до папки;
- File Filter — фильтр названий файлов по регулярному выражению;
- Batch Size — количество обрабатываемых файлов за одну итерацию;
- Keep Source File — сохранять ли файл после обработки.

GetHBase — извлечение данных из таблицы в HBase. Имеет следующие основные параметры:

- HBase Client Service — контроллер подключения к HBase;
- Table Name — название таблицы.

GetHTTP — выполнение GET-запросов. Имеет следующие основные параметры:

- URL — url-адрес сервера;
- File Name — имя файла, в который будет извлечены данные.

InvokeHTTP — выполнение HTTP-запросов. Имеет следующие основные параметры:

- HTTP Method — метод HTTP-запроса (GET, POST);
- Remote URL — url-адрес сервера.

ListenHTTP — запуск локального HTTP-сервера. Имеет следующие основные параметры:

- Base Path — путь для входящий подключений;
- Listening Port — прослушиваемый порт.

ListFile — получение списка файлов в папке. Имеет следующие основные параметры:

- Input Directory — папка для получения файлов;

- Listing Strategy — способ определения изменений (по времени изменения, по изменению за заданное количество минут);
- Recurse Subdirectories — нужно ли просматривать подпапки;
- Input Directory Location — тип папки (локальная, удаленная);
- File Filter — фильтр названий файлов по регулярному выражению.

PostHTTP — выполнение POST-запросов. Имеет следующие основные параметры:

- URL — url-адрес сервера;
- Send as FlowFile — отправлять как файл потока (истина, ложь);
- Content-Type — mime-тип отправляемых данных.

PutFile — сохранение файла. Имеет следующие основные параметры:

- Directory — папка для сохранения;
- Conflict Resolution Strategy — разрешение конфликтов (пропуск, замена, ошибка);
- Create Missing Directories — создавать ли несуществующие папки.

PutHBaseRecord — запись в HBase. Имеет следующие основные параметры:

- Record Reader — контроллер чтения из файла потока;
- HBase Client Service — контроллер подключения к HBase;
- Table Name — название таблицы;
- Row Identifier Field Name — название поля с идентификатором во входных данных.

PutSQL — запись в базу данных с помощью insert-запроса. Имеет следующие основные параметры:

- JDBC Connection Pool — контроллер подключения к БД;
- SQL Statement — sql запрос для выполения;
- Batch Size — количество запросов, выполняемых за один раз.

ScriptedTransformRecord — преобразование данных с помощью скрипта. Имеет следующие основные параметры:

- Record Reader — контроллер для чтения данных из файла потока;
- Record Writer — контроллер для записи данных в файл потока;
- Script Language — язык скрипта (Clojure, ECMAcript, Groovy, lua, python, ruby).

В системе были реализованы следующие etl-процессы:

- загрузка текстовых работ студентов;
- загрузка программных кодов студентов;
- загрузка текстов РПД;
- загрузка ведомостей с оценками;
- загрузка результатов опросов студентов;
- загрузка учебных планов.

Три первых процесса загружают текстовую информацию в хранилище, и схожи по своему принципу работы, за исключением того, что текстовые работы и коды учебных программ загружаются из команд MS Teams, а тексты РПД – из папки файлового хранилища на сервере.

Рассмотрим процесс загрузки текстовой информации в хранилище на примере загрузки файлов из MS Teams в хранилище с помощью Apache NiFi, приведенный на рисунке 8. По идентификаторам команд, полученным из таблицы msteams_settings, размещенной в PostgreSQL, выполняется извлечение прикрепленных файлов к сданному заданию с помощью Microsoft Graph API [19]. Затем выполняется преобразование файлов в текстовый формат и извлечение метаданных. После этого текст работы сохраняется в HBase, а метаданные в PostgreSQL. В случае ошибки производится отправка POST-запроса на сервер с данными об ошибке.

В случае, если загрузчик обнаружит исправленную версию текста, который уже был ранее загружен в хранилище, загрузка произойдет повторно. В этом случае, в HBase, благодаря настройкам версионности, сохранится очередная версия этого текста (но не более трех версий).

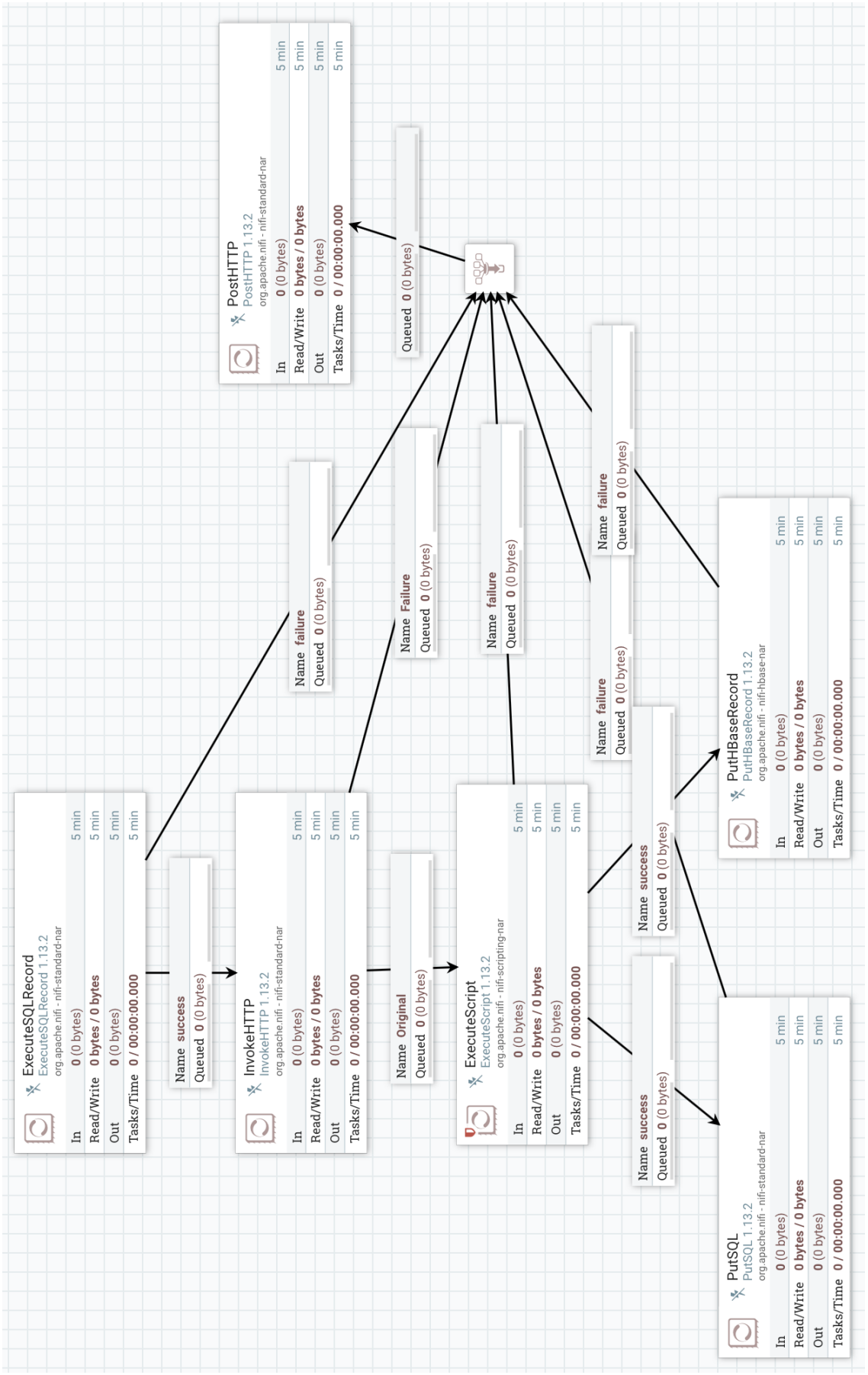

Рис. 8. Процесс загрузки заданий из MS Teams при помощи Apache Nifi

Рассмотрим функции обработчиков в этом процессе подробнее.

Обработчик ExecuteSQLRecord извлекает из PostgreSQL идентификаторы команд, из которых требуется выгрузить файлы, прикрепленные к сданному заданию.

Обработчик InvokeHTTP по полученным идентификаторам в MS Teams запрашивает с помощью Microsoft Graph API список сданных работ.

Обработчик ExecuteScript для каждой сданной работы скачивает прикрепленные файлы и выполняет при необходимости преобразование файла. Затем извлекает текст из файла и метаданные о файле.

Обработчик PutSQL сохраняет метаданные в PostgreSQL, а обработчик PutHBaseRecord сохраняет в HBase текст работы.

Информирование о возникающих ошибках выполняется с помощью обработчика PostHTTP, который выполняет оправку данных об ошибке на сервер по заданному адресу.

Можно выделить еще одну категорию etl-процессов, которые относятся к сервисам загрузки структурированной информации: загрузка ведомостей, учебных планов, результатов опросов студентов. Данная информация загружается из файлов формата MS Excel и имеют относительно хорошую структуру представленных в них данных.

Тем не менее, эти данные не могут быть загружены без предварительной обработки, т.к. необходимо по извлеченной текстовой информации определить идентификаторы основных сущностей, загружаемых в систему: id студента, id дисциплины, id направления обучения и т.д.

Рассмотрим процесс загрузки структурированной информации в хранилище на примере импорта ведомостей с помощью Apache NiFi (Рисунок 9). Ведомости хранятся в таких системах вуза, как «1С: Университет» и «Modeus». Данные предварительно выгружаются из этих систем и сохранятся в папке на сервере. Каждая ведомость обрабатывается и результат обработки сохраняется в хранилище. В случае ошибки производится отправка POST-запроса на сервер с данными об ошибке.

Рассмотрим функции обработчиков в этом процессе подробней.

Обработчик ListFile извлекает файлы из папки с ведомостями.

Обработчик ExecuteScript для каждой ведомости извлекает список студентов и группу. Если в ведомости встретился студент, который не найден в базе, он автоматически создается.

Если встретилась группа, которой еще нет в базе, то она создается. Группа понимается здесь в широком смысле: это может быть номер группы, указанный в системе «1С: Университет», либо команда, если загружается ведомость из системы «Modeus».

Таким образом хранилище пополняется данными о студентах и группах. Если такая ведомость уже была загружена, то данные по ней обновляются. Дальше передается данные из ведомости, которых не были ранее загружены в хранилище.

Обработчик PutSQL сохраняет данные по студенту из ведомости в PostgreSQL.

Информирование о возникающих ошибках выполняется с помощью обработчика PostHTTP, который выполняет оправку данных об ошибке на сервер по заданному адресу. Адресат, которому направляется сообщение, хранится в таблице system_tables базы данных PostgreSQL и извлекается оттуда при отправке оповещения.

Процессы загрузки результатов опросов студентов и учебных планов используют аналогичный подход и имеют подобные представленному на рисунке 9 процессы.

В процессе загрузки учебных планов происходит пополнение справочника дисциплин. Данный справочник в дальнейшем используется при загрузке текстов РПД для того, чтобы связать их с учебным планом.

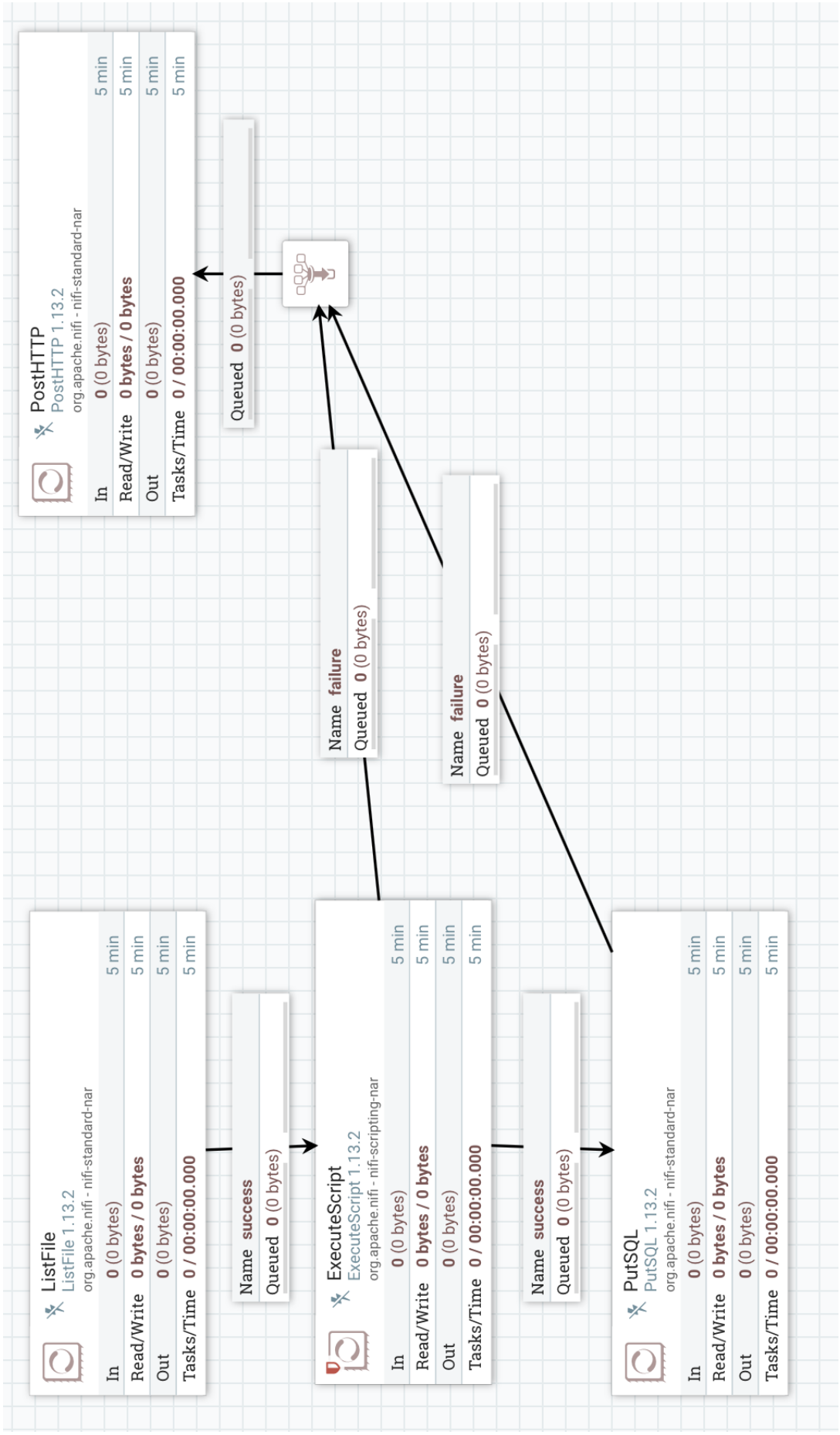

Рис. 9. Процесс загрузки ведомостей при помощи Apache Nifi

3.5. АВТОМАТИЗАЦИЯ ЗАДАЧ РАБОТЫ С ХРАНИЛИЩЕМ

При разработке системы используются git и github, что позволяет вести совместную работу над кодом.

Git — это программное обеспечение для отслеживания изменений в любом наборе файлов, обычно используемое для координации работы программистов, совместно пишущих исходный код во время разработки программного обеспечения.

GitHub — это веб-платформа для управления версиями и совместной работы разработчиков программного обеспечения. Она предоставляет репозиторий кода git и инструменты управления для совместной работы.

Для подключения к vpn-сети требуется набор персональных конфигурационных файлов, который необходимо выдать каждому разработчику сервисов для доступа к инфраструктуре. Подготовка такого комплекта требует определенных ресурсов: необходимо сгенерировать сертификат, взять ключи аутентификации, создать конфигурационный файл и затем упаковать все в архив. Для автоматизации этих действий был разработан bash-скрипт. Скрипт принимает на вход уникальный идентификатор пользователя и генерирует архив с необходимыми файлами. Код скрипта приведён в приложении 3. Результатом работы скрипта является комплект файлов для разработчика.

Необходимо обеспечить периодическое создание и хранение нескольких последних резервных копий базы данных PostgreSQL [20]. Для решения этой задачи был написан bash-скрипт. В процессе работы скрипт оставляет в специально определенной папке резервные копии не старше семи дней. Далее используется утилита pg_dumpall, предназначенная для записи всех баз данных в один файл в формате sql-скрипта. Полученный скрипт сжимается утилитой gzip. Резервное копирование выполняется по расписанию каждую ночь утилитой cron.

Разработку сервисов в системе выполняют несколько разработчиков. В процессе работы им требуется доступ к инфраструктуре хранилища. Для получения доступа разработчик обращается к администратору хранилища.

Администратор хранилища с помощью скрипта генерирует персональный набор конфигурационных файлов для подключения к внутренней сети vpn.

Используя полученный набор файлов, разработчик выполняет подключение к OpenVPN серверу с помощью ПО OpenVPN GUI версии 2. Установив подключение с vpn-сервером, разработчик получает доступ к внутренней сети на сервере.

Во внутренней сети разработчик имеет доступ к:

- СУБД PostgreSQL;
- сервисам Hadoop;
- HBase;
- NiFi;
- веб-приложению Hue;
- сервису авторизации Keycloak;
- развернутым backend приложениям.

С помощью графических инструментов управления базами данных можно подключиться к PostgreSQL.

Для взаимодействия с HBase используется интерактивный режим python и библиотека happybase. Также можно использовать web-приложение Hue для просмотра таблиц и данных в HBase.

Backend на python содержит необходимые реализации методов подключения к PostgreSQL с помощью SQLAlchemy и к HBase с использованием happybase.

Рассмотрим пример работы с PostgreSQL в листинге 1. Сначала выполняется создание подключения методом create_engine с помощью строки подключения из настроек. Затем создается метод получения сессий с помощью sessionmaker. Использование метода get_db позволяет получать подключения к PostgreSQL для дальнейшего использования.

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
```

```
from config import settings

engine = create_engine(settings.SQLALCHEMY_DATABASE_URI,
pool_pre_ping=True)
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)

def get_db() -> Generator:
    try:
        db = SessionLocal()
        yield db
    finally:
        db.close()
```

Листинг 1. Реализация подключения к PostgreSQL

Рассмотрим пример работы с HBase в листинге 2. Создан класс CheckedPool, который является наследником класса happybase.ConnectionPool и переопределяет метод connection. В методе connection выполняется получение подключений из пула подключений, их открытие и возврат как результата метода. После использования подключения закрываются и возвращаются в пул. Использование метода get_hbase позволят получать подключения к HBase для дальнейшего использования.

```
from contextlib import contextmanager

import happybase
from config import settings

class CheckedPool(happybase.ConnectionPool):
    @contextmanager
    def connection(self, timeout=None):
        connection = self._acquire_connection(timeout)
        try:
            connection.open()
            yield connection
        finally:
            connection.close()
            self._return_connection(connection)

HbasePool = CheckedPool(
    size=4,
    host=settings.HBASE_HOST,
    port=settings.HBASE_PORT,
```

```
    table_prefix=None)

def get_hbase() -> Generator:
    with HbasePool.connection() as db:
        yield db
```

Листинг 2. Реализация подключения к HBase

В листинге 3 приведен пример работы со Spark. В нем создается spark context, с помощью которого считываются структурированные данные из файла и записываются в PostgreSQL.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.config("spark.driver.memory", "12g")
    .config('spark.driver.extraClassPath', '/root/lib/postgresql-
42.2.22.jar') \
    .config("spark.jars.packages", "com.crealytics:spark-
excel_2.11:0.12.2") \
    .getOrCreate()


def load_xlsfile_to_pgsql(filename, db_pg, schema_pg,
tablename_pg):
    # read excel file
    df = spark.read.format("com.crealytics.spark.excel") \
        .option("useHeader", "true") \
        .option("inferSchema", "true") \
        .option("dataAddress", "'Sheet1'!A1") \
        .load(filename)

    # write to postgresql
    df.write.mode("overwrite").option("truncate", "true") \
        .format('jdbc') \
        .option("url", f"jdbc:postgresql://host_ip:5432/{db_pg}") \
        .option("driver", "org.postgresql.Driver") \
        .option("user", "tech_user") \
        .option("password", "tech_password") \
        .option("dbtable", f"{schema_pg}.{tablename_pg}") \
        .option("numPartitions", 60) \
        .save()
```

Листинг 3. Пример работы со Spark

3.6. ТЕСТИРОВАНИЕ РЕЗУЛЬТАТОВ

Основное назначение хранилища — предоставлять данные для интеллектуального анализа от интерпретации и прогнозирования до выработки рекомендаций.

Для реализации такого конвейера задач была разработана совокупность сервисов, выполняющих различные аналитические задачи [7]. Все сервисы можно подразделить на 2 категории (Рисунок 10):

- внутренние сервисы, которые служат для обеспечения стройной инфраструктуры аналитической системы и вызываются внешними сервисами для решения определенных подзадач;
- внешние (прикладные) сервисы, которые служат для взаимодействия с пользователями и реализуют одну из интересующих пользователя задач.

К внешним сервисам относятся следующие: сервис импорта данных в хранилище; сервис определения структурной сложности текста; сервис оценки удобочитаемости текста; сервис определения степени влияния изучаемых дисциплин на содержание студенческих работ; сервис оценки профессионального словаря, используемого в текстах; сервис оценки качества документирования программного кода.

Для реализации прикладных сервисов потребовалось реализовать не менее важный набор внутренних сервисов, среди которых можно выделить сервис конвертации файлов в разные форматы; сервис извлечения метаданных; сервис извлечения иерархической структуры текста; сервис извлечения сущностей из текстов; сервис формирования словаря профессиональных терминов; сервис определения близости заданных текстов; сервис извлечения программного кода из текста.

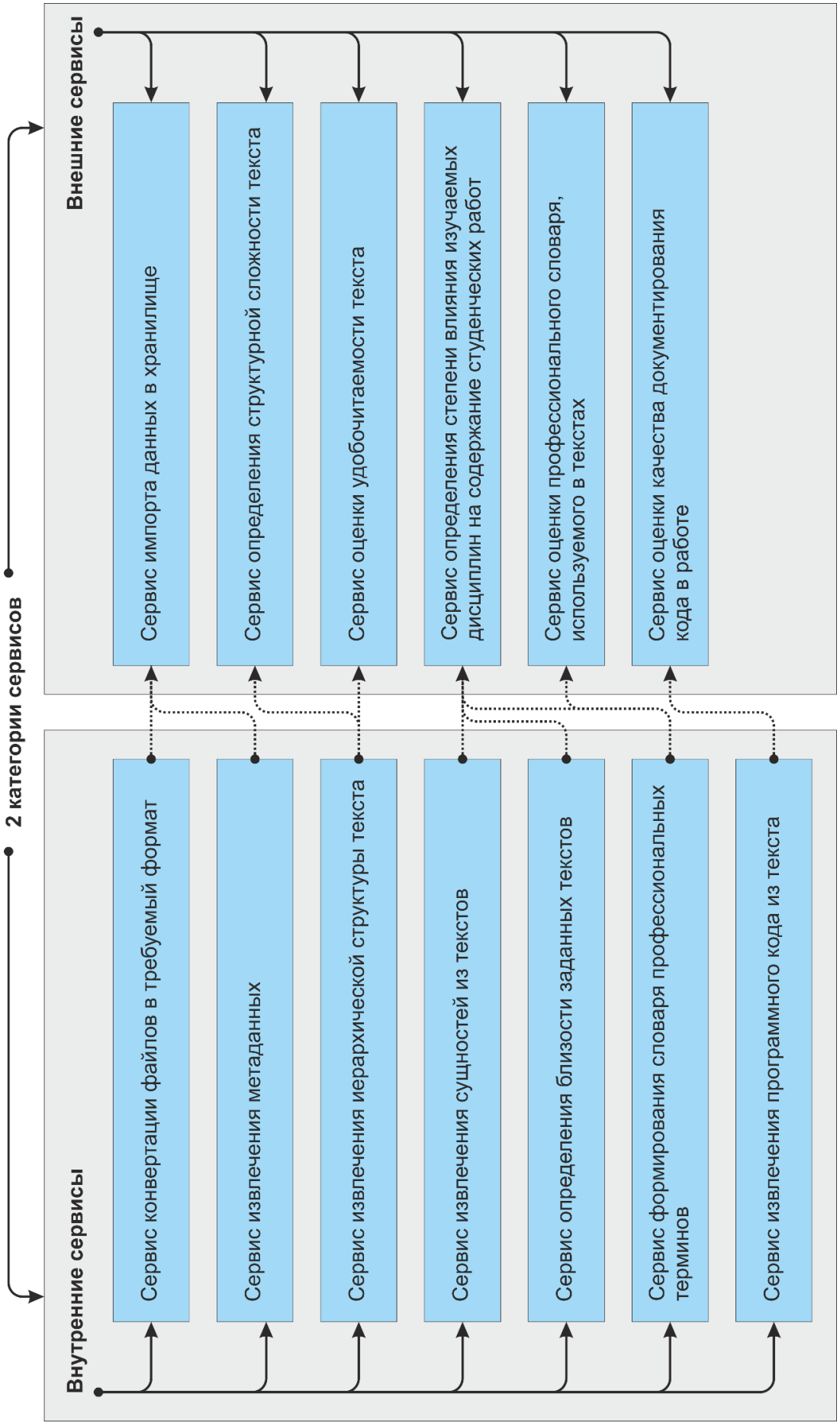

Рис. 10. Категории сервисов системы

Рассмотрим этапы работы внешних и внутренних сервисов на примере одной из функциональных возможностей системы - определению степени влияния изучаемых дисциплин на содержание студенческих работ.

Если проследить всю цепочку от момента загрузки данных в систему до получения конечного результата (Рисунок 11), то можно увидеть, что первым звеном является сервис импорта данных в хранилище. Данный сервис, в свою очередь, вызывает внутренний сервис конвертации содержимого файла во внутренний формат, необходимый для проведения дальнейшего анализа. Таким образом, на данном этапе формируется первый уровень хранилища, содержащий сырые данные.

После сервиса импорта отрабатывает и записывает результаты своей работы в хранилище сервис извлечения иерархической структуры текста, разделяющий целостный текст на взаимосвязанные части: содержание, главы, параграфы, пункты, фрагменты программного кода, приложения, список литературы. На данном этапе в хранилище появляется второй уровень иерархии сохраняемых сервисами данных.

После того, как структура текста известна, запускается прикладной сервис определения степени влияния изучаемых дисциплин на содержание студенческих работ. В основе сервиса лежит принцип определения сходства между двумя объектами (текстами) на основе некоторых критериев [4, 5]. Данный сервис использует в своей работе сервис извлечения сущностей из текстов, сервис определения близости текстов, сервис по формированию словаря профессиональных терминов. Результат работы данного сервиса сохраняется в хранилище на третьем уровне и может быть отображен пользователю по запросу.

Сконструированная таким образом архитектура системы позволяет создать иерархическую структуру сохраняемых данных в хранилище. Предложенный подход позволяет использовать результаты работы одних сервисов другими, а также предоставляет широкую платформу разноуровневых данных при разработке новых сервисов.

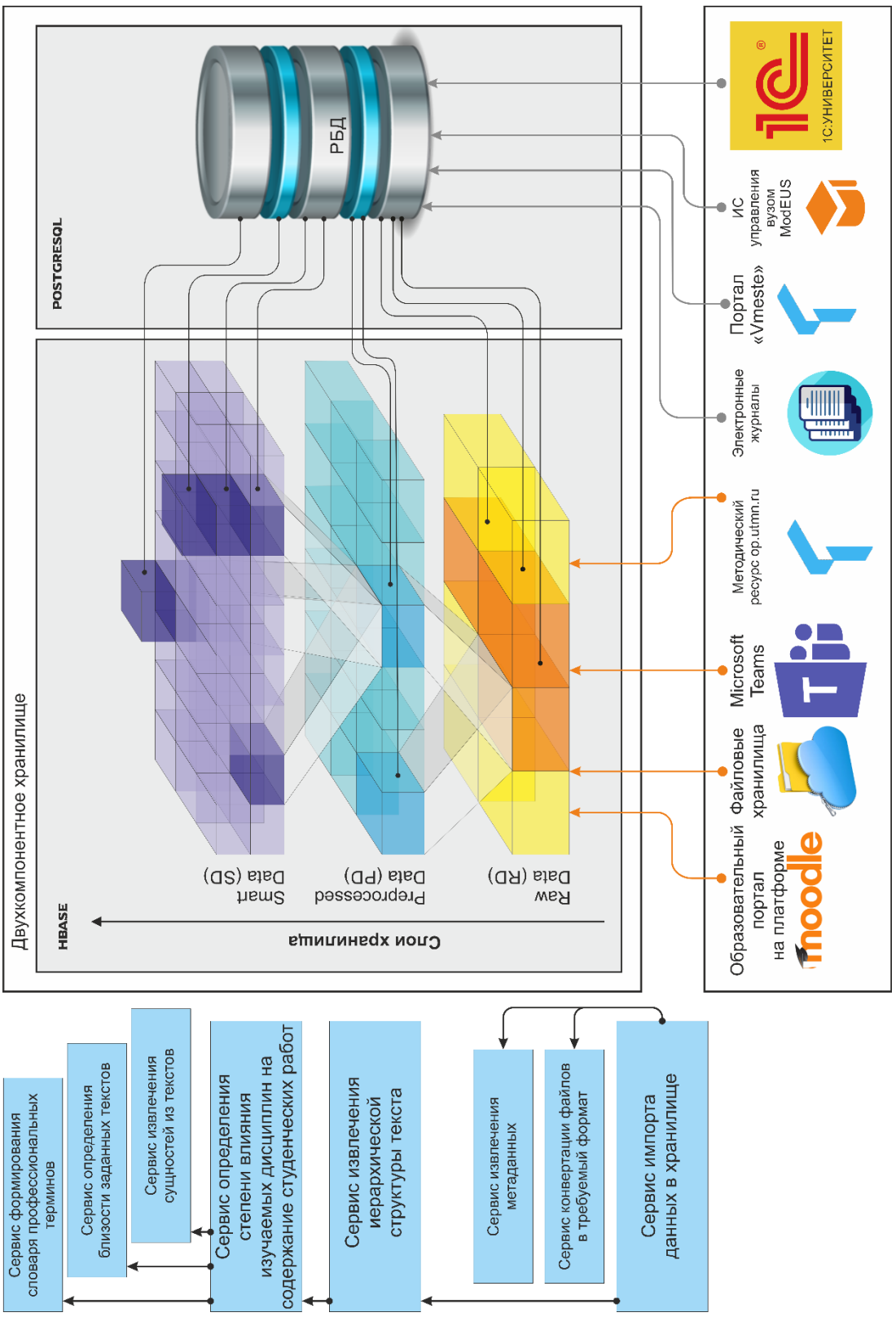

Рис. 11. Этапы работы сервисов

К инфраструктуре системы, развернутой на сервере, организован доступ для разработчиков интеллектуальных сервисов, каждый из которых является частью аналитической системы и предоставляет результаты работы алгоритмов машинного обучения с целью использования их для решения конкретных задач на реальных данных студентов ТюмГУ.

В хранилище загружены данные 865 студентов по 15 направлениям подготовки, 520 текстов ВКР, 20 учебных планов и 675 рабочих программ дисциплин.

Например, рассматривалась задача определения профпригодности студента по данным его цифрового следа. Для решения задачи были обработаны источники данных (тексты вакансий, размещенных работодателями, тексты профессиональных стандартов, рабочие программы дисциплин, тексты студенческих работ), что потребовало обращения к каждому слою информационного хранилища.

При внедрении новой системы в вузе неизменно появляются новые бизнес-процессы, в которые будут вовлечены как студенты, так и профессорско-преподавательский состав, включая руководителей подразделений.

Необходимо учитывать уже имеющиеся бизнес-процессы и избегать дублирования функций и данных, что способствует более легкому процессу интеграции и функционирования системы в целом. Обобщенный бизнес-процесс описываемой системы в нотации BPMN изображен на рисунке 12.

В связи с тем, что в Тюменском Государственном Университете для взаимодействия со студентами широко используется Microsoft Teams, при разработке новых бизнес-процессов было решено включить его как базовое средство сбора артефактов обучения студентов для дальнейшего размещения их в хранилище.

Представленная на рисунке 12 схема описывает базовый бизнес-процесс, позволяющий регулярно пополнять хранилище текстовыми данными. Для Института Математики и Компьютерных наук к таким данным относятся

текстовые работы, написанные студентами, и программные коды, реализованные в рамках поставленных задач на практических занятиях.

Основными участниками процесса являются студенты, преподаватели, менеджер проекта (один или несколько). Преподаватели создают задания в своих командах, следят за тем, чтобы задания вовремя выполнялись и отслеживают оповещения от сервиса загрузки работ. Менеджер проекта следит за тем, чтобы возникающие ошибки загрузки были своевременно устранены, а все необходимые данные загружены в хранилище. Функцией менеджера проекта также является ведение перечня команд MS Teams, из которых необходимо выгружать работы. Данный перечень вносится в настройки системы, и используется сервисом загрузки для сбора данных.

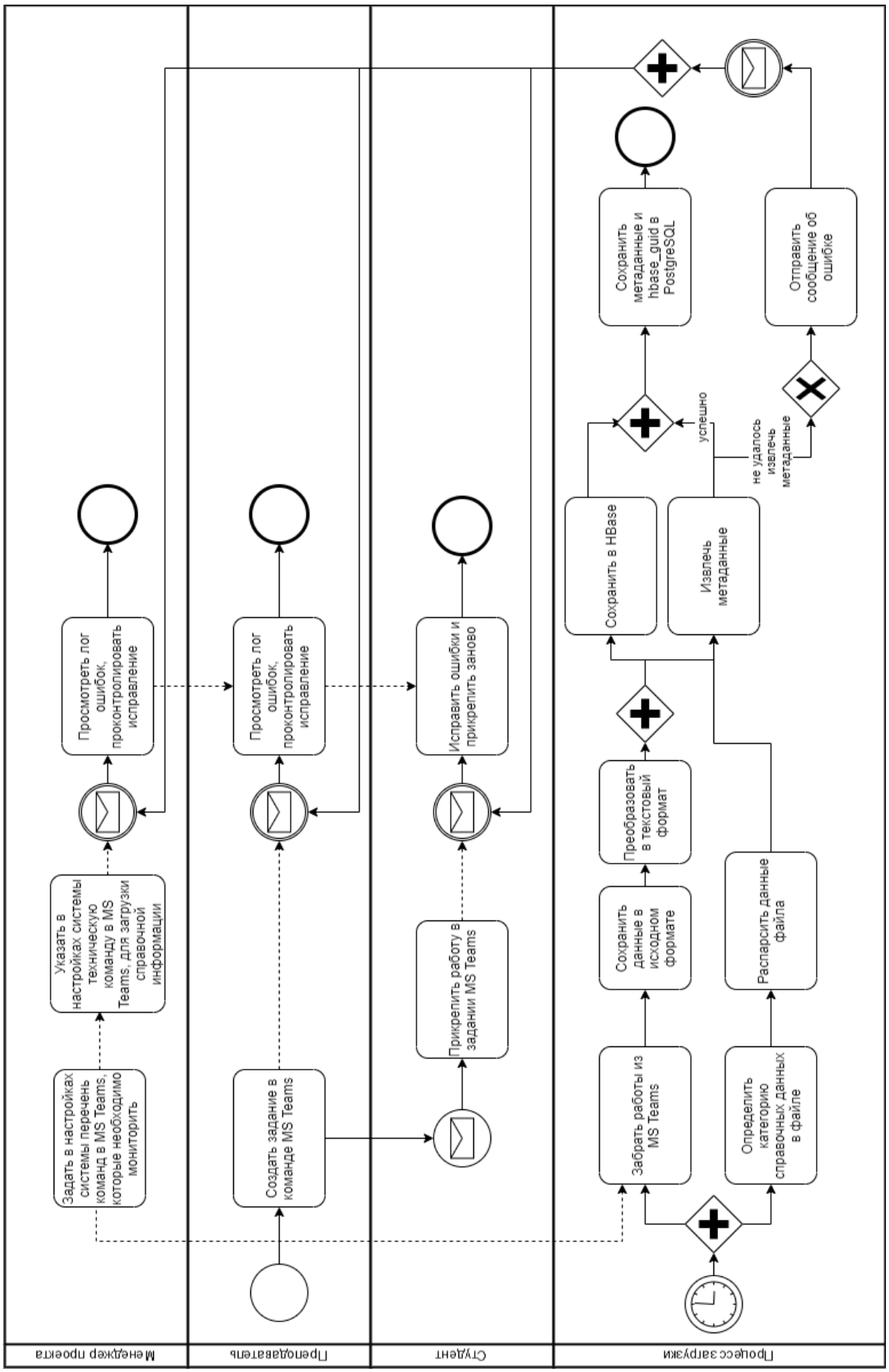

Рис. 12. Бизнес-процесс организации загрузки текстовых данных в хранилище

ЗАКЛЮЧЕНИЕ

В настоящее время система высшего образования развивается очень динамично, вузы для проведения мониторингов и управления образовательным процессом ощущают потребность в использовании разных данных (накопленная информация о студентах, итоговые срезы по направлениям, проведенные опросы, посты в социальных сетях, проводимые мероприятия, открытые курсы и т.д.).

Поэтому архитектура многослойного хранилища данных должна быть гибкой и масштабируемой, как в плане объема данных, так и появлении новых типов, структур, источников. И в этом ориентиром выступают информационные запросы и потребности конечных пользователей системы [12].

В ходе практики обозначенные задачи были поставлены и выполнены: описаны исходные данные и их источники, разработана структура хранилища информации, приведено описание компонентов системы. Система развернута на сервере, организован доступ для разработчиков к инфраструктуре системы.

Самая большая сложность состояла в получении данных из различных систем, уже функционирующих в образовательном пространстве. В настоящее время одни учётные системы установлены на серверах университета, находятся во внутренней сети и можно получить данные, зная структуру и доступ. Есть системы с открытым кодом (например, на платформе Moodle), где есть возможность доработки при необходимости. Но есть информационные ресурсы в удаленном доступе, где отсутствует единый регламент хранения данных, администрирование осуществляется централизовано, что представляет затруднения и увеличивает временные характеристики.

Ситуацию осложняет то, что одни и те же виды данных могут храниться в различных форматах, и необходимо обеспечить поддержку импорта данных из всего представленного многообразия имеющихся источников.

Пользователями данного хранилища являются разработчики интеллектуальных сервисов, каждый из которых является частью аналитической

системы вуза и предоставляет результаты работы алгоритмов машинного обучения с целью использования их для индивидуализации образовательных траекторий студентов высших учебных заведений.

Возможным направлением развития системы может стать получение квазипотоковых данных, показывающих динамику образовательного процесса в целом: от текущей наполняемости аудитории до эмоциональной окраски, отражающей восприятие материала лекций или обстановки на экзамене и зачете.

Результаты работы были апробированы на IV Международной научной конференции «Информатизация образования и методика электронного обучения: цифровые технологии в образовании» (Красноярск, 6-9 октября 2020), Всероссийской конференции молодых ученых «Математическое и информационное моделирование – 2020», Всероссийской конференции молодых ученых «Математическое и информационное моделирование – 2021». Полученные достижения были включены в Программу повышения квалификации «Педагог в современной цифровой (информационной) образовательной среде вуза» в Тюменском государственном университете в блок лекций по теме «Информационные сервисы для сопровождения индивидуальных образовательных траекторий студентов вуза».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Захарова И. Г. Big Data и машинное обучение в проектировании и реализации индивидуальных образовательных траекторий // Информатизация образования и методика электронного обучения: материалы II Междунар. науч. конф. Красноярск: Сибирский федеральный университет, 2018. С. 20–24.
2. Julius Murumba1, Elyjoy Micheni. Big Data Analytics in Higher Education: A Review // The International Journal of Engineering and Science (IJES). – 2017. – Volume 6, Issue 6, PP 14-21.
3. Горлушкина Н. Н., Коцюба И. Ю., Хлопотов М. В. Задачи и методы интеллектуального анализа образовательных данных для поддержки принятия решений // Образовательные технологии и общество. 2015. Т. 18. No 1. С. 472–482.
4. Ahuja R., Jha A., Maurya R., Srivastava R. (2019) Analysis of Educational Data Mining. In: Yadav N., Yadav A., Bansal J., Deep K., Kim J. (eds) Harmony Search and Nature Inspired Optimization Algorithms. Advances in Intelligent Systems and Computing, vol 741. Springer, Singapore.
5. Silva C., Fonseca J. (2017) Educational Data Mining: A Literature Review. In: Rocha Á., Serrhini M., Felgueiras C. (eds) Europe and MENA Cooperation Advances in Information and Communication Technologies. Advances in Intelligent Systems and Computing, vol 520. Springer, Cham.
6. Мамедова Г.А., Зейналова Л.А., Меликова Р.Т. Технологии больших данных в электронном образовании. // Открытое образование. -2017; (6): 41-48. https://doi.org/10.21686/1818-4243-2017-6-41-48
7. Иваненко О. А., Бакланов И. А., Чемакин Т. А., Воробьев А. М. Информационные сервисы для сопровождения индивидуальных образовательных траекторий студентов вуза // Информатизация образования и методика электронного обучения: цифровые технологии в образовании: материалы IV Междунар. науч. конф. Красноярск, 6–9

октября 2020 г.: в 2 ч. Ч. 2 / под общ. ред. М. В. Носкова. – Красноярск: Сиб. федер. ун-т, 2020. С. 115–118.

8. Воробьева М. С., Лобунцов Д. С. Подходы к прогнозированию образовательных результатов по данным цифрового следа студента // Информатизация образования и методика электронного обучения: цифровые технологии в образовании: материалы IV Междунар. науч. конф. Красноярск, 6–9 октября 2020 г.: в 2 ч. Ч. 2 / под общ. ред. М. В. Носкова. – Красноярск: Сиб. федер. ун-т, 2020. С. 73–77.
9. Шенгелия Д. Ю., Бакланов И. А., Воробьёв А. М., Иваненко О. А. Сравнительный анализ решений для задач сбора и хранения данных о результатах обучения студентов // Математическое и информационное моделирование: материалы Всерос. конф. молодых ученых, г. Тюмень, 1–8 июня 2020 г. / Министерство науки и высшего образования Российской Федерации, Тюменский государственный университет, Институт математики и компьютерных наук. Тюмень: Изд-во Тюменского государственного университета, 2020. Вып. 18. С. 208–217.
10. Захарова И. Г., Карпов М. Г., Лобунцов Д. С. Информационно-аналитическая поддержка управления образовательным процессом с использованием данных цифрового следа студента// Информатизация образования и методика электронного обучения: цифровые технологии в образовании: материалы IV Междунар. науч. конф. Красноярск, 6–9 октября 2020 г.: в 2 ч. Ч. 2 / под общ. ред. М. В. Носкова. – Красноярск: Сиб. федер. ун-т, 2020. С. 104–108.
11. Ahmed Yaseen Mjhool1, Ahmed Hazim Alhilali2, Salam Al-augby. A proposed architecture of big educational data using hadoop at the University of Kufa // International Journal of Electrical and Computer Engineering (IJECE) Vol. 9, No. 6, December 2019, pp. 4970~4978.
12. Таратухина Ю. В., Маркарян М. С. Общие принципы проектирования рекомендательного веб-сервиса по моделированию индивидуальной

образовательной траектории обучающихся // Открытое и дистанционное образование. 2016. No 2 (62). С. 77–82.

13. G. G. W. Mhon and N. S. M. Kham, "ETL Preprocessing with Multiple Data Sources for Academic Data Analysis," 2020 IEEE Conference on Computer Applications (ICCA), 2020, pp. 1-5.
14. Ahmed Yaseen Mjhool1, Ahmed Hazim Alhilali2, Salam Al-augby. A proposed architecture of big educational data using hadoop at the University of Kufa // International Journal of Electrical and Computer Engineering (IJECE) Vol. 9, No. 6, December 2019, pp. 4970~4978.
15. F. Gürcan and M. Berigel, "Real-Time Processing of Big Data Streams: Lifecycle, Tools, Tasks, and Challenges", 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2018, pp. 1-6.
16. Naik, N. “Docker container-based big data processing system in multiple clouds for everyone.” 2017 IEEE International Systems Engineering Symposium (ISSE) (2017): pp. 1-7.
17. Shantaram, Patil & Wagh, Prof & Sabri, Mustafa. (2020). SETTING UP HADOOP CLUSTER NETWORK USING DOCKER. SSRN Electronic Journal. 7, pp. 86-94.
18. Gerardus Blokdyk “Apache NiFi A Complete Guide - 2020 Edition”. 5STARCooks, 2020, 289 p.
19. Use the Microsoft Graph API [сайт] URL: https://docs.microsoft.com/en-us/graph/use-the-api (дата обращения: 01.06.2021).
20. PostgreSQL: Documentation [сайт] URL: https://www.postgresql.org/docs/ (дата обращения: 04.06.2021).

Приложение 1

Файл docker-compose для инфраструктуры системы

```
services:
  ds_postgres:
    container_name: ds-postgres
    image: postgres:12
    volumes:
      - ${POSTGRES_DATA}:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB_NAME}
    networks:
      ds_network:
        ipv4_address: 172.20.10.2

  ds_hadoop:
    container_name: ds-hadoop
    build:
      context: hadoop
    hostname: ds-hadoop
    volumes:
      - ${HADOOP_DATA}:/dfs
      - ${HBASE_HADOOP_DATA}:/hbase
    networks:
      ds_network:
        ipv4_address: 172.20.10.3

  ds_python:
    container_name: ds-python
    build:
      context: python
    depends_on:
      - ds_postgres
      - ds_hadoop
    networks:
      ds_network:
        ipv4_address: 172.20.10.4

  ds_csharp:
    container_name: ds-csharp
    build:
      context: csharp
    environment:
      - ASPNETCORE_URLS=http://+:5000
    networks:
      ds_network:
        ipv4_address: 172.20.10.5
```

```
ds_nginx:
  container_name: ds-nginx
  build:
    context: vue
  ports:
    - 80:80
    - 443:443
  volumes:
    - ${LETSENCRYPT_DATA}:/etc/letsencrypt
  environment:
    - DOMAIN=${DOMAIN}
    - EMAIL=${EMAIL}
  networks:
    ds_network:
      ipv4_address: 172.20.10.6

ds_auth:
  image: quay.io/keycloak/keycloak:10.0.2
  container_name: ds-auth
  command: [ "-Djboss.socket.binding.port-offset=0" ]
  environment:
    - KEYCLOAK_USER=${KEYCLOAK_USER}
    - KEYCLOAK_PASSWORD=${KEYCLOAK_PASSWORD}
    - DB_VENDOR=postgres
    - DB_ADDR=ds-postgres
    - DB_USER=${POSTGRES_USER}
    - DB_PASSWORD=${POSTGRES_PASSWORD}
    - DB_DATABASE=keycloak
  networks:
    ds_network:
      ipv4_address: 172.20.10.7

ds_hue:
  container_name: ds-hue
  image: gethue/hue:latest
  volumes:
    - ./hue.ini:/usr/share/hue/desktop/conf/z-hue.ini
  depends_on:
    - ds_postgres
  networks:
    ds_network:
      ipv4_address: 172.20.10.8

ds_nifi:
  container_name: ds-nifi
  image: apache/nifi:1.13.2
  volumes:
    - ${NIFI_DATA_CONF}:/opt/nifi/conf
```

```
      - ${NIFI_DATA_DATABASE}:/opt/nifi/database_repository
      - ${NIFI_DATA_FLOWFILE}:/opt/nifi/flowfile_repository
      - ${NIFI_DATA_CONTENT}:/opt/nifi/content_repository
      - ${NIFI_DATA_PROVENANCE}:/opt/nifi/provenance_repository
      - ${NIFI_DATA_STATE}:/opt/nifi/state
    networks:
      ds_network:
        ipv4_address: 172.20.10.9

networks:
  ds_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.10.0/24
```

Приложение 2

Docker-файл для тестового однонодового кластера Hadoop

```
FROM centos:7

ARG JAVA_VERSION=8

RUN set -eux && \
    yum install -y "java-1.$JAVA_VERSION.0-openjdk-devel" openssh-
server openssh-clients which && \
    yum clean all && \
    rm -rf /var/cache/yum

ARG HADOOP_VERSION=3.2.2

ARG HBASE_VERSION=2.4.1

ARG HIVE_VERSION=3.1.2

ARG SPARK_VERSION=3.1.1

ENV JAVA_HOME=/usr

ENV
PATH=$PATH:/opt/hadoop/bin:/opt/hadoop/sbin:/opt/hbase/bin:/opt/hi
ve/bin:/opt/spark/bin:/opt/spark/sbin

WORKDIR /

RUN set -eux && \
    yum install -y wget hostname && \
    wget -t 10 --max-redirect 1 --retry-connrefused -O
hadoop.tar.gz
"http://www.apache.org/dyn/closer.lua?filename=hadoop/common/hadoo
p-$HADOOP_VERSION/hadoop-$HADOOP_VERSION.tar.gz&action=download"
|| \
    wget -t 10 --max-redirect 1 --retry-connrefused -O
hadoop.tar.gz
"http://archive.apache.org/dist/hadoop/common/hadoop-
$HADOOP_VERSION/hadoop-$HADOOP_VERSION.tar.gz" && \
    mkdir -p /opt/hadoop && \
    tar zxvf hadoop.tar.gz \
          --directory=/opt/hadoop \
          --exclude=hadoop-$HADOOP_VERSION/share/doc \
          --strip 1 && \
    rm -fv hadoop.tar.gz && \
    { rm -rf /opt/hadoop/share/doc; : ; } && \
    wget -t 10 --max-redirect 1 --retry-connrefused -O
hbase.tar.gz
```

```
"http://www.apache.org/dyn/closer.lua?filename=hbase/$HBASE_VERSIO
N/hbase-$HBASE_VERSION-bin.tar.gz&action=download" || \
    wget -t 10 --max-redirect 1 --retry-connrefused -O
hbase.tar.gz
"http://archive.apache.org/dist/hbase/$HBASE_VERSION/hbase-
$HBASE_VERSION-bin.tar.gz" && \
    mkdir -p /opt/hbase && \
    tar zxvf hbase.tar.gz \
          --directory=/opt/hbase \
          --strip 1 && \
    rm -fv hbase.tar.gz && \
    wget -t 10 --max-redirect 1 --retry-connrefused -O hbase-
operator-tools.tar.gz
"http://www.apache.org/dyn/closer.lua?filename=hbase/hbase-
operator-tools-1.1.0/hbase-operator-tools-1.1.0-
bin.tar.gz&action=download" || \
    wget -t 10 --max-redirect 1 --retry-connrefused -O hbase-
operator-tools.tar.gz "http://archive.apache.org/dist/hbase/hbase-
operator-tools-1.1.0/hbase-operator-tools-1.1.0-bin.tar.gz" && \
    mkdir -p /opt/hbase-operator-tools && \
    tar zxvf hbase-operator-tools.tar.gz \
          --directory=/opt/hbase-operator-tools \
          --strip 1 && \
    rm -fv hbase-operator-tools.tar.gz && \
    wget -t 10 --max-redirect 1 --retry-connrefused -O hive.tar.gz
"http://www.apache.org/dyn/closer.lua?filename=hive/hive-
$HIVE_VERSION/apache-hive-$HIVE_VERSION-
bin.tar.gz&action=download" || \
    wget -t 10 --max-redirect 1 --retry-connrefused -O hive.tar.gz
"http://archive.apache.org/dist/hive/hive-$HIVE_VERSION/apache-
hive-$HIVE_VERSION-bin.tar.gz" && \
    mkdir -p /opt/hive && \
    tar zxvf hive.tar.gz \
          --directory=/opt/hive \
          --strip 1 && \
    rm -fv hive.tar.gz && \
    wget -t 10 --max-redirect 1 --retry-connrefused -O spark.tgz
"http://www.apache.org/dyn/closer.lua?filename=spark/spark-
$SPARK_VERSION/spark-$SPARK_VERSION-bin-
hadoop3.2.tgz&action=download" || \
    wget -t 10 --max-redirect 1 --retry-connrefused -O spark.tgz
"http://archive.apache.org/dist/spark/spark-$SPARK_VERSION/spark-
$SPARK_VERSION-bin-hadoop3.2.tgz" && \
    mkdir -p /opt/spark && \
    tar zxvf spark.tgz \
          --directory=/opt/spark \
          --strip 1 && \
    rm -fv spark.tgz && \
    yum autoremove -y && \
```

```
    # gets autoremoved, ensure it's added back as Hadoop scripts
need it
    yum install -y hostname && \
    yum clean all && \
    rm -rf /var/cache/yum

COPY entrypoint.sh /
COPY conf/core-site.xml /opt/hadoop/etc/hadoop/
COPY conf/hdfs-site.xml /opt/hadoop/etc/hadoop/
COPY conf/yarn-site.xml /opt/hadoop/etc/hadoop/
COPY conf/mapred-site.xml /opt/hadoop/etc/hadoop/
COPY conf/hbase-site.xml /opt/hbase/conf/hbase-site.xml
COPY conf/hive-conf.xml /opt/hive/conf/hive-site.xml
COPY conf/classpath.txt /opt/spark/conf/classpath.txt
COPY conf/spark-defaults.conf /opt/spark/conf/spark-defaults.conf
COPY conf/slaves /opt/spark/conf/slaves
COPY hadoop.sh /etc/profile.d/
COPY ssh/config /root/.ssh/

RUN set -eux && \
    /opt/hadoop/bin/hdfs namenode -format && \
    groupadd hadoop && \
    useradd -g hadoop hdfs && \
    useradd -g hadoop yarn && \
    mkdir -p /dfs/name && \
    mkdir -p /var/run/hdfs-sockets && \
    chown -R hdfs:hadoop /var/run/hdfs-sockets && \
    mkdir -p /opt/hadoop/logs && \
    chown -R hdfs:hadoop /dfs/name && \
    chgrp -R hadoop /opt/hadoop/logs && \
    chmod -R 0770 /opt/hadoop/logs && \
    mkdir -p /root/.ssh \
          /home/hdfs/.ssh \
          /home/yarn/.ssh && \
    chown hdfs /home/hdfs/.ssh && \
    chown yarn /home/yarn/.ssh && \
    chmod 0700 /root/.ssh \
               /home/hdfs/.ssh \
               /home/yarn/.ssh && \
    rm /opt/hive/lib/guava-19.0.jar && \
    cp /opt/hadoop/share/hadoop/hdfs/lib/guava-27.0-jre.jar
/opt/hive/lib/ && \
    ln -s /opt/hive/conf/hive-site.xml /opt/spark/conf/hive-
site.xml && \
    ln -s /opt/hive/lib/postgresql-9.4.1208.jre7.jar
/opt/spark/jars/postgresql-9.4.1208.jre7.jar && \
    cp /opt/spark/yarn/spark-3.1.1-yarn-shuffle.jar
/opt/hadoop/share/hadoop/yarn && \
```

```
    cp /opt/spark/jars/scala-library-2.12.10.jar /opt/hive/lib &&
\
    cp /opt/spark/jars/spark-core_2.12-3.1.1.jar /opt/hive/lib &&
\
    cp /opt/spark/jars/spark-network-common_2.12-3.1.1.jar
/opt/hive/lib && \
    cp /opt/spark/jars/spark-unsafe_2.12-3.1.1.jar /opt/hive/lib

ENV HDFS_NAMENODE_USER=hdfs
ENV HDFS_SECONDARYNAMENODE_USER=hdfs
ENV HDFS_DATANODE_USER=hdfs
ENV YARN_RESOURCEMANAGER_USER=yarn
ENV YARN_NODEMANAGER_USER=yarn

EXPOSE 8020 8042 8088 9000 9868 9870 10020 19888 50010 50020 50090
10002 8080 8085 9090 9095 2181 16010

CMD ["/entrypoint.sh"]
```

Приложение 3

Код скрипта генерации конфигурационных файлов

```
#!/bin/bash
#$1 - user

if [ "$#" -lt 1 ]; then
    echo "Missing argument."
    exit
elif [ "$#" -gt 1 ]; then
    echo "Too many arguments."
    exit
fi


cd /usr/share/easy-rsa/3.0.7
key=./pki/private/"$1".key
if test -f "$key"; then
    (echo "yes"; echo -en '\n') | ./easyrsa gen-req $1 nopass
else
    (echo -en '\n') | ./easyrsa gen-req $1 nopass
fi

error=$?
if [ "$error" != 0 ]; then
    echo "Can't create request. OpenSSL returned: $error"
    exit
fi

(echo "yes") | ./easyrsa sign-req client $1 batch RFFIgrantVPN
error=$?
if [ "$error" != 0 ]; then
    echo "Can't sign request. OpenSSL returned: $error"
    exit
fi

cert=./pki/issued/"$1".crt
if test -f "$cert" && test -f "$key"; then
    echo "Certificate and key with CNAME: $1 created successful."
else
    echo "Something bad happened."
    exit
fi

ca=./pki/ca.crt
ta=./pki/ta.key

conf=./ds-dev.ovpn
if test -f "$conf"; then
```

```
    rm $conf
fi
touch $conf

#config
echo "client" >> $conf
echo "remote 5.1.53.166 995" >> $conf
echo "proto tcp" >> $conf
echo "dev tap" >> $conf
echo "nobind" >> $conf
echo "" >> $conf
echo "ca ca.crt" >> $conf
echo "cert $1.crt" >> $conf
echo "key $1.key" >> $conf
echo "" >> $conf
echo "tls-client" >> $conf
echo "tls-auth ta.key 1" >> $conf
echo "" >> $conf
echo "compress lzo" >> $conf
echo "persist-key" >> $conf
echo "persist-tun" >> $conf
echo "verb 3" >> $conf

zip_conf=./pki/config/$1.zip

if test -f "$zip_conf"; then
    rm $zip_conf;
fi

zip -j --password 123 $zip_conf $conf $ca $cert $key $ta
rm $conf

exit
```