Версия от 9 мая 2025 г.

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Название института

Работа допущена к защите
Руководитель образовательной программы
«Прикладная математика и информатика»
_____ К.Н. Козлов
«_____» _____ 2025 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
РАБОТА БАКАЛАВРА

АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ КОНФИГУРАЦИЙ ЭЛЕМЕНТОВ ИНФРАСТРУКТУРЫ ПРОГРАММНЫХ СИСТЕМ ДЛЯ РАБОТЫ С БОЛЬШИМИ ДАННЫМИ

по направлению подготовки 01.03.02 Прикладная математика и информатика
Направленность (профиль) 01.03.02_02 Системное программирование

Выполнил
студент гр. 5030102/10201 И.И. Хамидуллин

Руководитель
д.т.н.,
профессор ВШПМиВФ Ф.А. Новиков

Консультант ВКР Д.Ю. Иванов

Консультант
по нормоконтролю Л.А. Арефьева

Санкт-Петербург — 2025

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Название института

УТВЕРЖДАЮ
Руководитель образовательной программы
_____ К.Н. Козлов
«_____» _____ 2025г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

студенту Хамидуллину Ильсафу Ильназовичу гр. 5030102/10201

1. Тема работы: Автоматическая генерация конфигураций элементов инфраструктуры программных систем для работы с большими данными.
2. Срок сдачи студентом законченной работы[1]: дд.мм.202X.
3. Исходные данные по работе[2]: статистические данные с сайта [3.1], а также из репозитория [3.4]; основным источником литературы является монография [3.3] и статья [3.2].
   3.1. Сайт Федеральной службы государственной статистики. — URL: http://www.gks.ru/ (дата обращения: 06.03.2019).
   3.2. *Adams P.* The title of the work // The name of the journal. — 1993. — Vol. 4, no. 2. — P. 201–213.
   3.3. *Babington P.* The title of the work. Vol. 4. — 3rd ed. — The address: The name of the publisher, 1993. — 255 p. — (Ser.: 10).
   3.4. The UC Irvine Machine Learning Repository. — URL: http://archive.ics.uci.edu/ml (visited on 06.03.2019).
4. Содержание работы (перечень подлежащих разработке вопросов):
   4.1. Обзор литературы по теме ВКР.
   4.2. Исследование программных продуктов.

[1] Последний день преддипломной практики согласно учебному плану.

[2] Текст, который подчёркнут и/или выделен в отдельные элементы нумерационного списка, приведён в качестве примера.

4.3. Разработка метода/алгоритма/программы.

4.4. Апробация разработанного метода/алгоритма/программы.

5. Перечень графического материала (с указанием обязательных чертежей):

5.1. Схема работы метода/алгоритма.

5.2. Архитектура разработанной программы/библиотеки.

6. Консультанты по работе[3]:

6.1. Должность, степень, Д.Ю. Иванов.

6.2. Должность, степень, Л.А. Арефьева (нормоконтроль).

7. Дата выдачи задания[4]: дд.мм.202X.

Руководитель ВКР _____ Ф.А. Новиков

Консультант[5] _____ Д.Ю. Иванов

Задание принял к исполнению дд.мм.202X

Студент _____ И.И. Хамидуллин

[3]Подпись консультанта по нормоконтролю пока не требуется. Назначается всем по умолчанию.

[4]Как правило не позднее 3 месяцев до ГИА (утверждение тем ВКР по университету) или строго не позже 30 дней до ГИА.

[5]В случае, если есть консультант, отличный от консультанта по нормоконтролю.

РЕФЕРАТ

На 31 с., 1 рисунок, 1 таблицу, 2 приложения

КЛЮЧЕВЫЕ СЛОВА: СТИЛЕВОЕ ОФОРМЛЕНИЕ САЙТА, УПРАВЛЕНИЕ КОНТЕНТОМ, PHP, MYSQL, АРХИТЕКТУРА СИСТЕМЫ.[6]

Тема выпускной квалификационной работы: «Автоматическая генерация конфигураций элементов инфраструктуры программных систем для работы с большими данными»[7].

В данной работе изложена сущность подхода к созданию динамического информационного портала на основе использования открытых технологий Apache, MySQL и PHP. Даны общие понятия и классификация IT-систем такого класса. Проведен анализ систем-прототипов. Изучена технология создания указанного класса информационных систем. Разработана конкретная программная реализация динамического информационного портала на примере портала выбранной тематики...[8]

В данной работе изложена сущность подхода к созданию динамического информационного портала на основе использования открытых технологий Apache, MySQL и PHP. Даны общие понятия и классификация IT-систем такого класса. Проведен анализ систем-прототипов. Изучена технология создания указанного класса информационных систем. Разработана конкретная программная реализация динамического информационного портала на примере портала выбранной тематики...

ABSTRACT

31 pages, 1 figures, 1 tables, 2 appendices

[6]Всего **слов**: от 3 до 15. Всего **слов и словосочетаний**: от 3 до 5. Оформляются в именительном падеже множественного числа (или в единственном числе, если нет другой формы), оформленных по правилам русского языка. *Внимание! Размещение сноски после точки является примером как запрещено оформлять сноски.*

[7]Реферат **должен содержать**: предмет, тему, цель ВКР; метод или методологию проведения ВКР; результаты ВКР; область применения результатов ВКР; выводы.

[8]ОТ 1000 ДО 1500 печатных знаков (ГОСТ Р 7.0.99-2018 СИБИД) на русский или английский текст. Текст реферата повторён дважды на русском и английском языке для демонстрации подхода к нумерации страниц.

KEYWORDS: STYLE REGISTRATION, CONTENT MANAGEMENT, PHP, MYSQL, SYSTEM ARCHITECTURE.

The subject of the graduate qualification work is «Automatic generation of configurations for infrastructure elements of software systems for big data processing».

In the given work the essence of the approach to creation of a dynamic information portal on the basis of use of open technologies Apache, MySQL and PHP is stated. The general concepts and classification of IT-systems of such class are given. The analysis of systems-prototypes is lead. The technology of creation of the specified class of information systems is investigated. Concrete program realization of a dynamic information portal on an example of a portal of the chosen subjects is developed...

In the given work the essence of the approach to creation of a dynamic information portal on the basis of use of open technologies Apache, MySQL and PHP is stated. The general concepts and classification of IT-systems of such class are given. The analysis of systems-prototypes is lead. The technology of creation of the specified class of information systems is investigated. Concrete program realization of a dynamic information portal on an example of a portal of the chosen subjects is developed...

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный этап развития цифровых технологий характеризуется стремительным увеличением объемов данных, генерируемых в различных сферах человеческой деятельности. Этот феномен, известный как "информационный взрыв требует принципиально новых подходов к обработке, хранению и анализу информации. В условиях, когда традиционные методы работы с данными становятся неэффективными, особую актуальность приобретают технологии больших данных (Big Data), предлагающие инновационные решения для извлечения ценных знаний из огромных массивов неструктурированной информации[4].

Актуальность темы данного исследования обусловлена несколькими ключевыми факторами. Во-первых, в эпоху цифровой экономики данные становятся стратегическим ресурсом, сравнимый по значимости с традиционными материальными активами[5]. Во-вторых, сложность современных информационных систем достигла такого уровня, когда ручная настройка их компонентов становится не только трудоемкой, но и потенциально подверженной человеческим ошибкам. В-третьих, переход к agile-методологиям и DevOps-практикам требует новых подходов к управлению инфраструктурой, обеспечивающих скорость, надежность и воспроизводимость развертывания сложных систем.

В контексте этих вызовов особое значение приобретает автоматизация процессов настройки и конфигурирования инфраструктуры для работы с большими данными. Концепция "Инфраструктура как код"(Infrastructure As Code)[6] предлагает подходы для управления и предоставления вычислительной инфраструктуры с помощью декларативных или скриптовых определений.

Традиционные подходы, основанные на ручном редактировании конфигурационных файлов и скриптов, не только требуют значительных временных затрат, но и создают риски возникновения "дрейфа конфигураций"(configuration drift)[7], когда фактическое состояние системы постепенно расходится с документально зафиксированным.

ПОСТАНОВКА ЗАДАЧИ

В современных условиях стремительного роста объёмов данных и усложнения архитектуры информационных систем ручное конфигурирование инфраструктуры для работы с большими данными становится неэффективным и подверженным ошибкам. Существующие инструменты автоматизации, такие как Terraform и Ansible, предоставляют общие механизмы развёртывания, но не предлагают специализированных решений для технологий Big Data, требующих согласованной настройки множества взаимосвязанных компонентов: систем хранения, потоковой обработки, ETL-конвейеров и инструментов визуализации.

Целью данной работы является разработка программного инструмента `Data Platform Deployer` (далее `dpd`), автоматизирующего процесс создания конфигурационных файлов для развёртывания платформы обработки данных на основе декларативного описания её компонентов пользователем. Инструмент принимает на вход описание целевой инфраструктуры в формате YAML и генерирует готовые к использованию артефакты:

- конфигурационные файлы `docker-compose.yml` для быстрого развёртывания всех необходимых сервисов в контейнерной среде;
- скрипты инициализации и базовые конфигурационные файлы для СУБД (PostgreSQL, ClickHouse), адаптированные под типовые задачи обработки данных;
- конфигурации для S3-совместимого хранилища (например, Minio);
- конфигурации для брокера сообщений Apache Kafka, включая создание топиков и настройки Kafka Connect с необходимыми коннекторами (Debezium для CDC, S3 Sink Connector);
- конфигурации для AKHQ — инструмента мониторинга Kafka и Kafka Connect;
- базовые настройки для подключения BI-систем (например, Apache Superset) к развернутым источникам данных.

При разработке инструмента `dpd` должны соблюдаться следующие критерии:

- Воспроизводимость: идентичные конфигурации при одинаковом входном описании.
- Масштабируемость: поддержка добавления новых типов компонентов через модули.

– Согласованность: автоматическая проверка зависимостей между сервисами.

Для достижения поставленной цели были определены следующие задачи:

A. Анализ предметной области и существующих подходов к развёртыванию платформ Big Data:
   – изучение типовых архитектурных паттернов платформ для обработки больших данных
   – исследование возможностей и ограничений существующих инструментов управления конфигурациями и IaC в контексте Big Data;
   – определение ключевых компонентов и их типовых конфигураций для включения в состав `dpd`.

B. Проектирование метамодели декларативного описания инфраструктуры:
   – разработка структуры YAML-файла для описания компонентов платформы, их параметров и взаимосвязей;
   – проектирование системы валидации входных конфигураций (например, на основе JSON Schema) для обеспечения корректности пользовательского ввода.

C. Разработка ядра генератора конфигураций (`dpd`):
   – реализация логики парсинга входного YAML-описания;
   – создание механизма шаблонизации для генерации конфигурационных файлов (`docker-compose.yml`, настройки сервисов и др.).

D. Реализация модулей генерации для ключевых компонентов платформы:
   – модуль для Apache Kafka и Kafka Connect (включая коннекторы Debezium PostgreSQL, S3 Sink);
   – модуль для СУБД PostgreSQL (источник данных);
   – модуль для аналитической СУБД ClickHouse (хранилище данных);
   – модуль для S3-совместимого хранилища Minio (архивное хранилище/Data Lake);
   – модули для вспомогательных инструментов: AKHQ (мониторинг Kafka), Apache Superset (BI).

E. Тестирование и валидация инструмента `dpd`:

- развёртывание тестовых стендов с различной конфигурацией при помощи сгенерированных артефактов;
- функциональное тестирование развернутых платформ для проверки корректности работы и взаимодействия компонентов;
- сравнительный анализ времени и сложности развёртывания платформы с использованием `dpd` и традиционных ручных методов.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Развертывание и управление инфраструктурой для систем обработки больших данных (Big Data) представляет собой сложную задачу, требующую координации множества разнородных компонентов, настройки их взаимодействия и обеспечения масштабируемости, надёжности и безопасности.

Современные подходы к управлению инфраструктурой стремятся автоматизировать эти процессы, однако специфика Big Data-систем накладывает дополнительные требования. В данном разделе рассматриваются существующие инструменты управления инфраструктурой и анализируются ключевые требования к интеграции компонентов, характерные для платформ обработки больших данных, с акцентом на решения, актуальные для российского рынка.

A. **Управляемые облачные сервисы (Managed Cloud Services)** на примере российских провайдеров Российские облачные провайдеры, такие как Yandex Cloud[номер]???вставитьлит и VK Cloud[номер]???вставитьлит (ранее Mail.ru Cloud Solutions), активно развивают свои портфели управляемых сервисов, предназначенных для работы с большими данными. Эти сервисы позволяют значительно упростить создание и обслуживание сложной инфраструктуры.

   Предлагаемые сервисы:

   – **Yandex Cloud:** Yandex Data Proc (управляемый сервис для Apache Spark™ и Apache Hadoop®), Yandex Managed Service for Apache Kafka®, Yandex Managed Service for ClickHouse®, Yandex Managed Service for Greenplum®, Yandex Managed Service for PostgreSQL, объектное хранилище Yandex Object Storage (совместимое с S3 API).
   – **VK Cloud:** управляемые базы данных (PostgreSQL, ClickHouse, MySQL и др.), сервис «Большие данные» на базе Arenadata Hadoop (ADH) и Arenadata Kafka (ADK) для пакетной и потоковой обработки, объектное хранилище (совместимое с S3 API).

   Механизм генерации конфигураций: при создании и настройке управляемых сервисов через веб-консоль, CLI или API провайдера пользователь указывает высокоуровневые параметры (тип и количество вычислительных узлов, версии ПО, базовые настройки сети и параметры безопасно-

сти). Облачная платформа автоматически генерирует и поддерживает низкоуровневые конфигурации виртуальной инфраструктуры и сервисов (например, *-site.xml для Hadoop, server.properties для Kafka). Возможности кастомизации расширены через дополнительные опции или параметры запуска.

Преимущества:

- значительное упрощение развёртывания и управления: время запуска инфраструктуры сокращается с недель или дней до часов или минут;
- снижение операционной нагрузки: провайдер обеспечивает обновление ПО, патчинг, мониторинг и доступность;
- встроенные механизмы масштабирования и отказоустойчивости;
- интеграция с сервисами экосистемы провайдера.

Недостатки:

- привязка к конкретному провайдеру (vendor lock-in);
- ограниченная гибкость в глубоких настройках;
- высокая стоимость при постоянной нагрузке;
- непрозрачность детальных конфигураций.

В. **Интегрированные платформы данных (Integrated Data Platforms)** на примере российских разработок На российском рынке представлены комплексные решения для жизненного цикла работы с данными, от сбора и хранения до обработки и анализа. Примеры: Arenadata и CedrusData.

- **Arenadata:** продукты на базе открытого кода, включая Arenadata Hadoop (ADH), Arenadata DB (Greenplum), Arenadata Streaming (Kafka, NiFi), Arenadata QuickMarts (ClickHouse). Для управления развёртыванием используется Arenadata Platform Manager (ADPM). Механизм генерации (ADPM): пользователь задаёт через интерфейс или декларативный файл состав кластера и ключевые параметры; ADPM генерирует и применяет конфигурации для всех компонентов, обеспечивая их согласованность.
- **CedrusData:** платформа для высокопроизводительной аналитики и обработки данных, включающая распределённое хранилище, SQL-обработку и инструменты управления. Механизм генерации: инсталлятор или скрипты запрашивают у администратора

параметры системы, после чего генерируются и применяются конфигурации для всех компонентов платформы.

Преимущества:

- единая точка входа и управления;
- проверенные интеграции и оптимальная совместимость;
- упрощённое развёртывание и обновление;
- техническая поддержка от вендора.

Недостатки:

- высокая стоимость лицензий и поддержки;
- «чёрный ящик» внутренних настроек;
- ограниченный выбор компонентов и версий;
- зависимость от экосистемы вендора;
- сложность освоения комплексных платформ.

C. **Kubernetes Operators** Операторы в Kubernetes позволяют управлять stateful-приложениями Big Data: Strimzi для Kafka, CrunchyData и Zalando для PostgreSQL, операторы для ClickHouse, Flink, Spark и др. Механизм генерации: пользователь задаёт CRD (Custom Resource Definition) с высокоуровневым описанием (например, `kind: Kafka, spec:replicas:3, storage:...`), оператор создаёт и управляет Kubernetes-объектами (Deployments, StatefulSets, ConfigMaps, Secrets) в соответствии с этим описанием.

Преимущества: декларативный подход, автоматизация жизненного цикла, нативная модель управления в Kubernetes. Недостатки: требует инфраструктуры и экспертизы в Kubernetes, каждый оператор соответствует одному сервису, интеграции между операторами частично ручные, не генерирует docker-compose.yml и не подходит для сред вне Kubernetes.

D. **Шаблонизаторы и модули для инструментов управления конфигурациями** Ansible, Chef, Puppet, SaltStack с шаблонами (Jinja2, ERB) и готовыми ролями/рецептами для конкретных приложений (PostgreSQL, Kafka и т.д.). Механизм генерации: переменные (например, число брокеров, параметры памяти) задаются в YAML-файлах, шаблоны конфигураций используют эти переменные для генерации файлов, которые затем распространяются на узлы.

Преимущества: гибкость, переиспользование кода, интеграция с существующими CI/CD. Недостатки: требует знания инструментов, описание

системы разбросано между плейбуками, шаблонами и переменными, сложнее задать стек целиком в одном декларативном файле.

Существующие решения предлагают разные уровни абстракции и автоматизации для развёртывания Big Data-систем. Управляемые облачные сервисы и интегрированные платформы обеспечивают высокий уровень автоматизации ценой гибкости и привязки к поставщику. Kubernetes Operators дают декларативность, но требуют экосистемы Kubernetes и экспертизы. Шаблонизаторы в конфигурационных инструментах позволяют генерировать файлы, но требуют значительной ручной настройки.

Подход, разрабатываемый в рамках данной работы, предлагает специализированный инструмент для автоматической генерации конфигураций распространённых Big Data-технологий из единого высокоуровневого YAML-файла, что снижает порог входа и ускоряет итерации при построении и тестировании платформ.

ГЛАВА 2. ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ

В начале XXI века человечество вступило в эпоху информации, характеризующуюся беспрецедентным ростом объемов генерируемых и накапливаемых данных. Этот феномен, получивший название "Большие данные"(Big Data), стал одной из определяющих тенденций технологического и социально-экономического развития. Большие данные описываются не просто их колоссальным объемом (Volume), измеряемым в петабайтах и экзабайтах, но и другими ключевыми характеристиками. К ним относятся высокая скорость (Velocity) поступления и необходимость быстрой обработки, часто в режиме реального времени, а также чрезвычайное разнообразие (Variety) форматов – от структурированных данных в реляционных таблицах до неструктурированных текстов, изображений, видео, аудиопотоков, данных с сенсоров и логов веб-серверов. Нередко выделяют и дополнительные характеристики, такие как достоверность (Veracity), указывающая на неопределенность и возможное наличие шумов, неточностей или пропусков в данных, и ценность (Value), подчеркивающая потенциальную пользу, которую можно извлечь из анализа этих данных [11].

Движущими силами этого информационного взрыва стали повсеместная цифровизация бизнес-процессов, распространение интернета и мобильных

устройств, рост популярности социальных сетей, развитие технологий Интернета вещей (IoT), когда миллиарды устройств непрерывно генерируют данные об окружающей среде и своем состоянии. Традиционные подходы к хранению и обработке данных, основанные на реляционных базах данных и централизованных вычислениях, оказались неспособны эффективно справляться с такими масштабами, скоростью и разнообразием информации [12].

Понимание того, что эти огромные массивы данных содержат скрытые закономерности, тенденции и знания, привело к формированию новой парадигмы. Вместо того чтобы рассматривать данные лишь как побочный продукт деятельности, организации начали видеть в них стратегический актив. Способность собирать, обрабатывать, анализировать большие данные и извлекать из них ценную информацию (insights) превратилась в критически важное конкурентное преимущество. Применение анализа больших данных охватывает практически все сферы деятельности:

- Бизнес и ритейл: Персонализация маркетинговых предложений, оптимизация цепочек поставок, прогнозирование спроса, анализ потребительского поведения, управление рисками, обнаружение мошенничества [13].
- Наука: Ускорение исследований в геномике, астрономии, физике частиц[14], климатологии, материаловедении.
- Здравоохранение: Персонализированная медицина, анализ медицинских изображений, прогнозирование эпидемий, оптимизация работы клиник, разработка новых лекарств.
- Производство: Предиктивное обслуживание оборудования, контроль качества продукции, оптимизация производственных процессов.
- Государственное управление: Улучшение городских служб ("умный город"), анализ транспортных потоков, прогнозирование и предотвращение чрезвычайных ситуаций, повышение эффективности государственных услуг.
- Финансы: Алгоритмический трейдинг, оценка кредитоспособности, выявление финансовых махинаций.

Таким образом, большие данные – это не просто технологический вызов, связанный с хранением и обработкой информации. Это фундаментальный сдвиг, открывающий новые возможности для инноваций, повышения эффективности и создания ценности во всех аспектах человеческой деятельности. Умение работать с большими данными становится необходимым навыком для специалистов

различных профилей, а построение надежной и эффективной инфраструктуры для их обработки – ключевой задачей для организаций, стремящихся сохранить и укрепить свои позиции в современном мире [15]. Неспособность адаптироваться к этой новой реальности и использовать потенциал больших данных может привести к потере конкурентоспособности и отставанию в развитии.

2.1. Программные системы для работы с большими данными

Рассмотрим ключевые компоненты нашей платформы данных: от источников до BI-уровня, указав, для чего каждый инструмент служит, какие задачи решает и какую роль играет в общей системе. Инструменты выбраны исходя из доклада Smart Data 2024 - State of Data RU Edition[10].

1. СУБД PostgreSQL PostgreSQL выступает в платформе в роли традиционного реляционного источника данных[16]. Он хранит структурированную информацию и обеспечивает гарантии ACID[17], сложные SQL-запросы и транзакционные операции. В рамках нашей архитектуры PostgreSQL отвечает за сохранность «исторических» и «оперативных» данных, а также выступает отправной точкой для потоковой репликации Change Data Capture[18] при помощи Debezium[19]. Основная цель PostgreSQL – быть надежным первоисточником, от которого запускаются процессы инкрементального экспорта изменений и пакетной выгрузки данных.
2. S3-хранилище (Minio) Minio в нашей платформе реализует объектное хранилище, совместимое с API Amazon S3[20]. Оно удобно для размещения файловых загрузок, дампов баз данных, бэкапов, а также — в качестве «озеро данных»[21] — для хранения сырых или уже подготовленных дата-сететов в формате Parquet, CSV, JSON и т. д. Minio обеспечивает горизонтальное масштабирование и возможность распределенного хранения больших объемов неструктурированных данных. Его задача – служить долговременным и недорогим репозиторием для «сырых» данных и результатов пакетных обработок.
3. Apache Kafka Kafka выступает в роли распределенного брокера сообщений и обеспечивает надежную передачу сообщений между компонентами системы[22]. Она поддерживает высокую пропускную способность, устойчивость к сбоям и возможность горизонтального масштабирования. В платформе Kafka используется для организации событийного потока

изменений из PostgreSQL (через Debezium) и передачи данных в потребителей – как для потоковой обработки, так и для загрузки в аналитическое хранилище. Kafka гарантирует упорядоченную доставку, сохранение истории сообщений (ретеншн) и управление потребительскими группами.

4. Веб-интерфейс для работы с Kafka - AKHQ AKHQ (ранее «Kafka HQ»)[23] – это веб-интерфейс для администрирования и мониторинга кластера Kafka. Он предоставляет удобный UI для просмотра топиков, чтения и отправки сообщений, управления партициями и оффсетами, мониторинга состояния брокеров и групп потребителей. В платформе AKHQ решает задачу оперативного контроля за событиями в шине: инженер может быстро проверить, какие данные передаются, обнаружить «залипания» потребителей или переполненные топики, а также проводить ручную отладку потоков без необходимости работы с CLI или написания собственных скриптов. Debezium и Kafka Connect Debezium – это платформа Change Data Capture (CDC), реализованная в виде набора коннекторов для Kafka Connect[24]. Kafka Connect, в свою очередь, представляет собой фреймворк для потоковой интеграции: он запускает коннекторы-источники (Source Connectors), которые читают данные из внешних систем (PostgreSQL, MongoDB и пр.), преобразуют их в события и записывают в топики Kafka, а также коннекторы-синкеры (Sink Connectors) для доставки данных из Kafka в хранилища (ClickHouse, S3 и т. д.). В нашей архитектуре Debezium Source Connector подключается к WAL PostgreSQL, транслирует изменения в Kafka, а далее отдельный Sink Connector записывает события в ClickHouse или выполняет другие действия. Цель этого двойного инструмента – обеспечить автоматическую и непрерывную синхронизацию данных между компонентами без написания собственного кода обработки.
5. ClickHouse ClickHouse[25] – это колоночная аналитическая СУБД с высокой скоростью выполнения сложных OLAP-запросов[26] на больших объемах данных. Она оптимизирована для чтения, эффективно сжимает колоночные данные и поддерживает параллельную обработку. В платформе ClickHouse служит основным аналитическим хранилищем, в которое из Kafka через Kafka Connect поступают агрегированные или сырые события. Благодаря своей архитектуре ClickHouse позволяет быстро строить

отчеты, дашборды и выполнять ad-hoc анализ, обрабатывая десятки и сотни миллионов строк за доли секунды.

6. Apache Superset Superset[27] – это BI-платформа с веб-интерфейсом, позволяющая создавать интерактивные дашборды, визуализации и аналитические отчеты поверх различных источников данных (SQL-базы, колоночные хранилища, дата-видео и т. д.). В нашей системе Superset подключается к ClickHouse и предоставляет бизнес-пользователям и аналитикам средства для построения графиков, таблиц, гео-карт и скриптов на SQL. Его цель – закрыть уровень представления данных: от сложных SQL-запросов непосредственного взаимодействия до простого drag-and–drop интерфейса для быстрого получения инсайтов.

В совокупности эти инструменты образуют сквозной конвейер данных: от первичного сбора и хранения, через трансформации и передачу событий, до хранения в аналитической СУБД и визуализации для конечных пользователей. Такое разделение ответственности позволяет использовать лучшие свойства каждого компонента и легко масштабировать или заменять отдельные части платформы по мере роста требований и объемов данных. Именно поэтому в рамках настоящей работы рассматривается вопрос создания инструмента, призванного облегчить рутинную работу инженера данных по настройке многочисленных компонентов, а также предоставить ему удобное средство для декларативного описания и развертывания конфигураций платформ данных. Исходя из всего вышеперечисленного, Use-Case диаграмма (диаграмма вариантов использования) разработанного инструмента dpd выглядит следующим образом (рис.2.1)

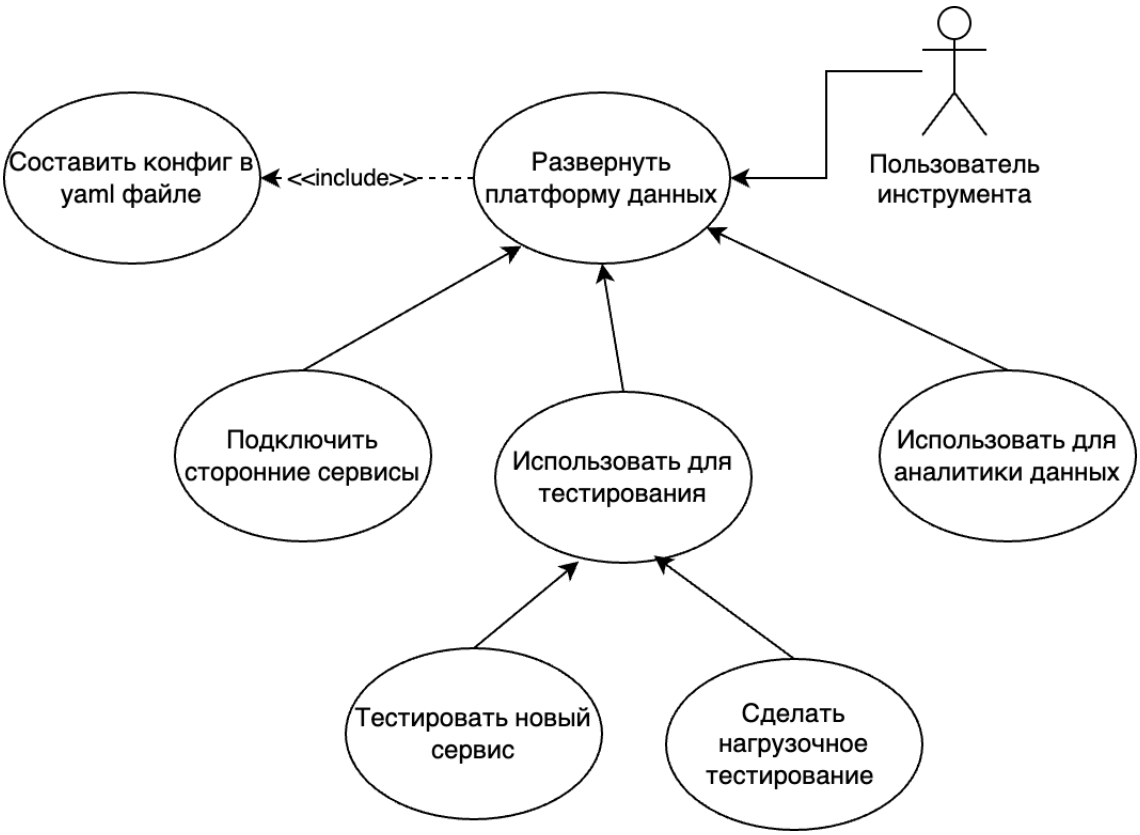

Рис.2.1. Use-Case диаграмма

ГЛАВА 3. РАЗРАБОТКА ИНСТРУМЕНТА

Предварительно важно определить чёткую последовательность этапов, которая позволит организовать работу над инструментом системно и управляемо. Это поможет сократить риски неопределенности, оптимально распределить ресурсы и обеспечить своевременный контроль качества на каждом шаге.

Хорошим стилем является наличие введения к главе. Во введении может быть описана цель написания главы, а также приведена краткая структура главы.

3.1. План разработки инструмента

Этап 1. Сбор и анализ требований На этом этапе формализуются функциональные и нефункциональные требования: определяется, какие компоненты Big Data стека поддерживаются, в каком формате задается исходный YAML, какие конфигурационные артефакты должны генерироваться.

Этап 2. Проектирование архитектуры Разрабатывается модульная архитектура инструмента, включающая парсер входного описания, ге-

нератор промежуточного представления (AST), набор шаблонов для конфигураций инструментов и механизм их объединения в итоговые файлы. Определяются границы ответственности каждого модуля, протокол взаимодействия между ними и формат плагинов для расширения функциональности.

Этап 3. Определение языка декларативного описания Уточняются синтаксис и семантика входного YAML: структура разделов, типы параметров, возможные зависимости и проверки корректности. Разрабатывается схема jsonschema для валидации пользовательских описаний на раннем этапе.

Этап 4. Реализация ядра: парсер и промежуточное представление Пишется компонент, который читает декларативный файл, проводит его валидацию по схеме (jsonschema), конструирует внутреннее дерево объектов (AST) с отображением всех сущностей и их связей. Этот модуль обеспечивает основу для дальнейших операций по генерации конфигураций.

Этап 5. Разработка генераторов конфигурационных файлов На основе AST реализуются плагины-генераторы для каждого типа артефакта:

- docker-compose.yaml с сервисами и сетями;
- Конфигурации PostgreSQL (postgresql.conf, init.sql);
- JSON-файлы коннекторов Debezium и S3Sink;
- Файлы настроек для ClickHouse;
- Конфигурационные файлы для AKHQ и Superset.

Каждый генератор использует шаблонизатор и преобразует параметры из AST в конкретные строки и блоки файлов.

Этап 6. Создание CLI-интерфейса Реализуется утилита командной строки, позволяющая пользователю запускать генерацию: передавать путь к входному файлу, указывать директорию вывода, включать опции валидации и отладки. CLI обеспечивает удобство использования инструмента в скриптах и CI/CD-пайплайнах [28].

Этап 7. Модуль тестирования и валидации Пишутся автоматические тесты: модульные тесты для парсера и генераторов, интеграционные — для проверки корректного результата генерации по ряду типовых YAML-конфигураций. Добавляются проверки на соответствие с

эталонными файлами и на корректность в Docker-среде (например, пробный запуск docker-compose up).

Этап 8. Документирование и примеры Готовится подробная документация: описание формата входного файла, руководство пользователя CLI, схемы и примеры конфигураций «из коробки» для типовых сценариев (EDW на PostgreSQL→Kafka→ClickHouse→Superset). Включаются рекомендации по расширению и отладке.

Этап 9. Пилотное развертывание и сбор обратной связи Инструмент разворачивается в тестовой среде или локально на реальных примерах, собираются отзывы от пользователей — инженеров и аналитиков. На основе полученных замечаний корректируются шаблоны, схемы и UX CLI.

Этап 10. Релиз и сопровождение Формируется релизная сборка, обеспечивается публикация в открытый репозиторий GitHub, настраивается процесс выпуска обновлений и приёма issue. Определяется модель поддержки: дорожная карта, приоритеты новых возможностей и исправлений.

3.2. Язык декларативного описания (DSL)

Практический опыт показывает, что повышение уровня абстракции и учёт специфики предметной области наиболее эффективно достигаются через разработку собственного языка предметной области - Domain Specific Language, DSL[3]. Такой язык представляет собой формальный аппарат, работающий непосредственно с понятиями и структурами предметной области, позволяя лаконично формулировать и решать большинство типовых задач.

В нашем случае DSL строится на основе YAML[30] – удобного человеко-ориентированного формата сериализации, концептуально схожего с языками разметки, но оптимизированного для записи и чтения распространённых структур данных.

Ключевые особенности синтаксиса YAML:

1. Отступы и вложенность

   Используются только пробелы (обычно 2 или 4) для обозначения уровней вложенности. Символ табуляции запрещён.

2. Пары «ключ–значение»
   Каждая запись имеет вид ключ: значение, где после двоеточия обязательно идёт пробел.
3. Списки
   Элементы маркируются дефисом и пробелом (- элемент).
4. Многострочные литералы
   Символ | сохраняет все разрывы строк. Символ > объединяет строки, заменяя отступы и разрывы единичными пробелами.
5. Якоря и ссылки
   Якорь (&имя) позволяет дать имя блоку значений. Ссылка (*имя) повторно вставляет ранее объявленный блок.
   Пример:

```
default: &base
имя: Oleg
возраст: 27

user_2:
<<: *base
```

Чтобы формализовать синтаксис DSL и задать конечное описание потенциально бесконечного множества допустимых конфигураций, мы опираемся на контекстно–свободную грамматику $G = \langle N,T,R,S\rangle$, где:
N – множество нетерминальных символов
T – терминальные (т. е. реальные лексемы)
R – правило вида $A \to \alpha$ (замена нетерминала A на строку символов α)
S – стартовый нетерминал.

По классификации Хомского такая грамматика относится ко второму типу (КСГ): в каждом правиле слева стоит ровно один нетерминал, который может быть заменён на любую допустимую цепочку из $A \cup B$.

Реализация парсера и генератора AST (абстрактного синтаксического дерева) опирается на ANTLR (ANother Tool for Language Recognition)[31]. Лексические правила (начинаются с большой буквы) описывают, как разбить входной текст на токены. Пример:

```
// Лексическое правило для целых чисел
```

```
INT : [0-9]+ ;
```

Синтаксические правила (начинаются с маленькой буквы) задают структуры из токенов. Пример:

```
// Синтаксическое правило для списка аргументов
args : expr (',' expr)* ;
```

Для группировки, повторений и альтернатив в ANTLR применяются:
«()» – группировка
«*» – 0 или более повторений
« +» – 1 или более
«?» – 0 или 1 раз
« |» – выбор одной из альтернатив
«:» и «;» – разделители начала и конца правил.

Описание языка DPD (Data Platform Deployer)

Язык DPD разработан для декларативного описания архитектуры платформы данных единым удобным форматом и автоматической генерации всех необходимых инструментов для быстрого развертывания и тестирования готового стенда. В общих чертах имеет следующую структуру:

```
project:
    name: data-platform-14
    version: 1.2.0
    description: This is a project for testing data platform
sources:
    - type: postgres
    name: postgres_1
    - type: postgres
    name: postgres_2
    - type: s3
    name: s3_1
streaming:
    kafka:
        num_brokers: 3
```

```
    connect:
        name: connect-1
storage:
    clickhouse:
    name: clickhouse-1
bi:
    superset:
    name: superset-1
```

В таблице (?) приведена часть полной грамматики, описывающая выборочные правила.

Грамматика ANTLR4	Комментарий
`grammar ConfigDSL;`	

`configFile`
`: projectDef`
`sourcesDef`
`streamingDef`
`storageDef`
`biDef EOF`
`;`
	Корневое правило: определяет общую структуру конфигурационного файла, состоящего из последовательных блоков

`projectDef`
`: PROJECT COLON NAME COLON`
`STRING VERSION COLON STRING`
`DESCRIPTION COLON STRING`
`;`
	Правило для секции "project": описывает метаданные проекта

`sourcesDef`
`: SOURCES COLON`
`sourceItem+`
`;`
	Правило для секции «sources»: определяет список источников данных.

sourceItem : DASH NAME COLON STRING TYPE COLON sourceType (PORT COLON NUMBER)? (USERNAME COLON STRING)? (PASSWORD COLON STRING)? (ACCESS_KEY COLON STRING)? (SECRET_KEY COLON STRING)? (REGION COLON STRING)? (BUCKET COLON STRING)? ;	Правило для описания одного источника данных: имя, тип (Postgres/S3) и опциональные параметры (порт, учётные данные, детали S3).
sourceType : POSTGRES | S3 ;	Правило для определения типа источника данных (PostgreSQL или S3).
streamingDef : STREAMING COLON (kafkaDef | connectDef)+ ;	Правило для секции «streaming»: задаёт компоненты потоковой обработки (Kafka или Kafka Connect).
kafkaDef : KAFKA COLON NUM_BROKERS COLON NUMBER ;	Правило для конфигурации Kafka: число брокеров.
connectDef : CONNECT COLON NAME COLON STRING ;	Правило для конфигурации Kafka Connect: имя инстанса.

storageDef : STORAGE COLON clickhouseDef ; clickhouseDef : CLICKHOUSE COLON NAME COLON STRING ;	Правило для секции «storage»: задаёт компонент хранения данных и его параметры (ClickHouse).
biDef : BI COLON supersetDef ; supersetDef : SUPERSET COLON NAME COLON STRING (USERNAME COLON STRING)? (PASSWORD COLON STRING)? ;	Правило для секции «bi»: задаёт инструмент BI (Apache Superset) и его опциональные параметры.

```
// Лексемы (tokens)
PROJECT     : 'project';
SOURCES     : 'sources';
STREAMING   : 'streaming';
STORAGE     : 'storage';
BI          : 'bi';
KAFKA       : 'kafka';
CONNECT     : 'connect';
CLICKHOUSE  : 'clickhouse';
SUPERSET    : 'superset';
NAME        : 'name';
VERSION     : 'version';
DESCRIPTION : 'description';
TYPE        : 'type';
PORT        : 'port';
USERNAME    : 'username';
PASSWORD    : 'password';
ACCESS_KEY  : 'access_key';
SECRET_KEY  : 'secret_key';
REGION      : 'region';
BUCKET      : 'bucket';
NUM_BROKERS : 'num_brokers';
POSTGRES    : 'postgres';
S3          : 's3';
COLON       : ':';
DASH        : '-';
STRING      : '"' ( ~[\\"] | '\\' . )*? '"' ;
NUMBER      : [0-9]+ ;
WS          : [ \t\r\n]+ -> skip ;
```

Лексемы (tokens): ключевые слова, разделители, строковые и числовые литералы, пробельные символы.

3.3. Выводы

Текст выводов по главе 3.

ГЛАВА 4. НАЗВАНИЕ ЧЕТВЁРТОЙ ГЛАВЫ. АПРОБАЦИЯ РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЯ, А ИМЕННО: МЕТОДА, АЛГОРИТМА, МОДЕЛИ ИССЛЕДОВАНИЯ

Хорошим стилем является наличие введения к главе. Во введении может быть описана цель написания главы, а также приведена краткая структура главы.

4.1. Название параграфа

4.2. Название параграфа

Пример ссылки на литературу [1—4].

4.3. Выводы

Текст выводов по главе 4.

ЗАКЛЮЧЕНИЕ

Заключение (2 – 5 страниц) обязательно содержит выводы по теме работы, *конкретные предложения и рекомендации* по исследуемым вопросам. Количество общих выводов должно вытекать из количества задач, сформулированных во введении выпускной квалификационной работы.

Предложения и рекомендации должны быть органически увязаны с выводами и направлены на улучшение функционирования исследуемого объекта. При разработке предложений и рекомендаций обращается внимание на их обоснованность, реальность и практическую приемлемость.

Заключение не должно содержать новой информации, положений, выводов и т. д., которые до этого не рассматривались в выпускной квалификационной работе. Рекомендуется писать заключение в виде тезисов.

Последним абзацем в заключении можно выразить благодарность всем людям, которые помогали автору в написании ВКР.

СЛОВАРЬ ТЕРМИНОВ

TeX — язык вёрстки текста и издательская система, разработанные Дональдом Кнутом.

LaTeX — язык вёрстки текста и издательская система, разработанные Лэсли Лампортом как надстройка над TeX.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Автономова Н. С.* Философский язык Жака Деррида. — М.: Российская политическая энциклопедия (РОССПЭН), 2011. — 510 с. — (Сер.: Российские Пропилеи).

2. *Котельников И. А.*, *Чеботаев П. З.* LaTeX по-русски. — 3-е изд. — Новосибирск: Сибирский Хронограф, 2004. — 496 с. — URL: http://www.tex.uniyar.ac.ru/doc/kotelnikovchebotaev2004b.pdf (дата обращения: 06.03.2019).

3. *Песков Н. В.* Поиск информативных фрагментов описаний объектов в задачах распознавания: дис. . . . канд. канд. физ.-мат. наук: 05.13.17 / Песков Николай Владимирович. — М., 2004. — 102 с.

4. *Kotelnikov I. A.*, *Chebotaev P. Z.* LaTeX in Russian. — 3rd ed. — Novosibirsk: Sibiskiy Hronograph, 2004. — 496 p. — URL: http://www.tex.uniyar.ac.ru/doc/kotelnikovchebotaev2004b.pdf (visited on 06.03.2019); (in Russian).

5. *Peskov N. V.* Searching for informative fragments of object descriptions in the recognition tasks: diss. … cand. phys.-math. sci.: 05.13.17 / Peskov Nickolay Vladimirovich. — M., 2004. — 102 p. — (in Russian).

6. SPbPU photo gallery. — URL: http://www.spbstu.ru/media/photo-gallery/ (visited on 06.03.2019).

7. TeX Live web site. — URL: https://www.tug.org/texlive/ (visited on 30.01.2025).

8. TeXstudio web site. — URL: https://www.texstudio.org/ (visited on 06.03.2019).

Приложение 1

Краткие инструкции по настройке издательской системы LaTeX

В SPbPU-BCI-template *автоматически выставляются необходимые настройки и в исходном тексте шаблона приведены примеры оформления текстово-графических объектов*, поэтому авторам достаточно заполнить имеющийся шаблон текстом главы (статьи), не вдаваясь в детали оформления, описанные далее. Возможный «быстрый старт» оформления главы (статьи) под Windows следующий[П1.1]:

A. Установка полной версии TeX Live [7]. В процессе установки лучше выставить параметр доустановки пакетов «на лету».
B. Установка TexStudio [8].
C. Запуск TexStudio и компиляция `my_chapter.tex` с помощью команды «Build&View» (например, с помощью двойной зелёной стрелки в верхней панели). *Иногда, для достижения нужного результата необходимо несколько раз скомпилировать документ.*
D. В случае, если не отобразилась библиография, можно
   - воспользоваться командой Tools → Commands → Biber, затем запустив Build&View;
   - настроить автоматическое включение библиографии в настройках Options → Configure TexStudio → Build → Build&View (оставить по умолчанию, если сборка происходит слишком долго): `txs:///pdflatex | txs:///biber | txs:///pdflatex | txs:///pdflatex | txs:///view-pdf`.

В случае возникновения ошибок, попробуйте скомпилировать документ до последних действий или внимательно ознакомьтесь с описанием проблемы в log-файле. Бывает полезным переход (по подсказке TexStudio) в нужную строку в pdf-файле или запрос с текстом ошибке в поисковиках. Наиболее вероятной проблемой при первой компиляции может быть отсутствие какого-либо установленного пакета LaTeX.

В случае корректной работы настройки «установка на лету» все дополнительные пакеты будут скачиваться и устанавливаться в автоматическом режиме. Если доустановка пакетов осуществляется медленно (несколько пакетов за один запуск

[П1.1] Внимание! Пример оформления подстрочной ссылки (сноски).

компилятора), то можно попробовать установить их в ручном режиме следующим образом:

1. Запустите программу: меню → все программы → MikTeX → Maintenance (Admin) → MiKTeX Package Manager (Admin).
2. Пользуясь поиском, убедитесь, что нужный пакет присутствует, но не установлен (если пакет отсутствует воспользуйтесь сначала MiKTeX Update (Admin)).
3. Выделив строку с пакетом (возможно выбрать несколько или вообще все неустановленные пакеты), выполните установку Tools → Install или с помощью контекстного меню.
4. После завершения установки запустите программу MiKTeX Settings (Admin).
5. Обновите базу данных имен файлов Refresh FNDB.

Для проверки текста статьи на русском языке полезно также воспользоваться настройками Options → Configure TexStudio → Language Checking → Default Language. Если русский язык «ru_RU» не будет доступен в меню выбора, то необходимо вначале выполнить Import Dictionary, скачав из интернета любой русскоязычный словарь.

Далее приведены формулы (П1.2), (П1.1), рис.П1.2, рис.П1.1, табл.П1.2, табл.П1.1.

$$\pi \approx 3{,}141. \tag{П1.1}$$

Рис.П1.1. Вид на гидробашню СПбПУ [6]

Таблица П1.1

Представление данных для сквозного примера по ВКР [5]

G	m_1	m_2	m_3	m_4	K
g_1	0	1	1	0	1
g_2	1	2	0	1	1
g_3	0	1	0	1	1
g_4	1	2	1	0	2
g_5	1	1	0	1	2
g_6	1	1	1	2	2

П1.1. Параграф приложения

П1.1.1. Название подпараграфа

Название подпараграфа оформляется с помощью команды `\subsection{...}`. Использование подподпараграфов в основной части крайне не рекомендуется.

П1.1.1.1. Название подподпараграфа

$$\pi \approx 3{,}141. \tag{П1.2}$$

Рис.П1.2. Вид на гидробашню СПбПУ [6]

Таблица П1.2

Представление данных для сквозного примера по ВКР [5]

G	m_1	m_2	m_3	m_4	K
g_1	0	1	1	0	1
g_2	1	2	0	1	1
g_3	0	1	0	1	1
g_4	1	2	1	0	2
g_5	1	1	0	1	2
g_6	1	1	1	2	2

Приложение 2

Некоторые дополнительные примеры

В приложении приведены формулы (П2.2), (П2.1), рис.П2.2, рис.П2.1, табл.П2.2, табл.П2.1[П2.1].

$$\pi \approx 3{,}141. \qquad \text{(П2.1)}$$

Рис.П2.1. Вид на гидробашню СПбПУ [6]

Таблица П2.1

Представление данных для сквозного примера по ВКР [5]

G	m_1	m_2	m_3	m_4	K
g_1	0	1	1	0	1
g_2	1	2	0	1	1
g_3	0	1	0	1	1
g_4	1	2	1	0	2
g_5	1	1	0	1	2
g_6	1	1	1	2	2

[П2.1] Внимание! Пример оформления подстрочной ссылки (сноски).

П2.1. Подраздел приложения

$$\pi \approx 3{,}141. \qquad \text{(П2.2)}$$

Рис.П2.2. Вид на гидробашню СПбПУ [6]

Таблица П2.2

Представление данных для сквозного примера по ВКР [5]

G	m_1	m_2	m_3	m_4	K
g_1	0	1	1	0	1
g_2	1	2	0	1	1
g_3	0	1	0	1	1
g_4	1	2	1	0	2
g_5	1	1	0	1	2
g_6	1	1	1	2	2