УТВЕРЖДАЮ

Директор ООО «Центр-инвест ИТ»

_____ Дорошкевич Е. В.

«19» мая 2026 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «УТКА»

Руководство администратора/пользователя

Описание функциональных характеристик и информация, необходимая для установки и эксплуатации ПО «УТКА»

Оглавление

1. Термины

ПО «УТКА» — группа микросервисов, который связаны между собой с помощью брокера сообщений RabbitMQ. Каждый из микросервисов выполняет ту или иную функцию в зависимости от собственного назначения.

Маршрут — цепочка из очередей и микросервисов, через которые проходит сообщение от источника к получателю.

Camunda / Draw.io — инструменты для проектирования и визуализации маршрутов.

utka-gateway — шлюз для маршрутизации сообщений в системе УТКА, получает входящие сообщения через RabbitMQ и направляет их по маршрутам.

utka-bridge — мост для передачи сообщений между разными экземплярами RabbitMQ, обеспечивает межсетевое взаимодействие между VLAN.

utka-coder — сервис для кодирования и декодирования данных, поддерживающий механизмы Base64, URL и преобразование кодировок символов.

utka-git — модуль УТКА, с помощью которого можно клонировать или обновить репозиторий из GitLab. Сервис, используя токен в GitLab, получает нужный репозиторий и передает файлы по RabbitMQ.

utka-git-lib — библиотека, используемая для упрощения работы с utka-git. Она помогает свести к минимуму ручное взаимодействие с сервисом, а также упрощает первичную выгрузку шаблонов из репозитория. Библиотека позволяет обновлять файлы из репозитория, при этом, в случае проблем в работе сервиса, есть возможность безопасно использовать уже существующие шаблоны.

utka-query — сервис для выполнения SQL-запросов в базах данных. Поддерживает команды SELECT, INSERT, UPDATE, DELETE. Возвращает результат в очередь RabbitMQ в формате JSON, XML или CSV.

utka-mapper — модуль УТКА, предназначенный для преобразования данных из одного формата в другой. Запросы поступают через RabbitMQ, ответы возвращаются в очередь, указанную в «reply_to». Доступные форматы: XML, JSON, YAML. Формат ответа задаётся через header «out_type».

utka-smtp — модуль УТКА, предназначенный для отправки электронных писем (e-mail) через SMTP-сервер. Поддерживает отправку вложений (файлов) и отображение письма в HTML.

utka-xslt — сервис для преобразования XML-документов с использованием XSLT-шаблонов версии 1.0. Получает запросы через очередь RabbitMQ, выполняет преобразование и возвращает результат.

BPMN — стандарт для моделирования бизнес-процессов, используемый для описания маршрутов в системе УТКА.

RabbitMQ — брокер сообщений, используемый для взаимодействия между микросервисами УТКА по протоколу AMQP.

Заголовки (headers) — метаданные сообщения в RabbitMQ.

Репозиторий (в контексте utka-git) — источник данных в GitLab, идентифицируемый по ключу (заголовок repository), содержащий необходимые файлы (шаблоны, конфигурации).

2. Описание ПО «УТКА»

ПО «УТКА» (Универсальный Транспортный Корпоративный Адаптер) представляет собой группу микросервисов, взаимодействующих через RabbitMQ. Основные функции:

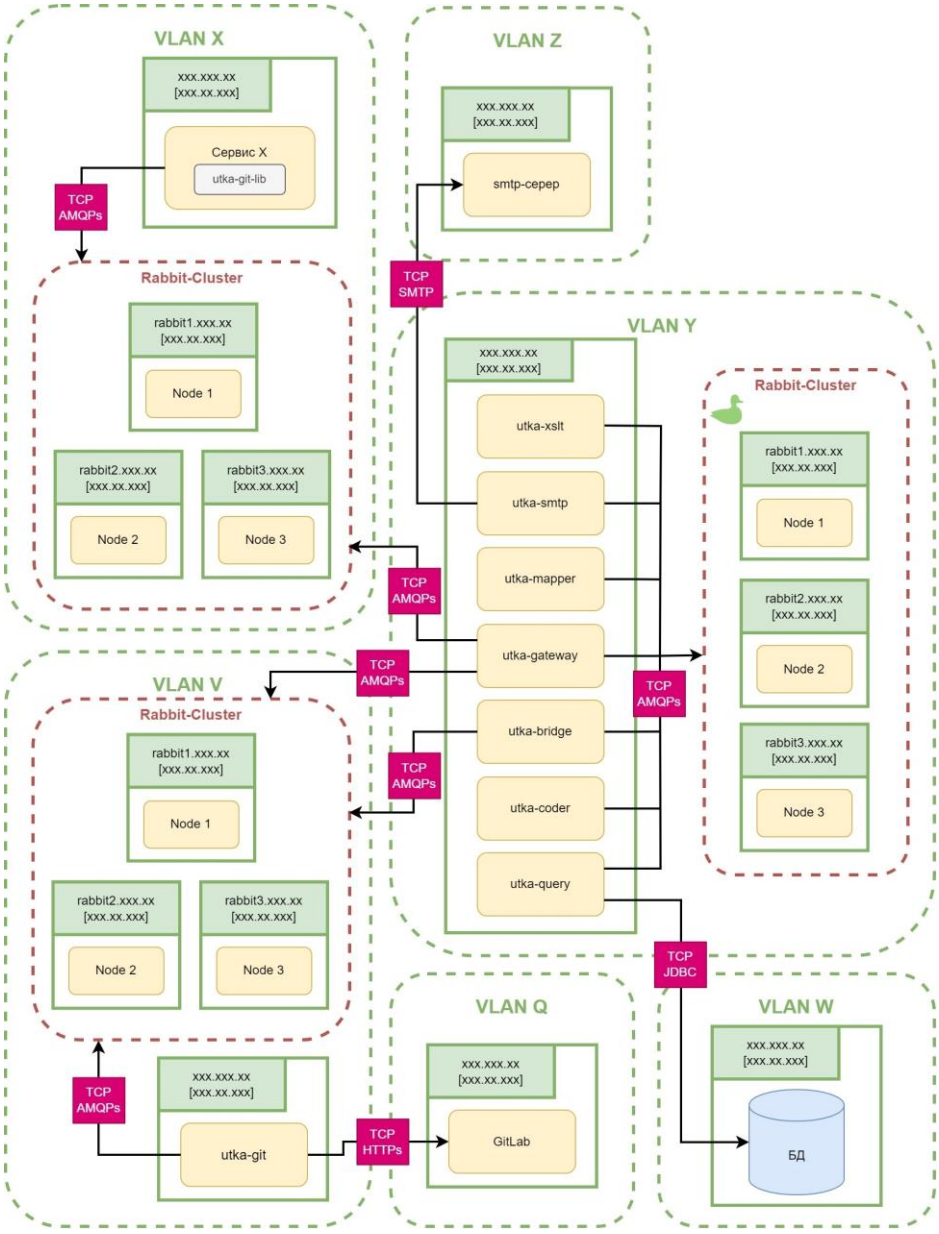
- Интеграция банковских систем (мобильный банк, система управления банковскими счетами, система учёта банковских операций).
- Обмен данными с внешними системами (государственные учреждения, банковские партнёры).
- Промежуточное преобразование форматов данных (JSON ↔ XML, XSLT-преобразования, преобразование XML/JSON/YAML).
- Кодирование и декодирование данных (Base64, URL, Charset).
- Получение и актуализация файлов (шаблонов, конфигураций) из GitLab-репозиториев.
- Выполнение SQL-запросов к базам данных с возвратом результата в форматах JSON, XML или CSV.
- Отправка электронных писем через SMTP-сервер с поддержкой вложений, HTML-форматирования и подстановки переменных.

ПО «**УТКА»** состоит из следующих элементов:

- utka-gateway;
- utka-bridge;
- utka-coder;
- utka-git;

- utka-git-lib;
- utka-query;
- utka-xslt;
- utka-mapper;
- utka-smtp;
- RabbitMQ-кластер;
- База данных (БД);
- SMPT-сервер;
- GitLab;
- OAuth2-сервер.

Общая диаграмма развёртывания системы УТКА:



Технологический стек:

- Язык программирования: Java 17
- Фреймворк: Spring Boot 3
- Брокер сообщений: RabbitMQ
- Протокол: AMQP
- Форматы данных: JSON, XML, YAML
- Формат маршрутов: BPMN / Draw.io

3. Программные и аппаратные требования к системе

№ n/n	Параметр	Минимально допустимые значения	Рекомендуемые значения
1	Операционные системы	Linux / Windows (поддержка Java 17)	Linux
2	Java	JDK / JRE 17	JDK 17
3	RabbitMQ	3.8+	3.12+ (кластер)
4	ОЗУ	4 ГБ (на все сервисы)	8 ГБ и более
5	Дисковое пространство	10 гб	20 ГБ+ (с учётом репозиториев GitLab)

4. Руководство администратора/пользователя

4.1. Установка

Для запуска ПО «УТКА» требуются JAR-файлы всех микросервисов и файлы конфигурации application.yml.

Запуск микросервисов осуществляется в командной строке с использованием команд:

java -jar utka-gateway.jar --spring.cloud.bootstrap.location=application-gateway.yml

java -jar utka-bridge.jar --spring.cloud.bootstrap.location=application-bridge.yml

java -jar utka-coder.jar --spring.cloud.bootstrap.location=application-coder.yml

java -jar utka-git.jar --spring.cloud.bootstrap.location=application-git.yml

java -jar utka-query.jar --spring.cloud.bootstrap.location=application-query.yml

java -jar utka-xslt.jar --spring.cloud.bootstrap.location=application-xslt.yml

java -jar utka-mapper.jar --spring.cloud.bootstrap.location=application-mapper.yml

java -jar utka-smtp.jar --spring.cloud.bootstrap.location=application-smtp.yml

После запуска микросервисов происходит автоматическое подключение к RabbitMQ-кластеру и создание необходимых очередей согласно конфигурационным файлам.

Для работы с utka-git-lib необходимо добавить зависимость в проект сервиса-потребителя:

```gradle
dependencies {
    implementation("ru.cib:utka-git-lib:1.0.0")
}
```

4.2. Основные компоненты

Логи работы микросервисов выводятся в стандартный вывод (stdout) и могут быть настроены в соответствии с политикой логирования Spring Boot.

Конфигурация системы задается в файлах application.yml. Основные настройки включают:

- Параметры подключения к RabbitMQ (адреса, виртуальные хосты, учетные данные);
- Настройки маршрутизации для utka-gateway;
- Конфигурация соединений для utka-bridge;
- Конфигурация очередей и параметров кодирования для utka-coder;
- Конфигурация соответствия ключей repository параметрам репозиториев
GitLab (url, ветка, токен) для utka-git;
- Конфигурация библиотеки utka-git-lib (путь к директории хранения, идентификатор репозитория, очередь для общения с utka-git);

- Конфигурация utka-xslt (в файле application-xslt.yml указывается путь к директории с XSL-шаблонами);
- Конфигурация utka-mapper и utka-smtp.
- Параметры авторизации (при использовании OAuth2);

Интерфейс управления системой представлен через RabbitMQ Management UI, где отображается состояние очередей, статистика сообщений и параметры соединений.

4.3. Работа с utka-gateway

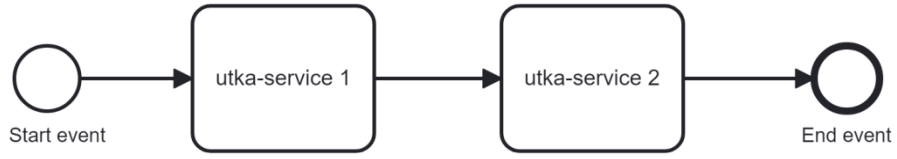
Сервис выполняет функцию шлюза для utka-сервисов. Сервис получает входящие сообщения посредством RabbitMQ, накладывает на них маршрутизацию, что позволяет поэтапно вызывать разные utka-сервисы. Маршруты в системе «УТКА» описываются с использованием BPMN-схем, создаваемых в Camunda Modeler или Draw.io. Любой маршрут должен начинаться с элемента Start Event, в свойствах которого обязательно указываются:

- inbound — имя очереди, которую слушает utka-gateway для этого маршрута;
- conn-id — идентификатор соединения с RabbitMQ из файла конфигурации;
- Опционально: oauth=true для включения проверки авторизации по OAuth2-токену.

Каждая ветвь маршрута должна завершаться элементом End Event, в свойствах которого опционально может быть указан параметр outbound — имя конечной очереди для ответа. В одном файле (BPMN или Draw.io) может находиться множество независимых схем маршрутизации.

Динамическое обновление маршрутов: При добавлении новых или изменении существующих файлов схем в директорию input сервиса utka-gateway происходит их автоматическая обработка, валидация и активация. Для каждого маршрута формируется уникальное имя (имя_файла=>маршрут), проверяется отсутствие конфликтов по очереди inbound и conn-id, после чего маршрут загружается в кэш. Старые версии схем перемещаются в директорию archive.

Пример BPMN схемы:



Пример схемы Draw.io:

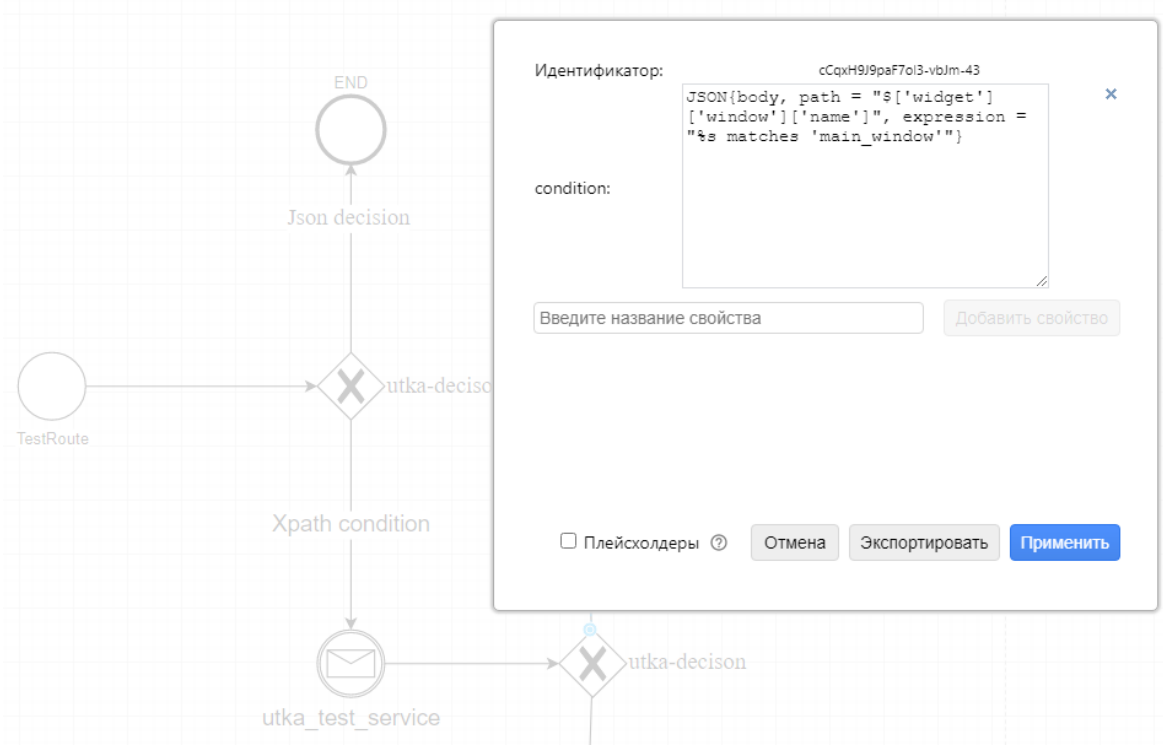


Диаграмма последовательности работы utka-gateway:

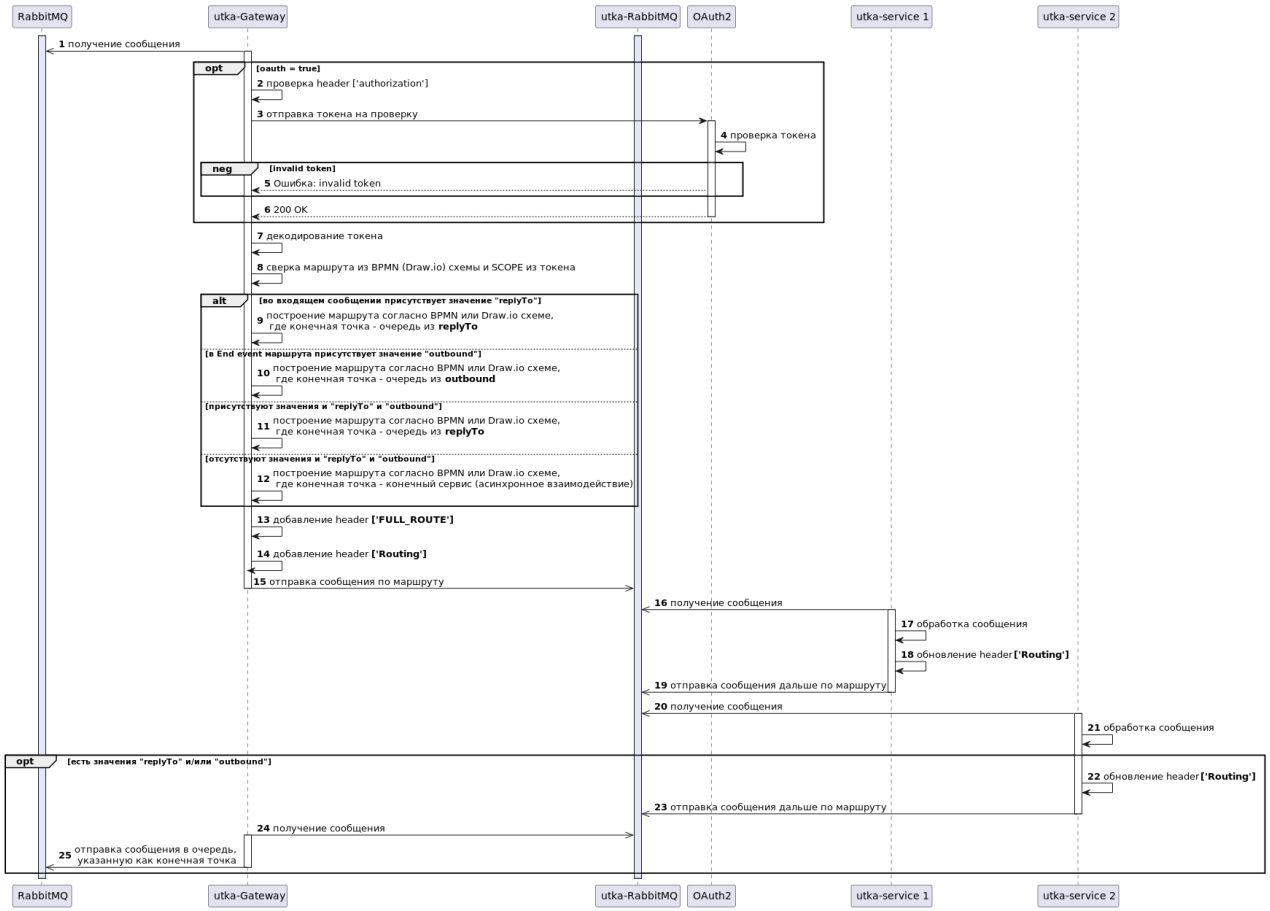
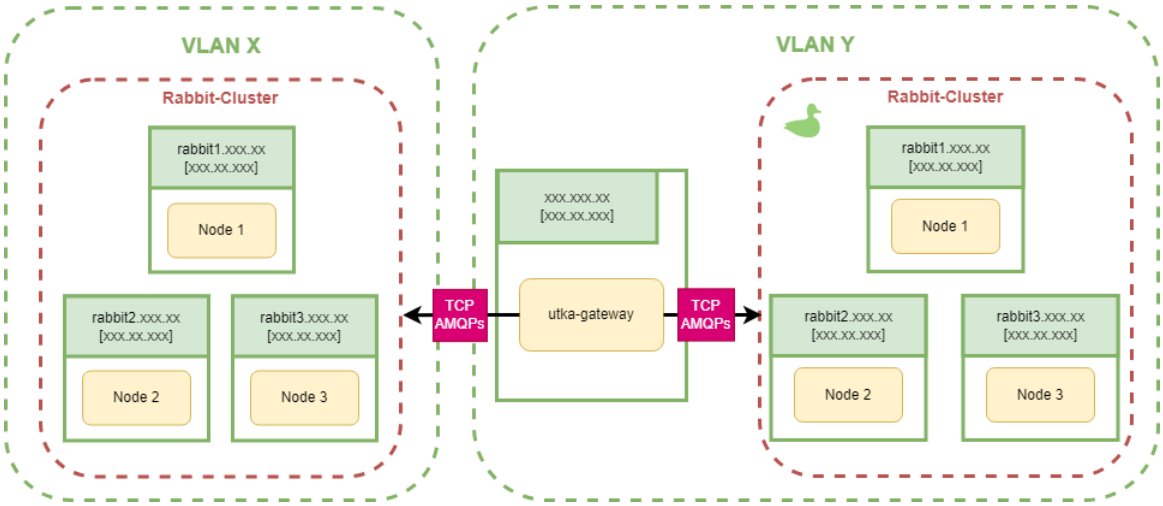


Диаграмма развёртывания utka-gateway:



4.4. Работа с utka-bridge

utka-bridge выполняет функцию передачи сообщений между разными экземплярами RabbitMQ, находящимися в различных VLAN. Конфигурация моста включает определение:

- Заголовка для идентификации соединения (параметр header);
- Подключений к RabbitMQ-кластерам в разных сетевых сегментах;
- Слушателей (consumers) для обработки очередей-мостов;

Сообщения передаются через utka-bridge с использованием заголовков:

- Target (обязательный) — идентификатор соединения с целевым RabbitMQ;
- Routing — определяет очередь назначения в целевом RabbitMQ;
- Sync — определяют тип взаимодействия (синхронный или асинхронный).
- Address —определяет в какую очередь будет отправлено сообщение

Логика определения режима работы utka-bridge:

Если заголовок address не указан — сообщение отправляется асинхронно.

Если address указан (в формате exchange/routing-key):

- при sync = true — синхронный режим (sendAndReceive, ответ возвращается по указанному address);
- при sync = false или sync не указан (значение по умолчанию false) — асинхронный режим.

Пример конфигурации utka-bridge и utka-gateway:

```yaml
# utka-gateway.yml
utka-gateway:
  clients:
    client1:
      addresses: <IP1>:<PORT1>,<IP2>:<PORT2>,<IP3>:<PORT3>
      virtual-host: test1
      username: test
      password: test
      useSsl: true
  channels:
    channel1:
      inbound: queue.in
      route:
        - queue1
        - utka-bridge-queue?Target=conn1  # передача в bridge
        - queue2
```

```yaml
# utka-bridge.yml
utka-bridge:
  header: 'Target'
  connections:
    conn1:  # идентификатор из Target
      addresses: - <IP1>:<PORT1>,<IP2>:<PORT2>,<IP3>:<PORT3>
      vhost: TB
      username: admin
      password: admin
  listeners:
```

```
    consumer1:
      addresses: - <IP1>:<PORT1>,<IP2>:<PORT2>,<IP3>:<PORT3>
      vhost: rabbit_test
      username: rabbit_user
      password: rabbit_user
      queue: utka-bridge-queue
```

Диаграмма последовательности работы utka-bridge:

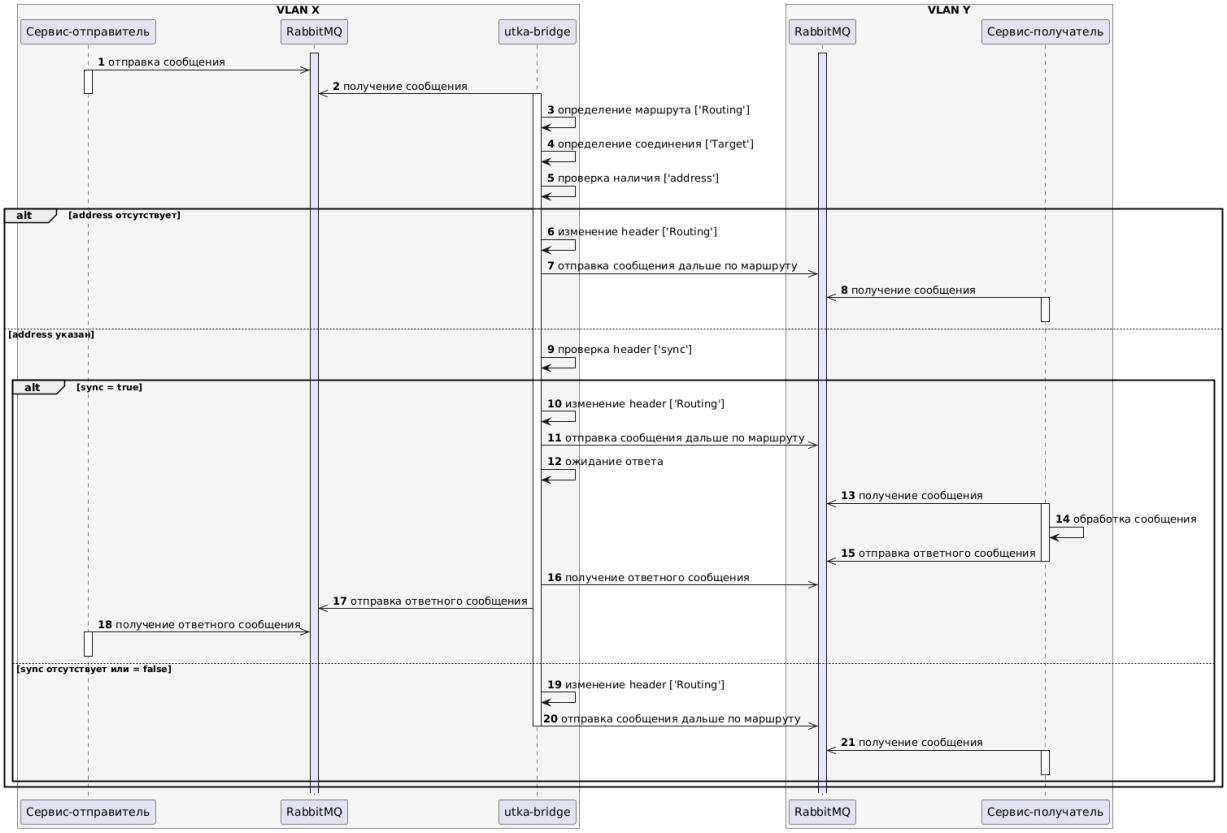
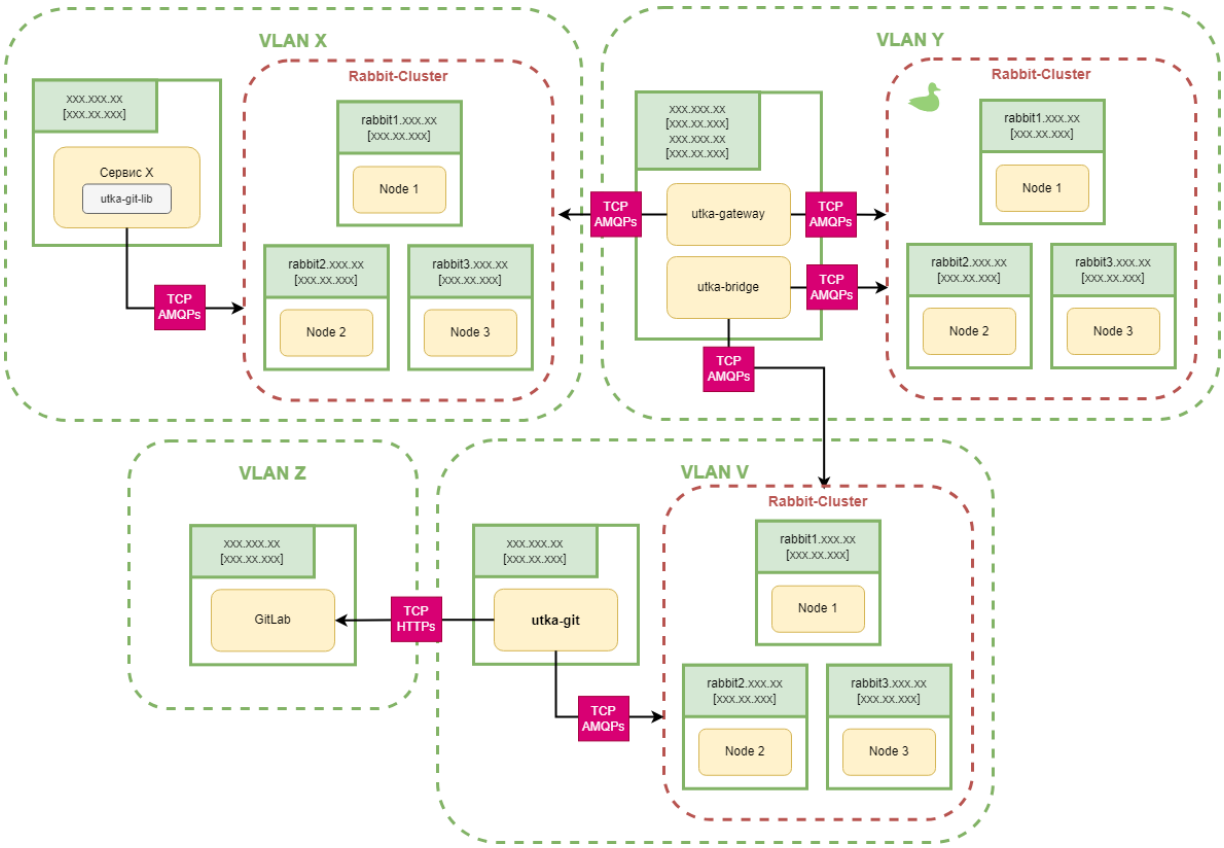


Диаграмма развёртывания работы utka-bridge:



4.5. Работа с utka-coder

Сервис слушает отдельные входные очереди для кодирования и декодирования для каждого механизма (Base64, URL, Charset). Имена очередей задаются в конфигурации.

При получении сообщения сервис определяет механизм по очереди, а также извлекает заголовки:

- type (для Base64 и URL) — формат ответа: JSON, XML, RAW (по умолчанию RAW);
- charset (для URL и Charset) — кодировка текста;
- from_charset / to_charset (для Charset) — исходная и целевая кодировки.

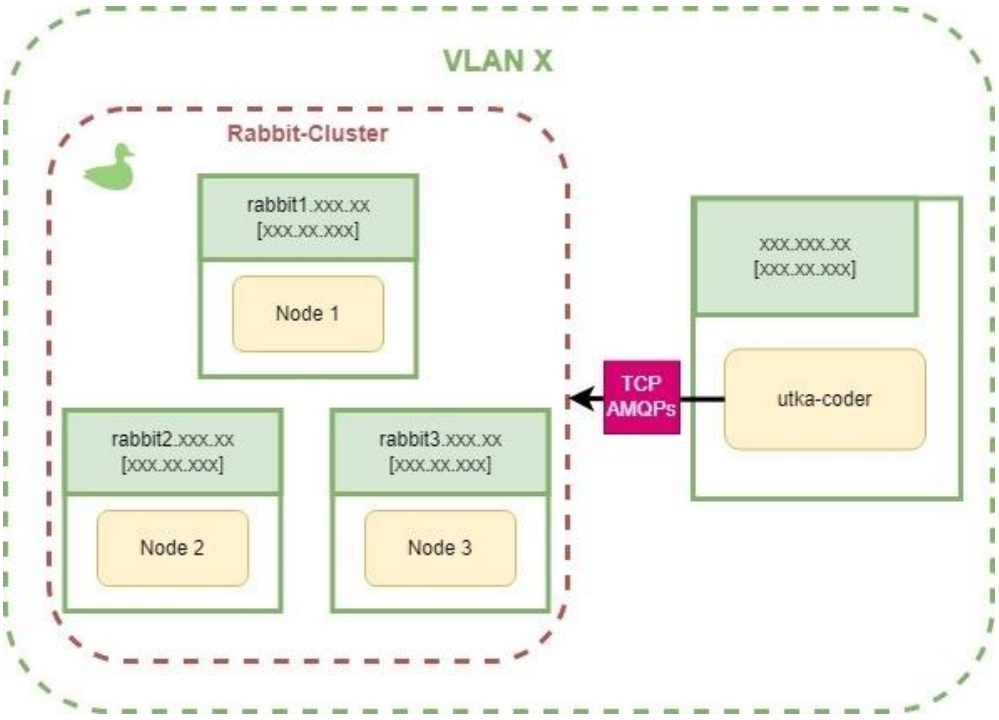
В зависимости от механизма выполняет:

- Base64 — декодирует или кодирует данные из/в Base64.
- URL — декодирует или кодирует URL-строку с учётом указанной кодировки.
- Charset — преобразует кодировку текста из from_charset в to_charset.

Формирует ответ в формате, заданном заголовком type (для Base64 и URL; для Charset всегда RAW).

Отправляет результат в очередь, указанную в заголовке reply_to.

Диаграмма развертывания utka-coder:



4.6. Работа с utka-git

Сервис utka-git предназначен для клонирования и обновления репозиториев из GitLab.

В конфигурационном файле utka-git настраивается соответствие между ключами (заголовок repository) и параметрами репозиториев:

- url до репозитория;
- ветка;
- токен для доступа к GitLab.

Алгоритм работы:

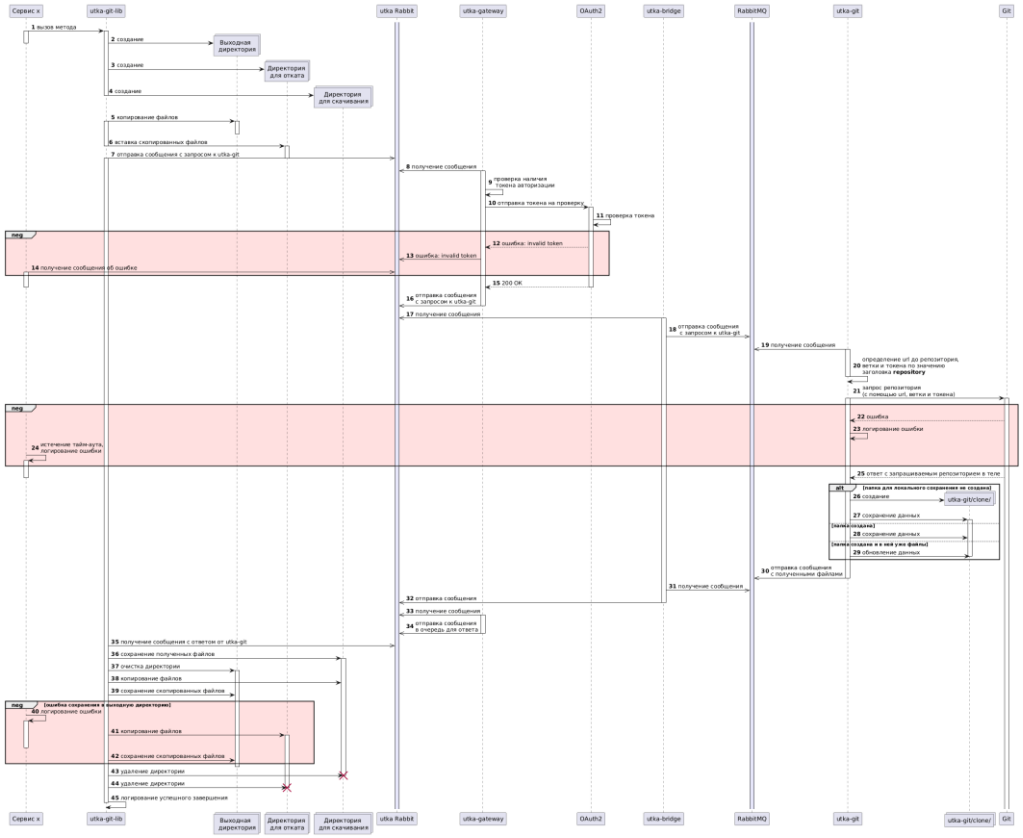
1. Сервис получает на вход сообщение из RabbitMQ с заголовком repository (далее Ключ), в котором сервис-отправитель передает значение репозитория, данные которого хочет получить.
2. По переданному Ключу сервис определяет url до репозитория, ветку и токен (все эти данные берутся из конфига к сервису), по которому будет далее обращаться.
3. С помощью url, токена и ветки Сервис клонирует репозиторий себе локально в папку, указанную в конфиге (если папка не создана, он ее

создаст сам).

4. Если данные были клонированы ранее, то сервис просто обновит их.
5. Полученные файлы Сервис отправляет сервису-отправителю в формате json:

```json
{
  "list": [
    {
      "fileName": "string",
      "bytes": "bytes"
    }
  ]
}
```

Диаграмма последовательности utka-git:



4.7. Работа с библиотекой utka-git-lib

Библиотека utka-git-lib используется для упрощения работы с сервисом utka-git. Она помогает свести к минимуму ручное взаимодействие с сервисом, а также упрощает первичную выгрузку шаблонов из репозитория.

Для подключения в проект необходимо добавить зависимость в build.gradle.kts:

```gradle
dependencies {
    implementation("ru.cib:utka-git-lib:1.0.0")
}
```

и настроить необходимую конфигурацию:

```yaml
utka-git:
  path: '/opt/spring/utka-xslt/templates/' # Путь к директории хранения шаблонов
  repoHeader: 'xslt'                       # Идентификатор репозитория в сервисе utka-git (значение заголовка "repository")
  queue: 'utka-git.queue'                  # Очередь для общения с utka-git
```

Функциональность:

- При первом запуске сервиса вместе с библиотекой, автоматически происходит первичная загрузка файлов из репозитория.
- Дальнейшая выгрузка доступна через вызов метода updateTemplates.

Алгоритм работы метода updateTemplates:

1. После вызова метода, библиотека создает 3 директории:
   - выходная директория (указывается в конфигурации, создается если не существует);
   - временная директория для отката файлов;
   - временная директория для скачивания.
2. После создания директорий происходит копирование файлов из выходной
   директории в директорию для отката.
3. Далее идет обращение к utka-git по rabbit, с указанием необходимого

репозитория в заголовке. Тело запроса остается пустым.

4. После получения ответа из utka-git, происходит запись данных во временную директорию для скачивания:

```json
{
  "list": [
    {
      "fileName": "test1.xml",
      "bytes":
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiID8+Cjx0ZXN0
PjE8L3Rlc3Q+"
    },
    {
      "fileName": "test2.xml",
      "bytes":
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiID8+Cjx0ZXN0
PjI8L3Rlc3Q+"
    }
  ]
}
```

5. После успешной записи файлов, происходит очистка выходной директории, после чего в неё перемещаются файлы из временной директории для скачивания.
6. По завершении выполнения метода, временные директории удаляются.
7. В случае возникновения ошибки на любом из этапов процесса выгрузки
файлов, в выходную директорию будут сохранены файлы из директории
для отката (используются старые версии).

4.8. Работа с utka-query

Сервис utka-query предназначен для выполнения SQL-запросов в базе данных. Название базы данных и текст SQL-запроса предварительно

задаются в конфигурации сервиса.

Сервис слушает очередь RabbitMQ, определённую в его конфигурации. Входящее сообщение должно содержать следующие заголовки (headers):

Заголовок	Тип	Обязательность	Описание
title	String	да	Название запроса из конфигурации (секция app.queries).
datasource	String	да	Название базы данных из конфигурации (секция app.database).
return_type	String	да	Формат ответа: JSON, XML или CSV.
column_separator	String	нет	Формат ответа: JSON, XML или CSV.

Свойства (properties):

- reply_to — очередь, в которую будет отправлен ответ.

Тело сообщения (payload):
Тело запроса передаётся в формате JSON, где ключ — это имя параметра из SQL-запроса (после двоеточия), а значение — подставляемое значение параметра. Пример:
Для конфигурации:

```yaml
queries:
  querie-test: SELECT * FROM ack_reg WHERE sndname = :sndname;
```

Payload будет:

```json
{"sndname":"ABS00"}
```

Диаграмма последовательности работы utka-query:

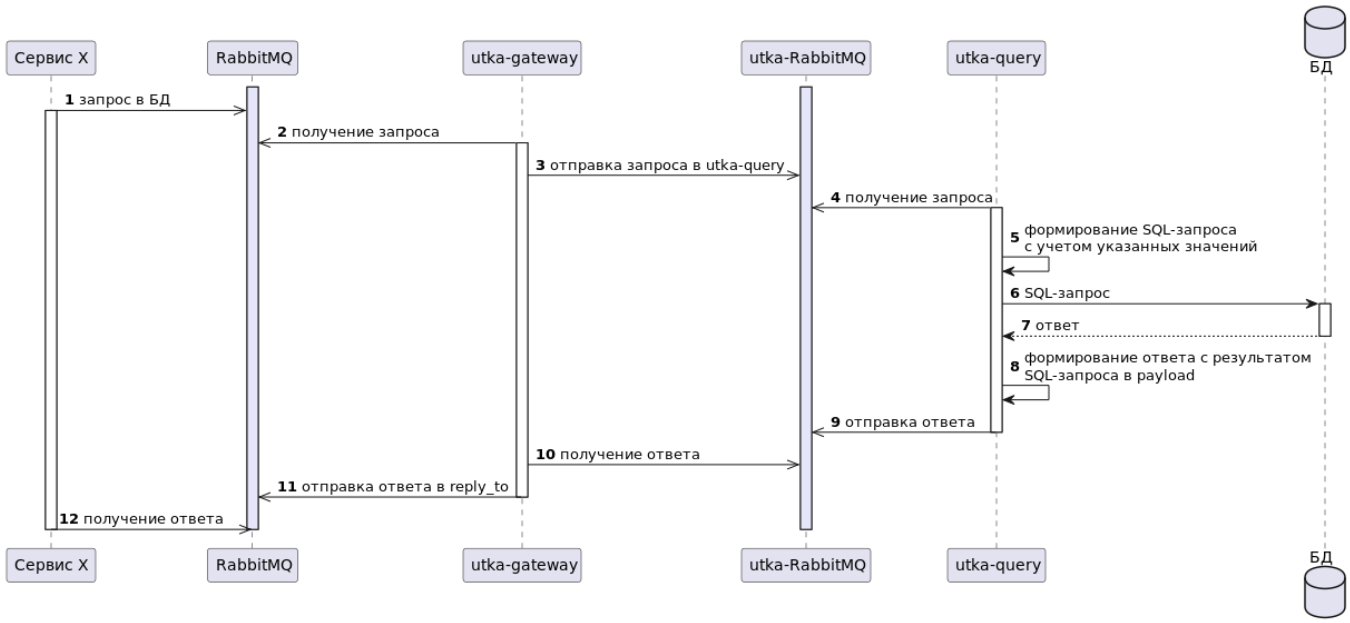
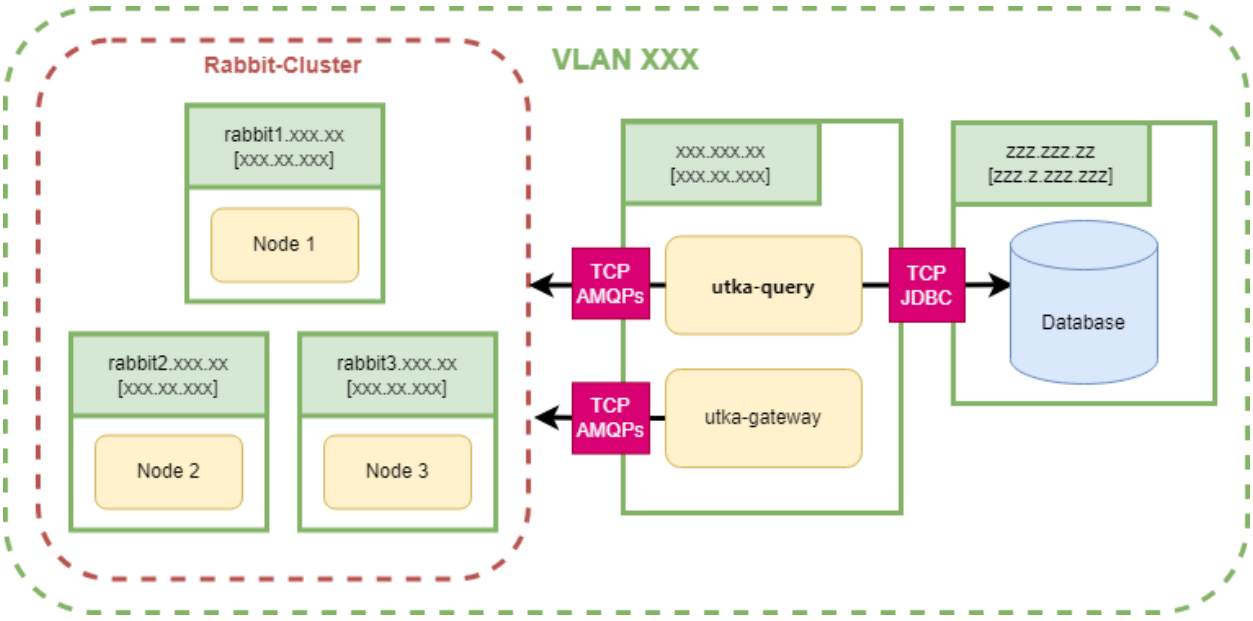


Диаграмма развертывания utka-query:



4.9. Работа с utka-xslt

Сервис utka-xslt предназначен для преобразования XML-документов с помощью XSLT-шаблонов версии 1.0. Запросы поступают через очередь RabbitMQ transformxml.queue. Ответ возвращается в очередь, указанную в заголовке reply_to, либо в следующую очередь маршрута, определённую в схеме.

Формат запроса

Headers

- template (string, обязательный) — имя файла XSL-шаблона (например, invoice.xsl).

Payload

- Исходный XML-документ в виде строки.

Алгоритм обработки

1. При запуске сервис загружает все XSL-шаблоны из директории, указанной в конфигурации, в оперативную память.
2. Получив сообщение, сервис извлекает значение заголовка template.
3. Выполняется поиск шаблона в памяти:
   - если шаблон найден, выполняется преобразование XML;
   - если шаблон не найден в памяти, сервис пытается загрузить его с диска из той же директории, затем выполняет преобразование;
   - если шаблон отсутствует и на диске, формируется сообщение об ошибке.
4. Результат преобразования (XML) отправляется:
   - в очередь, указанную в заголовке reply_to (если он присутствует);
   - либо в следующую очередь, определённую маршрутом в схеме.

Пример сообщения об ошибке (когда шаблон не найден):

xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<error>
  <consumer>
    <name>xslt</name>
    <queue>transform.queue</queue>
    <exchange>transform.exchange</exchange>
    <routing-key>transform.routing-key</routing-key>
  </consumer>
  <method>transformXml</method>
  <message>Exception in transformer</message>
</error>
```

Диаграмма последовательности работы utka-xslt

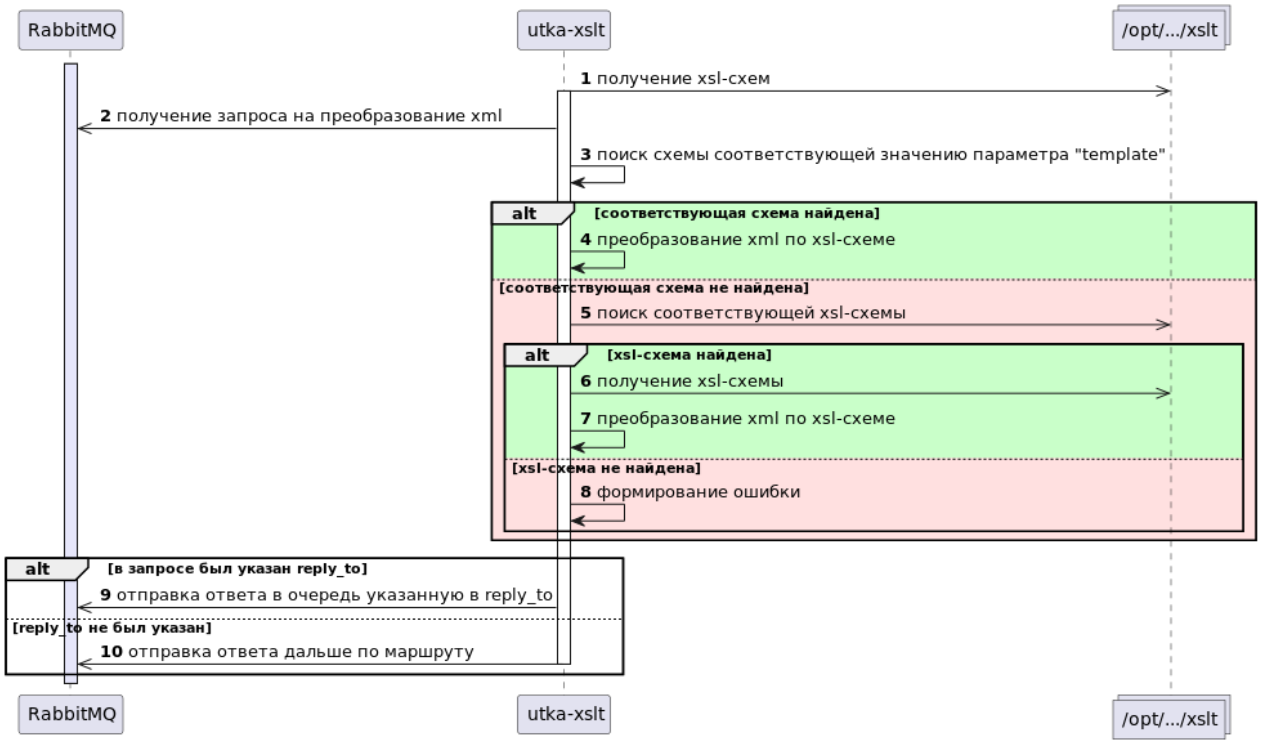
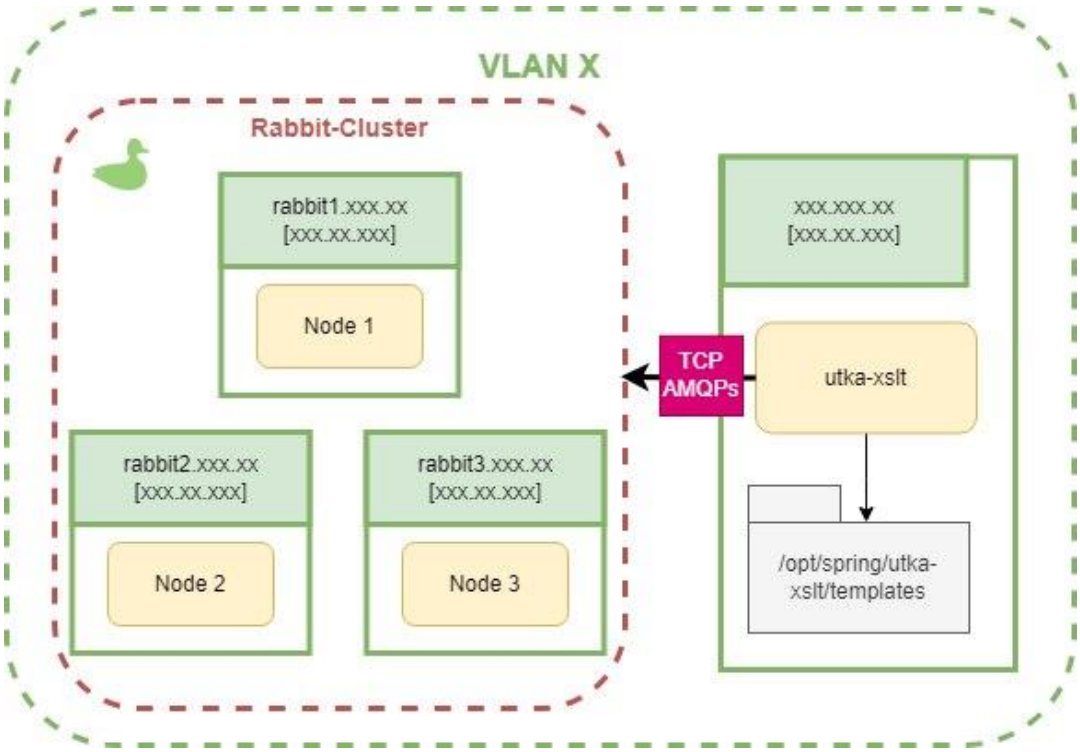


Диаграмма развертывания utka-xslt:



4.10. Работа с utka-mapper

Сервис utka-mapper предназначен для преобразования данных из одного формата в другой (XML, JSON, YAML).

Запросы поступают через RabbitMQ, ответы возвращаются в очередь, указанную в свойстве reply_to.

Формат запроса

Headers:

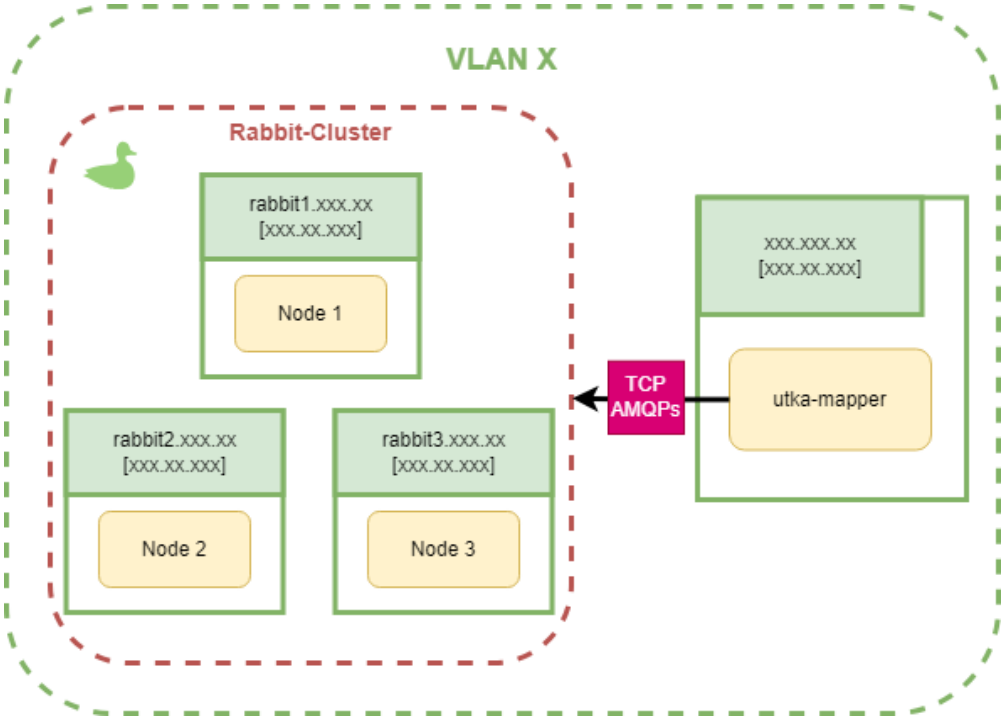
- in_type (обязательно) — входной формат (XML / JSON / YAML)
- out_type (обязательно) — выходной формат (XML / JSON / YAML)

Body: строка данных в формате, указанном в in_type.

Алгоритм работы

1. Сервис слушает очередь RabbitMQ.
2. По header in_type определяет входной формат.
3. Формирует структуру ключ-значение.
4. По header out_type преобразует данные в требуемый формат.
5. Результат отправляет в очередь reply_to.

Диаграмма развертывания utka-mapper:



4.11. Работа с utka-smtp

Формат запроса

Headers (обязательные):

- sender — отправитель (email)
- subject — тема письма (может содержать переменные для подстановки)
- recipients — получатели через запятую

Headers (дополнительные):

- text — шаблон тела письма с переменными (если не указан, используется пустое тело)
- with_files — флаг (true/false), указывает, что вложения берутся из поля files в payload
- filename — если указан, весь payload сохраняется как вложение с этим именем
- is_html — флаг (true/false), определяет, интерпретировать тело письма как HTML

Payload (JSON):

- переменные для подстановки в subject и text (ключ-значение)
- при with_files = true — массив files с полями name (имя файла) и data (byte array)

Алгоритм работы

1. Сервис слушает очередь RabbitMQ.
2. Из заголовков извлекает обязательные параметры: sender, subject, recipients.
3. Из заголовков извлекает дополнительные параметры: text, with_files, filename, is_html.
4. Парсит payload как JSON:
   - извлекает значения переменных для подстановки,
   - если with_files = true — извлекает массив files.
5. Выполняет подстановку переменных:
   - в subject (поддерживается путь {field[0]}),
   - в text (аналогично).
6. Формирует MIME-письмо:
   - устанавливает отправителя, получателей (разбивая recipients по запятой),
   - устанавливает тему (с подставленными переменными),
   - устанавливает тело (plain или HTML в зависимости от is_html).
7. Добавляет вложения:

- если with_files = true — каждое вложение из массива files (имя и byte array),
- если указан filename — весь payload прикрепляется как вложение с именем filename.

8. Отправляет письмо через SMTP-сервер.

Диаграмма развертывания utka-smtp:

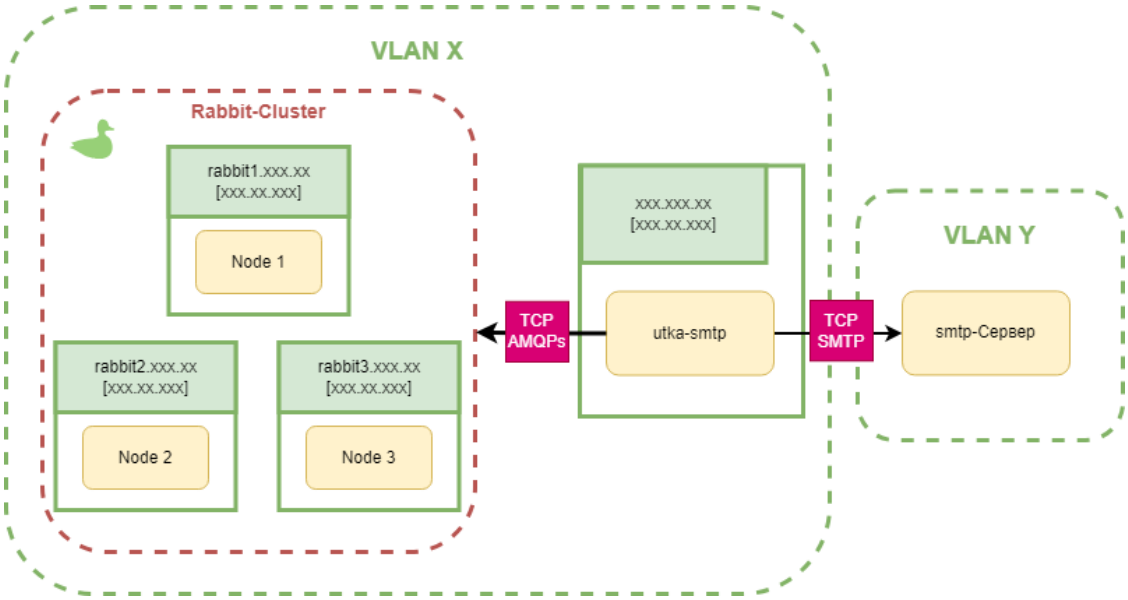
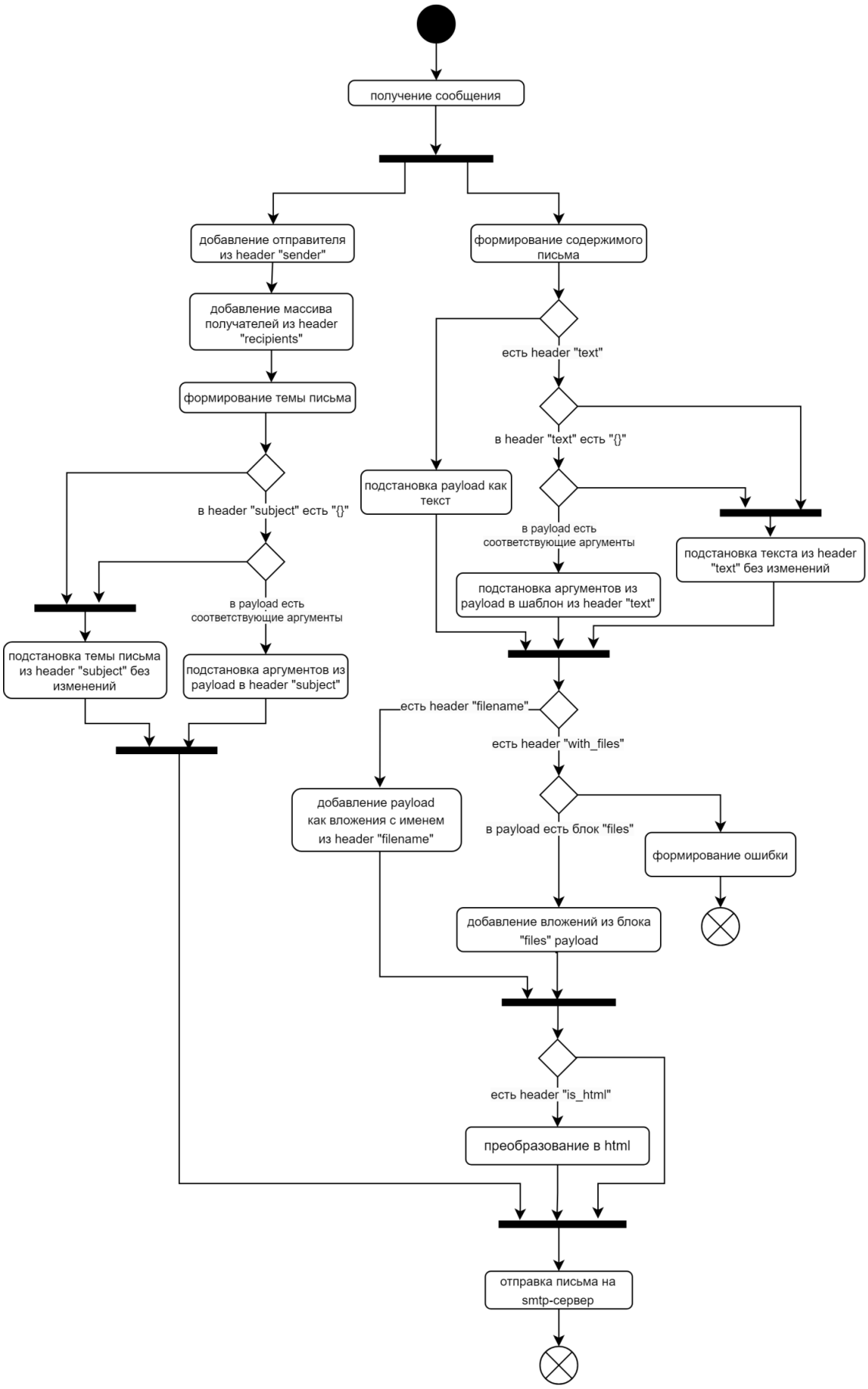


Диаграмма деятельности utka-smtp:



4.12. Мониторинг и диагностика

- RabbitMQ Management UI (очереди, сообщения, DLX).
- Логи микросервисов (stdout).
- Заголовки FULL_ROUTE и Routing (в gateway).
- Для utka-git / utka-git-lib — проверка локальных директорий.
- Для utka-xslt — информация о загрузке шаблонов в логах.

6. Ограничения

Сервис utka-gateway поддерживает получение (прослушивание) сообщений из очередей RabbitMQ, расположенных в различных VLAN. Однако отправка исходящих сообщений возможна только в пределах собственного кластера RabbitMQ системы УТКА (Utka-Rabbit-Cluster). Для передачи данных между разными сетевыми контурами (VLAN) обязательно используется сервис utka-bridge.

При построении маршрутов необходимо учитывать, что все ветви схемы должны завершаться событием End event.

Не рекомендуется добавлять в элементы схемы произвольные свойства (Extension properties), отличные от описанных в документации, так как это может привести к ошибкам в работе программы.

Использование авторизации по OAuth2-токенам требует предварительной настройки OAuth2-сервера и указания его адреса (issuer-uri) в конфигурации utka-gateway. При активации авторизации (oauth=true в схеме) проверка токена и его прав (scope) выполняется для каждого сервиса в маршруте.

7. Авторские права