Оглавление

Глоссарий

Apache Kafka — это распределённая система, предназначенная для обработки потоков данных в режиме реального времени или асинхронно. Apache Kafka называют брокером сообщений, тк она выступает в качестве посредника между интегрирующимися системами.
Kafka Topic — логический канал, очередь для публикации и извлечения сообщений.
Consumer — компонент, подписанный на Kafka-topic, который вычитывает сообщения.
Producer — компонент, публикующий данные в Kafka-topic.
API — интерфейс взаимодействия, предоставляющий доступ к данным и функциям сервиса другим системам.
REST API (Representational State Transfer Application Programming Interface) — это архитектурный стиль для создания веб-сервисов, который использует стандартные HTTP-методы (GET, POST, PUT, DELETE и др.) для взаимодействия между клиентом и сервером.
TLS (Transport Layer Security) — криптографический протокол, обеспечивающий защищённую передачу данных между узлами в сети Интернет.
Dead Letter Queue (DLQ) — специальный топик для неуспешно обработанных сообщений.
Prometheus — это система мониторинга и оповещения о событиях, хранящая данные в виде временных рядов.
Сайт — сайт «За честный бизнес» https://zachestnyibiznes.ru/
SASL (Simple Authentication and Security Layer) — это framework для аутентификации и обеспечения безопасности данных в протоколах, основанных на обмене сообщениями.
SCRAM-SHA-256 (Salted Challenge Response Authentication Mechanism with SHA-256) — это современный и безопасный механизм аутентификации, используемый для проверки подлинности пользователей.
Kubernetes — то платформа с открытым исходным кодом для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями.
AES-256 (Advanced Encryption Standard с ключом длиной 256 бит) — это симметричный алгоритм шифрования, который используется для защиты данных путём их преобразования в зашифрованный формат.
ELK — стек технологий для сбора, обработки, хранения и визуализации логов и данных.

Бизнес-контекст

Сервис «За честный бизнес» (https://zachestnyibiznes.ru/) предоставляет доступ к данным о компаниях, зарегистрированных на территории России, включая информацию о юридических лицах, индивидуальных предпринимателях, финансовых показателях компаний, судебных разбирательствах и других данных, получаемых из официальных источников.

Внешние системы и пользователи (юридические и физические лица) активно используют сервис для:

- Проверки благонадежности контрагентов
- Анализа финансового состояния компаний
- Оценки рисков при заключении договоров
- Получения актуальной информации о судебных делах и другого

Сервис получает и обновляет данные в реальном времени, используя Apache Kafka в качестве брокера сообщений для передачи информации из внешних источников. Данные поступают в формате сообщений в Kafka-топики, после чего они обрабатываются и сохраняются в базе данных сервиса.

Цели и задачи сервиса

Цели сервиса:

1. **Автоматизация получения данных** — получение данных из внешней системы в режиме реального времени через Kafka.
2. **Централизация и хранение данных** — структурирование и сохранение данных в единой базе для последующего использования и анализа.
3. **Упрощение доступа к данным** — предоставление данных через REST API для других сервисов и бизнес-пользователей.
4. **Обеспечение актуальности данных** — мгновенное обновление информации при поступлении новых данных в Kafka.
5. **Масштабируемость** — поддержка работы с высокими объёмами данных и возможностью увеличения пропускной способности при росте нагрузки.

Задачи сервиса:

1. **Автоматизация получения данных**

- Настроить подключение к Kafka с учетом параметров безопасности (TLS, авторизация).
- Настроить подписку на нужные топики с поддержкой нескольких потоков обработки.
- Реализовать асинхронную обработку сообщений для минимизации задержек.
- Настроить контроль смещения (offset) для обработки сообщений в правильном порядке.
- Добавить контроль целостности сообщений.

2. **Централизация и хранение данных**

- Создать структуру базы данных, соответствующую полям данных из Kafka.

- Разработать схему нормализации и индексирования данных.
- Реализовать обработку дубликатов и конфликтов при вставке в базу.
- Добавить возможность ведения истории изменений (например, с помощью временных меток).

3. **Упрощение доступа к данным**

- Разработать REST API для получения данных с возможностью фильтрации и сортировки.
- Добавить поддержку постраничной выдачи (пагинации).
- Настроить авторизацию и аутентификацию через OAuth2 или JWT.
- Реализовать мониторинг запросов (например, через Prometheus).

4. **Обеспечение актуальности данных**

- Реализовать обработку сообщений Kafka в реальном времени.
- Добавить механизм уведомлений об обновлениях (например, через WebSocket).
- Настроить механизм автоматического обновления данных в базе при поступлении новых сообщений.
- Добавить стратегию управления версиями данных (например, с использованием upsert).
- Настроить систему оповещений при задержках в потоке данных.

5. **Масштабируемость**

- Настроить горизонтальное масштабирование сервиса (добавление инстансов при увеличении нагрузки).
- Реализовать возможность работы в режиме отказоустойчивости (high availability).
- Оптимизировать количество потоков для обработки сообщений в зависимости от нагрузки.
- Добавить балансировку нагрузки между инстансами.
- Реализовать механизм graceful shutdown для корректного завершения работы при остановке сервиса.

Область применения

Внешнее использование:
Сайт «За честный бизнес» будет использоваться юридическими и физическими лицами для получения актуальной информации о компаниях, зарегистрированных на территории РФ. Сайт предоставит доступ к сведениям о регистрации, руководителях, финансовых показателях, судебных делах и другим данным, связанным с хозяйственной деятельностью компаний. Данные будут обновляться в режиме реального времени за счёт интеграции с внешними источниками через Kafka.

Внутреннее использование:
Сайт будет использоваться внутренними системами и аналитиками компании для мониторинга деятельности компаний, выявления потенциальных рисков и проведения анализа данных. Сервис будет поддерживать автоматический сбор данных, хранение в базе и предоставление аналитических отчётов через REST API для интеграции с другими внутренними системами.

Границы системы

задача начинается с получения данных из внешних источников через Kafka и заканчивается отображением этих данных на сайте «За честный бизнес» в структурированном и удобном для пользователя формате.
Входные данные поступают из внешних систем через Kafka в виде сообщений в заранее определённом формате. Данные включают информацию о юридических лицах, регистрационные сведения, финансовые показатели, судебные дела и прочие параметры, предусмотренные бизнес-логикой.

Что входит в состав реализуемых функций:

- Подключение к Kafka и получение данных в реальном времени.
- Обработка, валидация и преобразование полученных данных.
- Сохранение обработанных данных в базу данных.
- Обновление существующих данных при получении новой информации.
- Предоставление данных через REST API для отображения на сайте и интеграции с другими системами.
- Ведение логов обработки и ошибок.

Что не входит в состав реализуемых функций:

- Внесение изменений в исходные данные (данные остаются в неизменном виде после получения).
- Ручное редактирование данных через интерфейс сайта.
- Проверка достоверности полученных данных (за корректность данных отвечает внешняя система-источник).
- Разработка алгоритмов анализа и оценки достоверности данных (сервис только отображает предоставленные данные).

Ограничения и допущения

1. **Обработка запросов:**

   Сайт «За честный бизнес» должен быть спроектирован с расчётом на обработку до **5000 одновременных запросов в секунду**, чтобы обеспечить стабильную работу при высоких пиковых нагрузках и одновременных обращениях большого числа пользователей.

2. **Задержка обработки данных:**

   Система должна обеспечивать минимальную задержку при обработке данных. Время обработки каждого сообщения из Kafka, включая валидацию и запись в базу данных, не должно превышать **300 миллисекунд**.

3. **Общая задержка для конечного пользователя:**

   Система должна обеспечивать отклик на пользовательский запрос в течение **не более 1 секунды**, включая обработку на сервере и передачу данных пользователю

через веб-интерфейс. Это обеспечит комфортную работу с системой и высокое качество пользовательского опыта.

4. **Масштабируемость и устойчивость:**

   Система должна быть спроектирована с возможностью горизонтального масштабирования при увеличении объёмов данных и числа запросов. Ожидаемый прирост нагрузки через 1 год составляет **+20%** запросов и далее ежегодное увеличение на **15%**.

5. **Надёжность и отказоустойчивость:**

   В случае отказа отдельных компонентов системы (например, брокеров Kafka или базы данных) система должна сохранять работоспособность, используя резервные копии и механизмы автоматического восстановления. Время простоя в случае аварийной ситуации не должно превышать **5 минут**.

6. **Формат и структура данных:**

   Данные, поступающие из внешних источников через Kafka, имеют фиксированный формат, определённый в техническом задании. Система не должна изменять структуру данных и обязана корректно обрабатывать только данные в установленном формате.

7. **Доступ к данным:**

   Доступ к данным на сайте предоставляется только авторизованным пользователям через веб-интерфейс или REST API. Публичный доступ к API для внешних пользователей не предусмотрен.

8. **Ограничения по объёму данных:**

   Система должна поддерживать хранение информации о не менее чем **10 миллионах юридических лиц** с возможностью дальнейшего расширения.

9. **Корпоративные требования:**

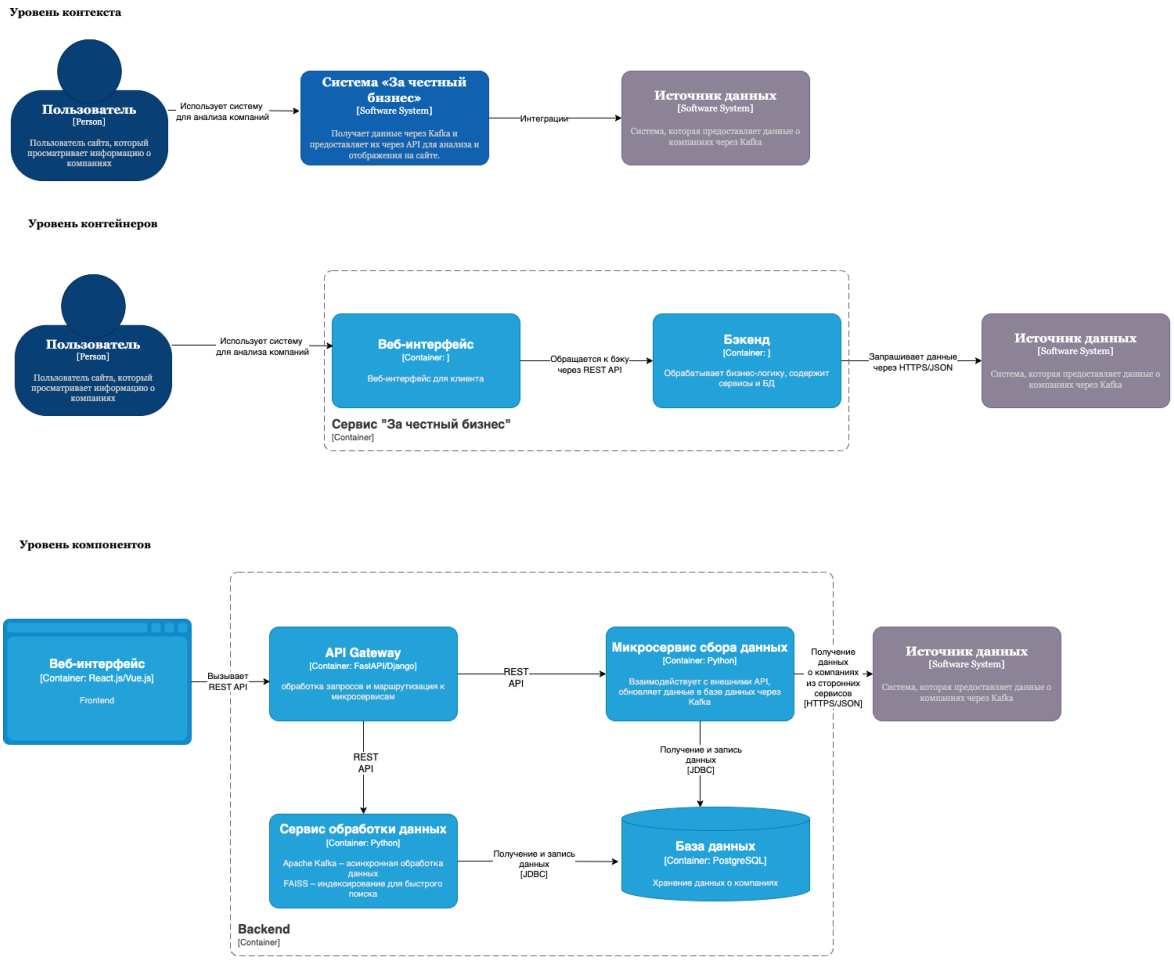
- Система должна соответствовать требованиям информационной безопасности компании.
- Данные должны храниться и обрабатываться в соответствии с требованиями законодательства о защите персональных данных.
- Логи обработки данных должны храниться не менее **180 дней** с возможностью быстрого доступа для анализа инцидентов.

10. **Поддержка многопоточности:**

Для обеспечения высокой пропускной способности система должна использовать многопоточную обработку данных из Kafka и асинхронную обработку пользовательских запросов через REST API.

Схемы и диаграммы

Диаграмма C4



Функциональные требования

1. Подключение к Kafka

1.1. Необходимо подключиться к кластеру Кафка: analytics-kafka-cluster
1.2. Подключение должно осуществляться через безопасный протокол (SSL/TLS).
1.3. Аутентификация через SASL.
1.4. Устойчивость к сбоям при подключении: необходимо настроить механизмы повторных попыток и таймаутов:
- Retries: 5;
- Heartbeat Interval: 3 секунды;
- Session Timeout: 15 секунд.

2. Подписка на топики

2.1. Сервис должен получать данные из топика **Company_data**

2.2. Сервис должен поддерживать настройку смещения для чтения сообщений из топика. Смещение должно быть настраиваемо динамически, при помощи параметров, без изменений кода и пересборки дистрибутива. Варианты настройки:
   2.2.1. earliest: начать с первого сообщения в топике. Это необходимо при первом запуске сервиса, чтобы обработать все доступные сообщения.
   2.2.2. latest: начать с последнего сообщения в топике. Этот вариант подходит, если нужно обрабатывать только новые сообщения, пропуская старые.
   2.2.3. custom: возможность указания конкретного смещения (например, для восстановления после сбоя или при повторной обработке определенных данных).
2.3. Параметр auto.commit = true, auto.commit.interval.ms = 5000 (5 сек)
2.4. Параметры топика **CompanyData**:

Топик	Назначение	Пример структуры данных	Число разделов	Макс. время хранения данных	Макс. размер сообщения	Приложение-продюсер	Приложение-потребитель	Ключ партиционирования
Doubtful_transactions	Подозрительные транзакции	{ "key": null, "headers": null, "value": { "company_id": "123456", "company_name": "Example LLC", "status": "active", "updated_at": "2025-03-12T12:00:00Z" }, "timestamp": "2025-03-12T12:00:10Z" }	3	1 неделя	1MB	Compa_Data_System	Company_Data_Consumer	company_id

3. Обработка сообщений

3.1. Необходимо реализовать параллельную обработку сообщений.
3.2. В случае дублирования событий сервис должен корректно обработать повторные сообщения.
3.3. После успешной обработки сообщения сервис отправляет подтверждение в Kafka, указывая последний обработанный offset.
3.4. Если сервис перезапускается или теряет соединение с Kafka, он начинает читать с последнего подтвержденного сообщения, избегая повторной обработки.

4. Обработка ошибок и восстановление

4.1. Сервис автоматически пытается восстановить соединение в течение заданного времени (30 секунд).
4.2. Если после нескольких попыток (3 попытки) соединение не восстанавливается, сервис генерирует ошибку и уведомляет администратора.

4.3. Обработка сообщений с ошибками (Dead Letter Queue). Если сообщение невозможно обработать из-за ошибок (например, неверный формат данных), оно направляется в Dead Letter Queue (DLQ). Эти сообщения могут быть проверены и исправлены позже.

5. **Сохранение данных в базе данных**:

   Обработанные данные должны сохраняться в базе данных (PostgreSQL) для последующего доступа и анализа. Система должна обеспечивать транзакционную целостность при записи данных и поддерживать эффективное управление ошибками, включая возможность отката в случае необходимости.

   5.1. БД: Company_Data_Updates
   5.2. Схема: Company_Data
   5.3. Таблица: Company

6. **Соответствие полей сообщения полям базы данных** (маппинг на поля БД, source2target)

Маппинг:

Пример сообщения	Поле сообщения	Тип данных в сообщении	Поле в таблице БД	Тип данных в БД	Описание
{"company_id": "123456", "company_name": "Example LLC", "status": "active", "updated_at": "2025-03-12T12:00:00Z"}	company_id	string	company_id	bigint	ID компании
	company_name	string	company_name	varchar	Наименование компании
	status	string	company_status	varchar	Статус компании
	updated_at	string	update_timestamp	timestamp	Время последнего обновления

Нефункциональные требования

1. Производительность

1.1. Сервис должен обрабатывать до **2000 сообщений в секунду** в обычном режиме.
1.2. При увеличении нагрузки система должна быть способна обрабатывать до **10 000 сообщений в секунду** без сбоев и снижения производительности.
1.3. Время обработки одного сообщения не должно превышать **300 мс** в 95% случаев.
1.4. В случае превышения нагрузки система должна масштабироваться горизонтально за счёт увеличения количества потоков обработки.

2. Надёжность

2.1. Сервис должен обеспечивать **доступность 99.9%** в течение года (не более **8 часов** простоя в год).
2.2. В случае сбоя сервис должен автоматически перезапускаться и восстанавливать

соединение с Kafka в течение **30 секунд**.
2.3. В случае отказа узла Kafka сервис должен переключиться на резервный кластер в течение **10 секунд**.
2.4. Все изменения данных в базе данных должны выполняться в транзакции для обеспечения целостности данных.

3. Безопасность

3.1. Для защиты данных, передаваемых по сети, необходимо использовать **TLS 1.2** или выше.
3.2. Аутентификация при подключении к Kafka и к базе данных должна осуществляться через **SASL** с использованием механизмов SCRAM-SHA-256.
3.3. При работе с базой данных следует применять **принцип минимальных привилегий** — каждый пользователь и сервис должен иметь доступ только к тем данным, которые необходимы для выполнения задач.
3.4. Пароли и ключи аутентификации должны храниться в зашифрованном виде с использованием алгоритма **AES-256**.
3.5. В системе должна быть реализована защита от атак типа **SQL-инъекций** и **XSS**.

4. Масштабируемость

4.1. Сервис должен поддерживать **горизонтальное масштабирование** за счёт увеличения количества инстансов сервиса в Kubernetes.
4.2. Система должна автоматически распределять нагрузку между экземплярами при увеличении количества подключений к Kafka.
4.3. Новые инстансы должны автоматически добавляться в пул обработки сообщений при достижении 80% нагрузки.

5. Логирование и обработка ошибок

5.1. Если при маппинге или записи в базу данных возникает ошибка, сервис должен логировать её с уровнем ERROR.
5.2. Все сообщения, которые не удалось обработать, должны сохраняться в **Dead Letter Queue (DLQ)** для последующей обработки вручную.
5.3. Если сервис не может обработать сообщение в течение **3 попыток**, сообщение должно быть направлено в DLQ.
5.4. Должен вестись журнал успешных и неуспешных операций с фиксацией следующих данных:

- Время получения сообщения;
- Время обработки сообщения;
- Статус обработки (успех или ошибка);
- Описание ошибки (в случае неудачи).

6. Логирование событий

6.1. Подключение и отключение от Kafka должно фиксироваться в логе с уровнем INFO.
6.2. Изменение состояния соединения с Kafka (например, потеря соединения) должно фиксироваться с уровнем WARNING.
6.3. Логи должны сохраняться в файловой системе и в централизованной системе логирования (например, **ELK**).
6.4. Логи должны храниться в течение **30 дней**.

7. Мониторинг и метрики

7.1. Сервис должен быть интегрирован с системой мониторинга **Prometheus** для сбора метрик в реальном времени.
7.2. Основные метрики для мониторинга:

- Количество обработанных сообщений за последний час — **не менее 10 000 сообщений**;
- Количество ошибок за последний час — **не более 5 ошибок**;
- Средняя задержка при обработке сообщений — **не более 0.3 секунд**;
- Количество сообщений в DLQ — **не более 100 сообщений**.
  7.3. Система мониторинга должна визуализировать метрики через **Grafana**.
  7.4. При превышении допустимого уровня ошибок (например, более **50 ошибок в минуту**) система должна генерировать оповещение в Slack и отправлять уведомление по email.

8. Удобство использования

8.1. Сервис должен запускаться в виде Docker-контейнера с поддержкой автоматического обновления через **CI/CD**.
8.2. В конфигурации сервиса должны использоваться переменные среды для упрощённой настройки параметров Kafka и базы данных.
8.3. Конфигурация сервиса должна быть описана в файле формата YAML для упрощения развёртывания в Kubernetes.
8.4. Время запуска сервиса должно составлять **не более 10 секунд**.

9. Совместимость

9.1. Сервис должен работать на операционных системах **Linux** и **macOS**.
9.2. Сервис должен поддерживать работу с версиями Kafka **от 2.6.0 и выше**.
9.3. Версия PostgreSQL должна быть **не ниже 13.x**.
9.4. Сервис должен использовать Python **3.11** или выше.