# Анализ SLA веб и мобильного приложения

Климачёва Екатерина М4200с

# Описание исходного SLA

- Продукт: веб- и мобильное приложение с платежным и пользовательским контуром.
- Команда: 2 backend, 2 frontend, mobile, QA, PO, Team Lead

Что фиксирует SLA

- Uptime: 99.5–99.9% для критических сценариев (авторизация, оплата, ЛК)
- TTR (Time to Response): 1–2 часа для критических инцидентов
- TTF (Time to Fix): ≤ 24 часа для P1
- Rollback rate: ≤ 10%
- Performance: LCP 2.0–2.5 сек
- USAT: ≥ 4.0

Задача SLA

- Установка измеримых критериев надежности и стабильности сервиса
- Регламентация сроков обработки инцидентов
- Определение целевых ориентиров эксплуатационного качества
- Закрепление зон ответственности между технической командой и бизнесом

# Перегибы, недочеты, риски

1. **Нереалистичные временные обязательства**
   - TTF (Time to Fix) < 24 ч для критических инцидентов
   - фактически нарушался в релизах №1, №3, №9
   - не учитывает сложность архитектурных и интеграционных дефектов
2. **Завышенные требования к доступности**
   - uptime 99.5% при зависимости от внешних API
   - релиз №3 показал фактический uptime 85%
   - ответственность за third-party не выделена отдельно
3. **Отсутствие приоритизации метрик**
   - все показатели SLA имеют одинаковый вес
   - бизнес-критичность инцидентов не отражена
4. **Штрафы и компенсации не определены**
   - что происходит при нарушении SLA, кто несет ответственность
   - Риск: Неопределенность в отношениях с бизнесом
5. **Отсутствие Force Majeure clause**
   - релизы №3, №7, №10 показали внешние факторы
   - Риск: Команда отвечает за то, что не контролирует

# Оценка SLA в контексте чёрных лебедей

Что срабатывает:

- формализованный TTR позволяет быстро начать реакцию;
- наличие метрик создает единое поле ожиданий;
- rollback как допустимый механизм стабилизации;
- observability и мониторинг поддерживают диагностику.

Что не срабатывает:

- одинаковый SLA для штатного и аварийного режима;
- жесткий TTF усиливает давление в кризисе;
- отсутствие сценариев потери данных и отказа инфраструктуры;
- нет матрицы приоритетов при множественных инцидентах.

# Что можно было бы не делать

1. **Не обещать Uptime 99.9%**

Без полноценного DevOps, резервирования не возможно. Релиз №3 показал, что даже 99.5% - верхняя граница.

2. **Не вводить штрафы за срыв релизов**

Практика показывает, что штрафы мотивируют скрывать проблемы, а не решать их. При этом «чёрные лебеди» происходили практически в каждом релизе.

3. **Не фиксировать одинаковый TTF для всех критических багов**

Архитектурная проблема и конфигурационный hotfix  требуют разного времени.

# Что можно доработать

1. Дифференциация SLA

- отдельные режимы: Normal / Degraded / Emergency;
- разные целевые показатели для каждого режима.

2. Матрица приоритетов инцидентов

- P0–P3 с различными TTR, TTF и ответственными;
- привязка к бизнес-метрикам.

3. SLA для внешних зависимостей

- fallback-механизмы;
- circuit breaker;
- договоренности с поставщиками API.

4. Усиление операционной зрелости

- формализованный rollback-процесс;
- freeze period перед релизами;
- operational readiness checklist.

# Вывод

**Оценка SLA: 7/10**

Плюсы:

- SLA использовался управленческим инструментом
- SLA имел четкие метрики.
- Команда пересматривала договоренности
- Улучшение метрик к релизам 8–10.

Минусы:

- SLA изначально переоценивал технические возможности команды.
- Слабо учитываются внешние и организационные риски.