Проектирование архитектуры программных систем

Д5

Проект: Платформа для управления персональными финансами "FinTrack"

Выполнили:

Сидоров Артемий БПИ225

Шмараев Артём БПИ225

(БПИ228)

ПИ ФКН НИУ ВШЭ · 2024/2025 учебный год

Содержание

1. Идентификация внешних систем для "FinTrack"

1.1. Банковские API

Примеры:

- Tinkoff API
- Sberbank API
- Alfa-Bank API

Данные для обмена:

Тип данных	Направление	Пример структуры (JSON)	Частота обновления
Список счетов	Банк → FinTrack	{"accounts": [{"id": "123", "balance": 50000}]}	1 раз/день
Транзакции	Банк → FinTrack	{"transactions": [{"date": "2024-05-20", "amount": -1000}]}	Реал-тайм (webhook)
Категории трат	FinTrack → Банк	{"category": "Food", "bankCategoryId": "12"}	При ручной настройке

Ограничения:

- Аутентификация: OAuth 2.0 (токен с TTL 1 час).
- Rate Limiting: 100 запросов/минуту (Tinkoff), 50/минуту (Sberbank).
- Форматы: JSON (основной), XML (для legacy-систем).

Юридические требования:

- Подписание договора с банком.
- Сертификация PCI DSS для работы с платежами.

Риски:

- Изменение формата API банком без обратной совместимости.
- Блокировка аккаунта при частых запросах.

Метод интеграции:

Банки: Event-Driven (Kafka)

Проблема:

- Банковские API имеют rate limits (например, 100 запросов/минуту).
- Прямые синхронные вызовы могут привести к отказам при высокой нагрузке.

Решение:

- Kafka как брокер сообщений:
    - Топики:
        1. bank.sync.requests (запросы на синхронизацию).
        2. bank.transactions (транзакции для обработки).
    - Поток данных:
        1. Пользователь инициирует синхронизацию → POST /sync → событие в bank.sync.requests.
        2. BankAdapter читает событие → запрашивает данные у Tinkoff API → сохраняет в БД → отправляет транзакции в bank.transactions.
        3. Сервис srv-transactions обрабатывает транзакции.

Почему Kafka, а не RabbitMQ?

Критерий	Kafka	RabbitMQ
Throughput	До 1M сообщений/сек	До 50K сообщений/сек
Надежность	Сохранение сообщений на диск	Потеря при падении очереди
Масштабируемость	Горизонтальное масштабирование	Вертикальное

Вывод: Kafka выбрана из-за высокой пропускной способности и надежности.

1.2. Сервис уведомлений

Варианты:

- Telegram Bot API (для push-уведомлений).
- SendGrid (email через SMTP/API).

Данные для обмена:

Тип уведомления	**Данные**	**Пример запроса (POST)**
Превышение бюджета	{"userId": 456, "message": "Лимит на еду превышен на 20%!"}	https://api.telegram.org/bot<TOKEN>/sendMessage
Напоминание о цели	{"email": "user@mail.com", "text": "До цели 'Отпуск' осталось 20 000 руб."}	SendGrid API v3

Ограничения:

Telegram:

- Лимит: 30 сообщений/секунду на бота.
- Обязательный HTTPS.

SendGrid:

- 100 emails/день на бесплатном тарифе.
- Требует DKIM/SPF-настроек для доставки.

Риски:

- Письма попадают в спам (для email).

Метод интеграции:
Проблема:

- Нужна простая и быстрая интеграция с Telegram/SendGrid.

Решение:

- REST API (HTTP/1.1 + HTTPS):
    - Эндпоинты:
        - POST /notify (отправка уведомлений).
        - GET /notify/status (проверка доставки).

    - Паттерн "Фасад":

Почему REST, а не WebSockets?

- REST:
    - Простота реализации.
    - Стандартные механизмы retry (экспоненциальная задержка).
- WebSockets:
    - Избыточен для одноразовых уведомлений.
    - Сложнее масштабировать.

1.3 Сервисы аналитики

Примеры:

- Google Analytics 4 (GA4).
- Amplitude (для поведенческой аналитики).

Данные для обмена:

Событие	Данные (JSON)	Частота
Открытие отчета	{"event": "report_view", "reportType": "monthly"}	В реальном времени
Экспорт данных	{"format": "PDF", "userId": "789"}	По запросу

Ограничения:

- GDPR: Анонимизация userId (хэширование email).
- GA4: Максимум 500 событий/секунду на проект.
- Формат: Только JSON.

Риски:

- Накопление больших объемов данных → затраты на хранение.

Метод интеграции:

Аналитика: gRPC (Protocol Buffers)

Проблема:

- Большой объем данных (события просмотров отчетов, клики).

Решение:

- gRPC + Protocol Buffers:
    - Преимущества:
        - Бинарный формат → меньше нагрузки на сеть.
        - Поддерка потоковой передачи (streaming).

Почему gRPC, а не REST?

Критерий	gRPC	REST (JSON)
Скорость	В 5-10 раз быстрее	Зависит от размера JSON
Типизация	Строгая (через .proto)	Динамическая
Поддержка	Сложнее для фронтенда	Универсальность

Вывод: gRPC выбран для внутренней аналитики из-за производительности.

Таблица ЗАР: Интеграция с внешними системами

Название решения	Контекст	Проблема	Принятое решение	Альтернативы	Последствия
Интеграция с Tinkoff API	Премиум-пользователи требуют автоматической синхронизации транзакций.	Банки используют разные API (форматы, аутентификация), возможны rate limits.	Адаптер (BankAdapter) + Kafka: - Единый интерфейс для всех банков. - Асинхронная обработка через Kafka для обхода rate limits.	1. Прямые REST-вызовы из сервиса. 2. Webhooks от банка.	Задержка данных до 5 мин. Требует поддержки Kafka.
Уведомления через Telegram	Пользователи должны получать мгновенные оповещения о бюджете.	Нужна надежная доставка, но SMS/email дороги или медленные.	Telegram Bot API: - HTTPS + Webhooks. - Кэширование сообщений в Redis при недоступности чата.	1. Email (SendGrid). 2. SMS (Twilio).	Зависимость от интернета. Лимит 30 сообщений/сек.
Аналитика через Google Analytics	Сбор данных о популярности отчетов для улучшения UX.	GDPR требует анонимизации данных.	GA4 + gRPC: - Анонимизация userId (хэш email). - gRPC для скорости.	1. Amplitude (дороже). 2. Внутренняя BI-система (сложность разработки).	Затраты на объем данных. Нужен DPA с Google.
Экспорт данных в PDF	Пользователи хотят выгружать отчеты в удобном формате.	Генерация PDF требует ресурсов и может тормозить систему.	Асинхронная генерация: - Отправка задачи в Celery. - S3 для хранения PDF.	1. Синхронная генерация (риск таймаутов). 2. Использование браузерных библиотек.	Задержка выдачи файла. Требует настроенного S3.

2. Swagger документаци

Документацию можно запустить на https://editor.swagger.io/
Необходимо вставить содержимое swagger.yaml

3. UML диаграмма взаимодействия с внешними сервисами