



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» Кафедра 806
Группа М8О-406Б-21 Направление подготовки 01.03.02 «Прикладная математика и
информатика»

Профиль Информатика

Квалификация: бакалавр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему: Проектирование системы переноса и генерации взаимосвязанных
данных из производственной среды при тестировании образовательной
платформы

Автор ВКРБ:	Мезенин Олег Александрович	(_____)
Руководитель:	Миронов Евгений Сергеевич	(_____)
Консультант:	Ляпина Светлана Юрьевна	(_____)
Консультант:	—	(_____)
Рецензент:	—	(_____)

К защите допустить

Заведующий кафедрой № 806	Крылов Сергей Сергеевич	(_____)
---------------------------	-------------------------	---------

_____ мая 2025 года

Москва 2025

РЕФЕРАТ

Выпускная квалификационная работа бакалавра состоит из 25 страниц, 5 рисунков, 5 использованных источников, 1 приложения.

TBD: расписать keywords

TBD: расписать анотацию

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ	4
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	5
ВВЕДЕНИЕ	6
1 Постановка задачи и теоретические предпосылки	9
1.1 Тестирование	9
1.1.1 Процесс тестирования	9
1.2 Ручное и автоматизированное тестирование	10
1.3 Сценарии использования	11
1.3.1 Перенос данных для проведения тестовых сценариев	11
1.3.2 Диагностика ошибки	12
1.3.3 Тестирование новой миграции	13
1.3.4 Перенос данных для автоматизированного тестирования	14
1.3.5 Генерация данных для проведения тестовых сценариев	14
1.4 Определение требований к системе	15
1.4.1 Функциональные требования	16
1.4.2 Требования к свойствам архитектуры	16
2 Проектирование системы, разработка языка и алгоритма	18
2.1 Архитектура системы	18
2.2 Метаграфы	21
2.3 Алгоритм переноса данных	22
ЗАКЛЮЧЕНИЕ	23
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	24
ПРИЛОЖЕНИЕ А Исходный код	25

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей выпускной квалификационной работе бакалавра применяют следующие термины с соответствующими определениями:

Перенос данных — это процесс копирования и перемещения данных из одного источника или системы в другую без удаления данных из оригинальной базы данных

CI/CD — сокращение от Continuous Integration и Continuous Deployment, представляет собой совокупность практик и инструментов, применяемых в области разработки программного обеспечения с целью автоматизации и оптимизации процессов интеграции, тестирования и развёртывания приложений

DevOps-инженер — это специалист, работающий на стыке разработки программного обеспечения (development) и его эксплуатации (operations), задачей которого является создание надёжного и эффективного процесса разработки, тестирования и развёртывания приложений

CLI — сокращение от Command Line Interface, представляет собой интерфейс взаимодействия пользователя с операционной системой или программным обеспечением, который осуществляется посредством ввода текстовых команд в терминал

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

БД — база данных

СУБД — система управления базами данных

ВВЕДЕНИЕ

В настоящее время рынок онлайн-образования демонстрирует значительный рост. Всё большее число людей различных возрастных категорий предпочитают дистанционные форматы, которые обеспечивают гибкость и доступность обучения независимо от географического положения. Сегмент онлайн-образования, направленный на школьников, составляет значительную часть рынка. [1] В связи с этим внимание следует уделить платформам, разработанным для школьников, их родителей и учителей.

Подобные системы позволяют объединить учебные материалы, задания, оценки и обратную связь в одном пространстве. Для школьников это обеспечивает обучение в интерактивной и увлекательной форме, для родителей — возможность контролировать достижения своих детей и участвовать в их образовательном процессе, а для учителей — эффективное управление учебным процессом и возможность персонализированного подхода в обучении.

Эти платформы требуют надёжной и эффективной системы управления данными для обеспечения стабильной работы и поддержки большого количества пользователей. Одним из распространённых решений в области управления базами данных является PostgreSQL [2] — реляционная система управления базами данных с открытым исходным кодом.

В процессе разработки платформы важно не только работать над техническими аспектами, но и уделять особое внимание качеству и надёжности. Это делает тестирование незаменимой стадией разработки. Благодаря тестированию можно убедиться в стабильности и бесперебойной работе систем.

Использование реальных данных и проведение тестов в производственной среде позволяет максимально точно оценить работоспособность системы. Но такой подход сопряжён с рисками, такими как нарушения конфиденциальности данных, потенциальные сбои в работе системы и возможное негативное влияние на реальных пользователей.

Чтобы тестирование образовательных платформ было безопасным и в то же время эффективным, необходимо создать тестовую среду, максимально приближённую к производственным условиям. Создание тестовой среды требует тщательной проработки и учета всех аспектов, связанных с

функционированием платформы. Это включает в себя настройку инфраструктуры, идентичной производственной, и воспроизведение всех взаимосвязей данных, которые присутствуют в реальных условиях эксплуатации.

Особую значимость в этом контексте приобретает работа с данными. Для достижения максимальной схожести с производственной средой необходимо использовать тестовые данные, которые точно отражают объёмы, структуры и взаимосвязи данных, присутствующие в реальной системе.

Создание тестового набора данных может быть реализовано с использованием различных методик. Один из подходов включает полный перенос данных из производственной базы данных с последующей анонимизацией конфиденциальной информации. Утилита `pg_dump` [3] может быть использована для экспорта данных из PostgreSQL, обеспечивая перенос структуры и содержимого базы данных.

Для защиты пользовательской информации и анонимизации данных можно применить утилиту `pg_anonymizer` [4], которая помогает скрыть чувствительные данные, заменяя их на фиктивные значения, что сохраняет конфиденциальность пользователей.

Несмотря на эффективное использование таких средств, основной недостаток данного метода заключается в невысокой скорости переноса всех данных, что может быть критичным при работе с крупными наборами данных и требовать значительных временных затрат для выполнения всей операции.

Частичный перенос данных может быть более эффективным для тестирования, так как зачастую не требуется полный объём данных для покрытия большинства сценариев. Однако этот подход сопряжён с рядом вопросов и потенциальных сложностей.

При полном переносе данных тестирующий получает доступ ко всем данным, что позволяет ему проводить тестовые сценарии. В случае частичного переноса данных требуется определить и описать необходимые для тестирования данные, избегая глубокого изучения структуры базы данных и её связей. Возникает вопрос: каким образом тестирующему эффективно описать данные, нужные для проведения тестов?

Следующим вопросом является сохранение целостности данных. В случае полного переноса реляционная целостность поддерживается автоматически, поскольку все взаимосвязанные данные переносятся

полностью, обеспечивая корректность и согласованность системы. При частичном переносе необходимо уделить особое внимание поддержанию целостности данных. Недопустимо, чтобы после переноса в тестовой среде данных было недостаточно для выполнения тестовых сценариев. Следовательно, второй вопрос заключается в следующем: как обеспечить целостность данных, минимизируя при этом объём переносимых данных?

Перенос данных из производственной среды с использованием анонимизации представляет собой лишь один из способов получения тестовых данных. Альтернативный подход состоит в генерации данных на основе заданных характеристик. Как может быть реализован этот процесс? Пользователь задаёт характеристики, такие как возраст, пол или средний балл по математике; каждый из параметров может сопровождаться набором или диапазоном значений. Программа затем генерирует тестовые данные, основываясь на фактических данных, либо на схеме данных производственной базы, а также на указанных пользователем характеристиках. Таким образом, полученные данные будут иметь структуру, аналогичную реальным данным, но не будут связаны с конкретными записями в производственной базе. Но для этого метода также возникают два ключевых вопроса: как пользователю описать необходимые данные и как их корректно сгенерировать.

Дипломная работа будет посвящена проектированию системы, обеспечивающей перенос взаимосвязанных данных из одной базы данных в другую на основе пользовательских запросов с применением методов анонимизации, а также способной генерировать тестовые данные. В рамках исследования будут рассмотрены алгоритмы переноса и генерации данных, а также разработка языка для описания данных, необходимых пользователю.

1 Постановка задачи и теоретические предпосылки

1.1 Тестирование

Чтобы определить требования к нашей системе, для начала нужно разобраться с основными аспектами тестирования программного обеспечения.

1.1.1 Процесс тестирования

Тестирование — это процесс оценки системы или её компонентов с целью проверки их соответствия заданным требованиям. Основная цель тестирования заключается в выявлении дефектов, обеспечении качества и подтверждении работоспособности программного обеспечения.

Процесс тестирования можно разбить на несколько ключевых этапов [5]:

а) Планирование:

- разрабатывается стратегия тестирования, включая выбор типов тестирования;
- создается тест-план, где описываются ресурсы, сроки, и критерии начала и завершения тестирования;
- проводится анализ рисков, связанных с тестированием и определяются действия для минимизации их воздействия.

б) Подготовка:

- разработка тестовых случаев и сценариев на основе требований и спецификаций;
- подготовка тестовой среды, включая оборудование и программные средства, необходимые для проведения тестов;
- подготовка данных для тестирования и настройка автоматизированных тестов, если они предусмотрены.

в) Проведение:

- непосредственное выполнение тестов согласно плану тестирования;
- фиксация и документирование результатов тестирования, выявление дефектов и несоответствий.

г) Совершенствование:

- анализ результатов тестирования и составление отчёта по итогам выполненной работы;
- обмен опытом и полученными знаниями внутри команды и с другими заинтересованными сторонами;
- обновление и улучшение тестовых документов и процессов на основе полученных данных.

Наша система будет применяться на этапе подготовки: с её помощью можно будет создавать тестовые данные.

1.2 Ручное и автоматизированное тестирование

Тестирование можно классифицировать на ручное и автоматизированное.

Ручное тестирование предполагает процесс, при котором тестировщик выполняет тесты без помощи автоматизированных инструментов. Оно позволяет глубже вникнуть в пользовательский опыт, и его легко применять к новым или часто изменяющимся функциональностям, однако оно может быть времяемким, подвержено человеческим ошибкам и не всегда позволяет повторить результаты для их сравнения.

Автоматизированное тестирование предполагает использование программных инструментов для автоматизации выполнения тестовых сценариев. Это предполагает разработку скриптов, которые автоматически выполняют тесты и сверяют результаты с ожидаемыми. Автоматизированное тестирование обеспечивает высокую скорость и стабильность результатов, что делает его эффективным для регрессионного тестирования и стабильных функциональностей, однако требует значительных ресурсов для разработки скриптов и не всегда подходит для тестирования пользовательских интерфейсов и динамически изменяющихся требований.

Оба подхода часто используются совместно для достижения наилучшего результата. Обычно ручное тестирование применяют на начальных стадиях или для исследовательского тестирования, тогда как автоматизированное тестирование используют для повторяющихся задач или сложных сценариев, которые требуют стабильного выполнения.

1.3 Сценарии использования

Перед определением требований к нашей системе рассмотрим возможные сценарии использования.

Определим роли:

- тестировщик — специалист, занимающийся проверкой качества, надёжности и работоспособности программного обеспечения;
- разработчик — специалист, занимающийся разработкой программного обеспечения;
- DevOps-инженер — специалист, занимающийся поддержкой процессов CI/CD.

Будем называть метаданными входные данные, включающие в себя правила переноса, генерации или анонимизации данных.

1.3.1 Перенос данных для проведения тестовых сценариев

Роли:

- тестировщик;
- разработчик.

Цель: создать тестовые данные на основе существующих реальных данных для их дальнейшего применения в тестовых сценариях.

Предпосылки:

- разработчиком написаны метаданные, включающие в себя правила переноса и анонимизации данных;
- тестировщик осведомлен о метаданных, необходимых для проведения тестовых сценариев;
- тестировщик имеет уникальный идентификатор исходной базы данных;
- имеется пустая тестовая база данных, доступ к которой имеется у тестировщика.

Основной сценарий:

- а) Тестировщик осуществляет изменения в метаданных при необходимости.
- б) Тестировщик инициирует запуск CLI, указывая следующие параметры:

- уникальный идентификатор исходной базы;
- уникальный идентификатор целевой базы или параметры подключения к ней;
- метаданные или уникальный идентификатор метаданных.

в) Тестировщик ожидает завершения процесса переноса данных.

Постусловия:

- тестировщик применяет данные, сгенерированные на основе реальных данных, в тестовых сценариях.

Исключения:

- тестировщик получает ошибку некорректных метаданных, если он менял их, после пункта б.

1.3.2 Диагностика ошибки

Роли:

- тестировщик.

Цель: провести диагностику ошибки, возникшей с определенными данными в продуктивной среде.

Предпосылки:

- в продуктивной среде обнаружены проблемы с определенными данными, такими как данные пользователя;
- тестировщик владеет уникальным идентификатором данных, в которых выявлена ошибка в продуктивной среде;
- тестировщик имеет уникальный идентификатор исходной базы данных;
- имеется пустая тестовая база данных, доступ к которой имеется у тестировщика.

Основной сценарий:

- а) Тестировщик описывает метаданные, включая данные, в которых произошла ошибка.
- б) Тестировщик инициирует запуск CLI, указывая следующие параметры:
 - уникальный идентификатор исходной базы;
 - уникальный идентификатор целевой базы или параметры подключения к ней;
 - метаданные.

в) Тестировщик ожидает завершения процесса переноса данных.

Альтернативный сценарий:

- тестировщик может взять готовые метаданные, если они ему подходит, а не описывать свои.

Постусловия:

- тестировщик осуществляет взаимодействие с перенесенными данными и проводит диагностику ошибки.

Исключения:

- тестировщик получает ошибку некорректных метаданных после пункта б.

1.3.3 Тестирование новой миграции

Роли:

- тестировщик;
- разработчик.

Цель: провести тестирование новой миграции.

Предпосылки:

- тестировщик имеет уникальный идентификатор исходной базы данных;
- имеется пустая тестовая база данных, доступ к которой имеется у тестировщика;
- разработчик создал новую миграцию, нуждающуюся в тестировании;
- разработчик предоставил метаданные к миграции, содержащие правила анонимизации и переноса данных.

Основной сценарий:

а) Тестировщик инициирует запуск CLI, указывая следующие параметры:

- уникальный идентификатор исходной базы;
- уникальный идентификатор целевой базы или параметры подключения к ней;
- предоставленные метаданные.

б) Тестировщик ожидает завершения процесса переноса данных.

Постусловия:

- тестировщик взаимодействует с перенесенными данными и

осуществляет тестирование миграций.

1.3.4 Перенос данных для автоматизированного тестирования

Роли:

- тестировщик;
- разработчик;
- DevOps-инженер.

Цель: обеспечить возможность проведения автоматизированного тестирования изменений через сценарий CI/CD.

Предпосылки:

- разработчиком написаны метаданные, включающие в себя правила переноса и анонимизации данных;
- тестировщиком реализованы автоматизированные тесты;
- DevOps-инженер разработал сценарий CI/CD, который, в случае появления нового запроса на слияния в системе контроля версий, инициирует развёртывание тестового окружения с пустой базой данных и осуществляет перенос необходимых данных с помощью нашей системы.

Основной сценарий:

- а) Разработчик вносит изменения в исходный код и создает новый запрос на слияние в системе контроля версий.
- б) Запускается процесс CI/CD, который выполняет следующие действия:
 - развёртывание тестового окружения с пустой базой данных;
 - перенос необходимых данных в тестовую базу с помощью нашей системы;
 - автоматический запуск тестов для проверки корректности внесённых изменений.

1.3.5 Генерация данных для проведения тестовых сценариев

Роли:

- тестировщик;
- разработчик.

Цель: создать тестовые данные на основе существующих реальных данных для их дальнейшего применения в тестовых сценариях.

Предпосылки:

- разработчиком написаны метаданные, включающие в себя правила генерации данных;
- тестировщик осведомлен о метаданных, необходимых для проведения тестовых сценариев;
- тестировщик имеет уникальный идентификатор исходной базы данных;
- имеется пустая тестовая база данных, доступ к которой имеется у тестировщика.

Основной сценарий:

- а) Тестировщик осуществляет изменения в метаданных при необходимости.
- б) Тестировщик инициирует запуск CLI, указывая следующие параметры:
 - уникальный идентификатор исходной базы;
 - уникальный идентификатор целевой базы или параметры подключения к ней;
 - метаданные или уникальный идентификатор метаданных.
- в) Тестировщик ожидает завершения процесса генерации данных.

Постусловия:

- тестировщик применяет сгенерированные данные в тестовых сценариях.

Исключения:

- тестировщик получает ошибку некорректных метаданных, если он менял их, после пункта б.

1.4 Определение требований к системе

Требования к системе можно классифицировать на две основные категории: функциональные и архитектурные. Функциональные требования описывают конкретный набор функций и возможностей, которые система должна обеспечивать для удовлетворения потребностей пользователей. Требования к свойствам архитектуры системы фокусируются на качественных характеристиках системы, которые обеспечивают её

устойчивую и эффективную работу.

1.4.1 Функциональные требования

Система должна поддерживать следующий функционал:

- Перенос и анонимизация данных. Система должна обеспечивать возможность переноса взаимосвязанных данных между базами данных, а также их анонимизацию. Правила выбора взаимосвязанных данных и правила анонимизации задаются метаданными.
- Генерация данных. Система должна предоставлять функционал для генерации данных. Правила генерации задаются метаданными.
- Работа с метаданными. Система должна осуществлять проверку корректности метаданных, а также обеспечивать функциональность для загрузки предварительно подготовленных метаданных или их интеграции на этапе ввода данных.
- Гибкость в выборе базы данных. Необходимо внедрить функциональность, позволяющую пользователю выбрать базу данных, указав как уникальный идентификатор базы, так и строку подключения.

1.4.2 Требования к свойствам архитектуры

Архитектура системы должна соответствовать следующим критериям:

- Безопасность. Архитектура должна обеспечивать защиту от несанкционированного доступа к базе данных, а также обеспечивать защиту конфиденциальной информации.
- Надёжность. Система должна функционировать без сбоев в течение продолжительных периодов времени. Это предполагает наличие мер по обеспечению отказоустойчивости и возможности восстановления после сбоев.
- Высокая производительность. Архитектура должна обеспечивать эффективную обработку больших объёмов данных.
- Асинхронная обработка запросов. Поскольку перенос или генерация данных может занимать значительное время, система должна поддерживать выполнение запросов в асинхронном режиме. Это

означает, что клиентские приложения не должны блокироваться в ожидании завершения операции, но должны иметь возможность проверять статус выполнения запроса по мере необходимости.

2 Проектирование системы, разработка языка и алгоритма

2.1 Архитектура системы

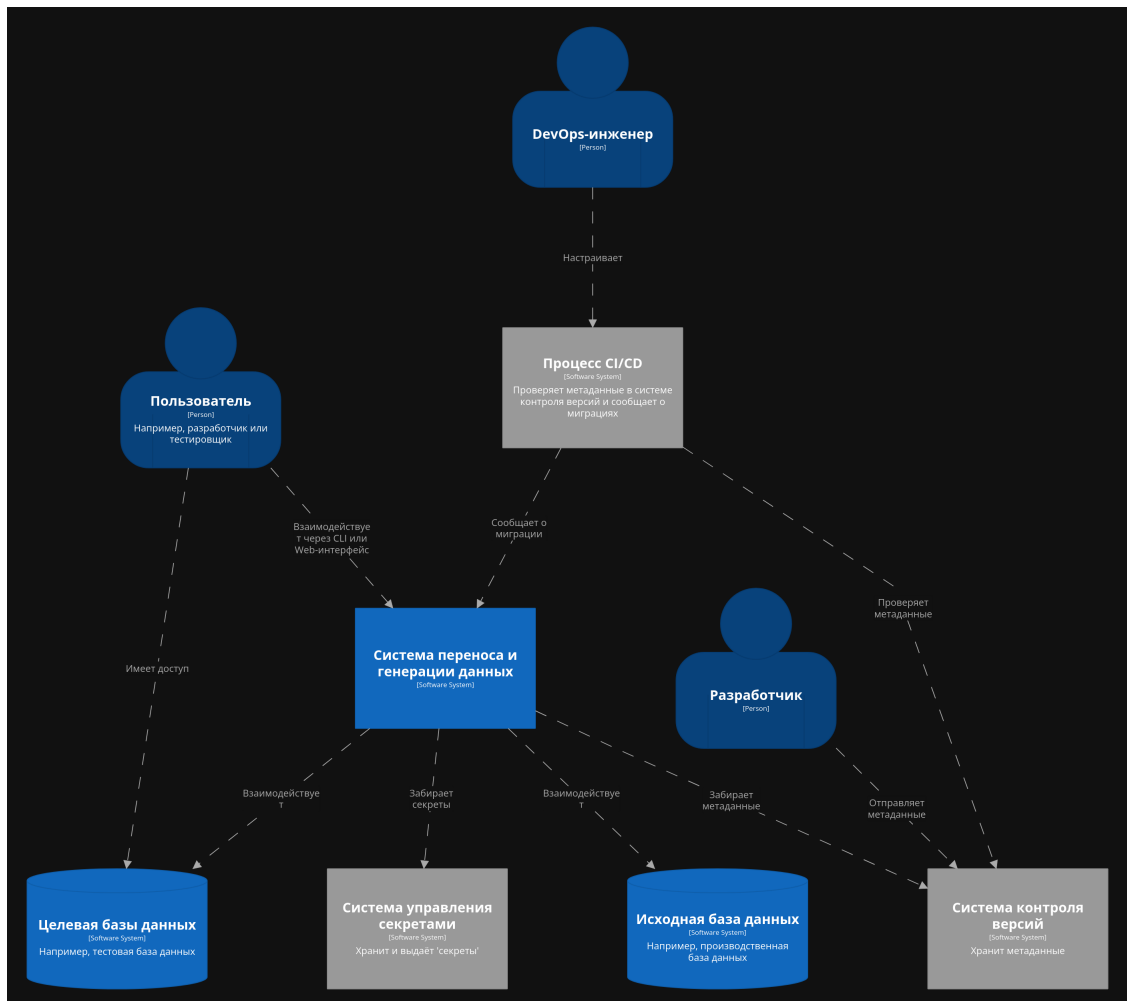


Рисунок 1 – System Context

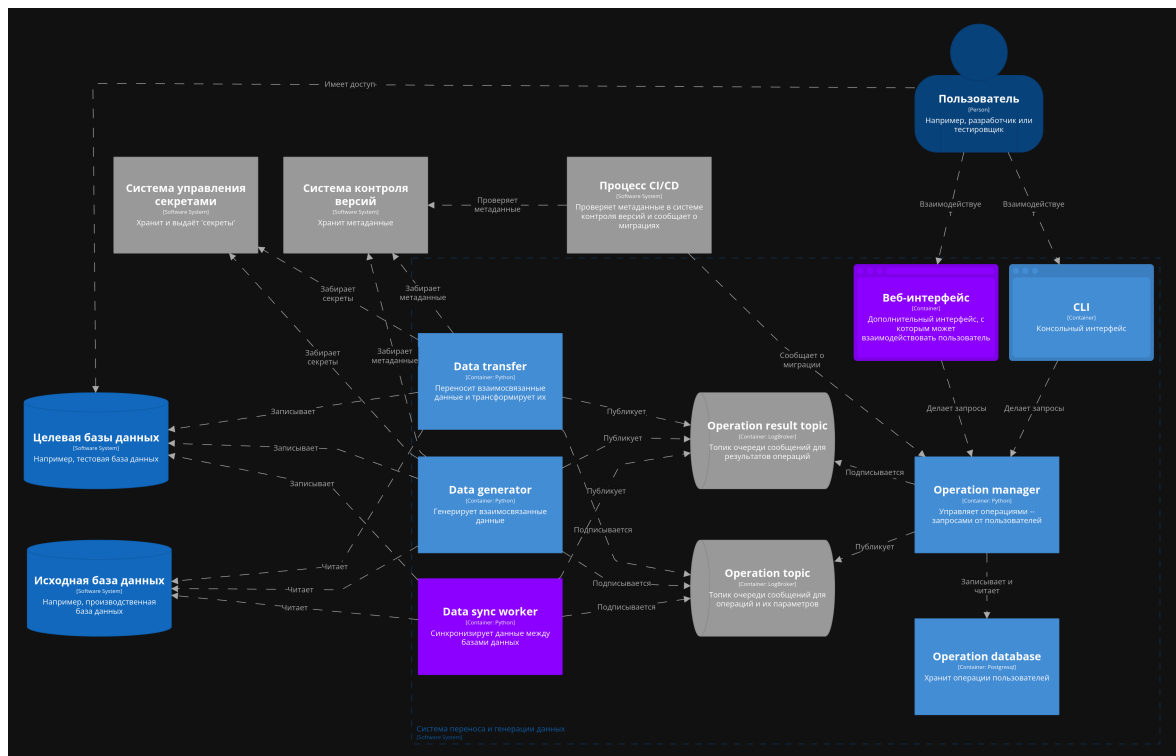


Рисунок 2 – Containers

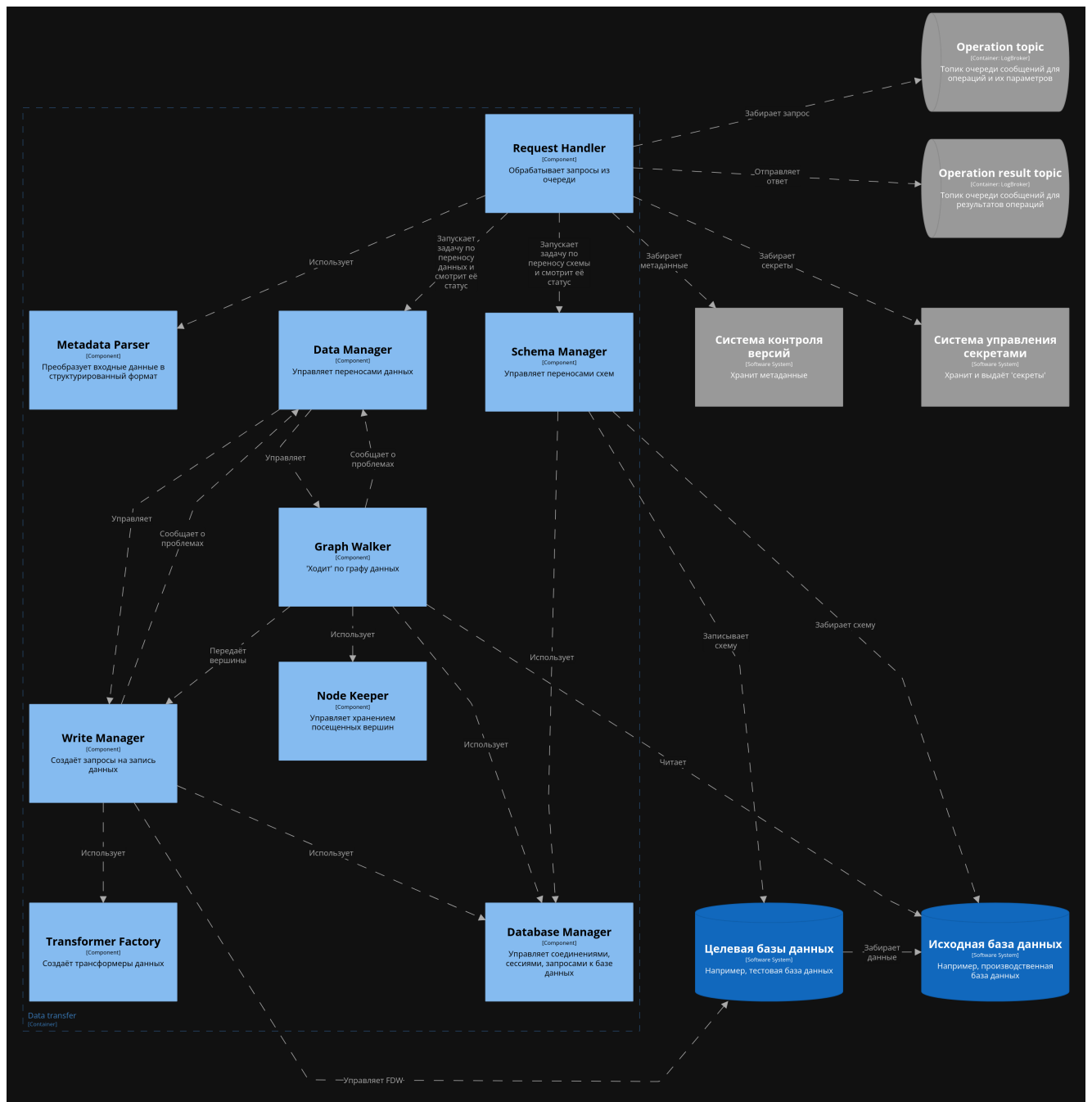


Рисунок 3 – Data Transfer Components

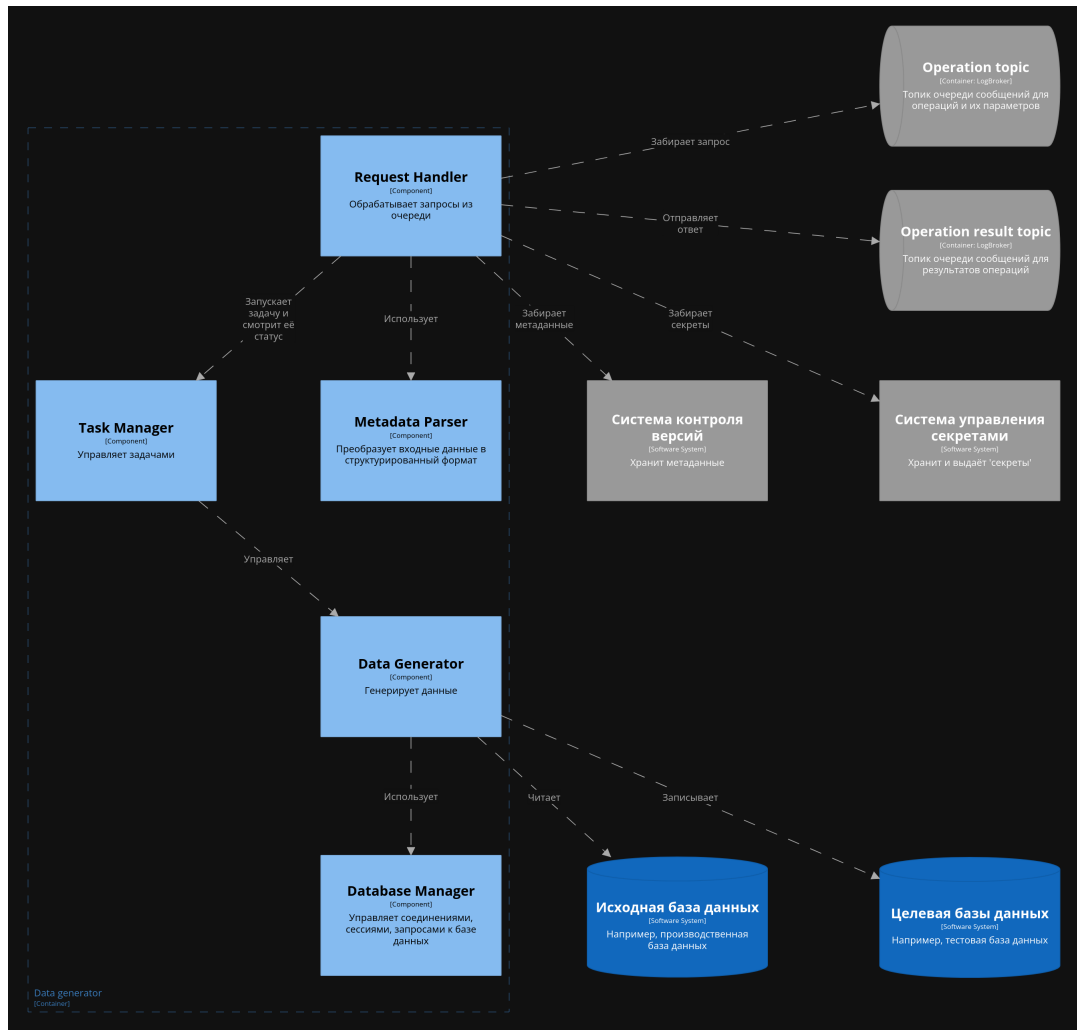


Рисунок 4 – Data Generator Components

2.2 Метаграфы

TBD: расписать эту секцию получше

$MG = \langle V, MV, E, ME \rangle$ – метаграф, где V – множество вершин, MV – множество метавершин, E – множество рёбер, ME – множество метарёбер.

$v_i = \{atr_k\}$, $v_i \in V$, где v_i – вершина, atr_k – атрибут.

$mv_i = \langle \{v_j\}, \{atr_k\} \rangle$, $mv_i \in MV$, $v_j \in V$, где mv_i – метавершина, v_j – вершина, atr_k – атрибут.

$e_i = \langle v_s, v_e \rangle$, $e_i \in E$, $v_s, v_e \in V$, где e_i – ребро, v_s – исходная вершина, v_e – конечная вершина.

$me_i = \langle mv_s, mv_e, \{atr_k\} \rangle$, $me_i \in ME$, $mv_s, mv_e \in MV$, где me_i – метаребро, mv_s – исходная метавершина, mv_e – конечная метавершина, atr_k – атрибут.

Ограничение: $\forall mv_i, mv_j \in V, i \neq j \Rightarrow \{v_x\}_i \cap \{v_y\}_j = \emptyset$

2.3 Алгоритм переноса данных

TBD: расписать эту секцию получше

Для реляционной базы данных метаграф это:

- вершины обозначают данные; атрибуты – полезная нагрузка;
- метавершины обозначают таблицы, содержащие данные; атрибуты – структура таблицы;
- рёбра показывают, как связаны данные;
- метарёбра показывают, как связаны таблицы (по Foreign Key); атрибуты – названия полей, по которым связаны таблицы.

```
1 // TBD: написать алгоритм
2 BFSforMetagraph(graph, startVertex)
3   // Инициализация
4   queue ← {}
5   visited ← {}
6
```

Рисунок 5 – Алгоритм

ЗАКЛЮЧЕНИЕ

TBD: написать заключение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исследование рынка онлайн-образования. — URL: <https://main.talenttech.ru/research/issledovanie-rynka-onlajn-obrazovaniya/> (дата обращения 01.03.2025).
2. PostgreSQL: The world's most advanced open source database. — URL: <https://www.postgresql.org/> (дата обращения 02.01.2025).
3. PostgreSQL: Documentation: 17: pg_dump. — URL: <https://www.postgresql.org/docs/current/app-pgdump.html> (дата обращения 02.01.2025).
4. PostgreSQL: pg_anonymize, a new extension for simple and transparent data anonymization. — URL: https://www.postgresql.org/about/news/pg_anonymize-a-new-extension-for-simple-and-transparent-data-anonymization-2606/ (дата обращения 02.01.2025).
5. Блэк Р. Ключевые процессы тестирования. — Лори, 2025. — ISBN 978-5-85582-392-9.

ПРИЛОЖЕНИЕ А

Исходный код

TBD: ссылка на исходный код