ОБРАТНАЯ СВЯЗЬ ПО МОК-СОБЕСЕДОВАНИЮ

Общий вывод

Ты держался уверенно, был спокойным и открытым, хорошо рассказывал про опыт.

Но по технической части проявились системные провалы почти во всех блоках: SDLC, требования, архитектура, базы данных, REST, очереди, BPMN, безопасность.

Это не проблема - у многих джунов это так.

Проблема в другом: ты уверенно отвечал даже там, где не знаешь. Это усиливает ощущения несоответствия уровня.

При желании ты можешь быстро закрыть пробелы, но потребуется структура и честная работа над базой.

---

1. SDLC и роль системного аналитика

❌ Что не получилось

- Не знал, что такое SDLC и попросил расшифровку.
- Ответ про SDLC = «предпродакшн, продакшн, постпродакшн» - неправильная модель.
- Определение роли аналитика трактовал как «архитектор», что неверно.
- Не понимаешь разграничение ролей архитектор vs системный аналитик.

✔️ Что учить

- SDLC
- Роль системного аналитика:

  требования, сценарии, спецификация API, интеграции, артефакты.

- Отличие SA vs BA vs архитектор.

🔍 Как искать

`"SDLC основы для аналитика"`, `"роль системного аналитика простыми словами"`, `"system analyst vs solution architect"`.

---

2. Требования и документация (BRD, SRS, виды требований)

❌ Что не получилось

- Путаница BRD vs SRS (перепутал функциональные и нефункциональные требования).
- Называл «требования к срокам» как отдельный вид - это неверно.
- Не знал классификацию требований: бизнес, пользовательские, системные, бизнес-правила.
- SLA в моменте не знал.

✔️ Что учить

- BRD: ценность, цели, бизнес-требования.
- SRS: функциональные + нефункциональные требования + ограничения.
- Виды требований: бизнес / пользовательские / системные / бизнес-правила.
- SLA, SLO, SLI.

🔍 Поиск

`"BRD SRS разница просто"`, `"виды требований аналитика"`, `"SLA примеры"`.

---

3. Архитектура: MSA, SOA, транзакции, ACID

❌ Что не получилось

- MSA описал поверхностно, без ключевых понятий: независимые сервисы, fault isolation, CI/CD, scaling.
- SOA перепутал с user-story mapping - грубая ошибка.
- Транзакции описал неправильно (одна транзакция = один запрос - неверно).
- ACID назвал «масштабируемость» вместо устойчивости и долговечности.

✔️ Что учить

- MSA: сервисная изоляция, API, event-driven архитектура.
- SOA: ESB, оркестрация, централизованная шина.
- ACID: atomicity, consistency, isolation, durability.
- Что такое транзакции, commit/rollback.

🔍 Поиск

`"microservices for beginners"`, `"SOA vs microservices"`, `"ACID простыми словами"`.

---

4. REST / HTTP / API

❌ Что не получилось

- REST описал как «JSON и YAML протокол» - неверно (стиль, а не протокол).
- Нет понимания уровней зрелости REST.
- Разница PATCH и PUT - частично правильно, но с ошибками.
- API Gateway назвал частью хореографии - неверно.
- gRPC - не понимаешь концепцию (HTTP/2, protobuf, streaming).
- SOAP vs REST объяснил поверхностно и местами неверно.

✔️ Что учить

- REST: ресурсы, idempotency, коды ответов.
- PUT vs PATCH.
- gRPC: streaming, binary, IDL.
- API Gateway: routing, rate limiting, auth.
- SOAP: XML, WSDL, строгая схема.

Синхронные интеграции

- Модель: *запрос → ответ*, система ждёт результат.
- Примеры: **REST, SOAP, gRPC, RPC**.
- Когда нужны: сразу получить успех/ошибку.

Асинхронные интеграции

- Модель: сообщение отправлено → обработается позже, ответа сразу нет.
- Примеры: **Kafka, RabbitMQ, Webhooks, Pub/Sub**.
- Когда нужны: высокая нагрузка, очереди, событийная архитектура.

---

5. Очереди, Kafka, RabbitMQ

✔️ Что получилось

- Хорошо рассказал про partition, consumer, offset.
- Нормально ориентируешься в Kafka и RabbitMQ.
- Правильно объяснил DLQ и retry.

❌ Где слабости

- Часть терминов путаешь (pub/sub vs direct exchange).
- «Kafka хранит всё всегда» - не совсем корректно (retention).

- Не отличаешь оркестрацию от хореографии.

✔️ Что учить

- Kafka retention.
- Exchange types в RabbitMQ.
- Event-driven vs orchestration.

🔍 Поиск

`"Kafka for system analysts"`, `"RabbitMQ routing explained"`.

---

6. Базы данных: нормализация, индексы, JOIN, NoSQL

❌ Что не получилось

- SQL реляционные vs нереляционные перепутал полностью.
- Привёл Excel как нереляционную базу данных - критическая ошибка.
- Нормальные формы объяснил частично верно, но с большими ошибками.
- Шардирование / репликация - не знаешь.
- Денормализацию трактовал неверно (ускоряет запись - нет, чаще ускоряет чтение).
- JOIN (left/right) объяснил неверно.

✔️ Что учить

- Разница SQL / NoSQL.
- Нормальные формы по 2–3 предложения.
- JOIN: inner, left, right, full.
- Шардирование, репликация.
- Денормализация: когда нужна.

🔍 Поиск

`"реляционная база данных простыми словами"`, `"normal forms explained"`, `"join types visualization"`.

---

7. Практическое задание - ERD (пиццерия)

❌ Что не получилось

- FK поставлены в неправильном направлении.
- Users → Order_id - делает связь 1:1.
- Orders → OrderItem_id - логически неверно.
- Модель не соответствует условиям задачи.

✔️ Что учить

- Моделирование сущностей через: User → Order → OrderItem → Product.
- Как ставить FK.
- Как строить N:M через связующую таблицу.

🔍 Поиск

`"ERD basics many to many examples"`

---

8. JSON - выбрал правильный ответ, всё ок

Это единственная тема, где ошибок почти нет.

---

9. REST-задание (URL)

❌ Что не получилось

- Неправильный формат даты `'2020-10-10'`.

- Не понимаешь, когда user_id нужен, а когда - нет.

✔️ Что учить

- Query params без кавычек.
- user-scoped API (через токен).
- Формат URL: `GET /tickets?city=moscow&date=2020-10-10`.

🔍 Поиск

`"REST URL conventions"`, `"user scoped endpoints best practices"`.

---

10. Sequence diagram

разбирали

---

11. BPMN / UML

❌ Что не получилось

- XOR и inclusive gateway перепутал.
- Отличие activity diagram от BPMN - не понимаешь.
- Use case diagram - не знаешь.
- Много терминов вспоминал на ходу.

✔️ Что учить

- Типы BPMN gateways.
- Activity vs BPMN.
- Use case elements.
- Sequence diagram - базовые обозначения.

🔍 Поиск

`"BPMN gateways explained"`, `"activity vs BPMN difference"`.

12. Безопасность

❌ Что не получилось

- Authentication / authorization - перепутал местами.
- JWT описал неверно: токен не создаётся «при регистрации».
- HTTPS: указал OAuth вместо TLS - критическая ошибка.

✔️ Что учить

- JWT: структура (header.payload.signature).
- AuthN vs AuthZ.
- HTTPS = HTTP + TLS.

🔍 Поиск

`"JWT for beginners"`, `"authn vs authz explained"`,

ИТОГОВАЯ РЕКОМЕНДАЦИЯ

Ты не справился с большей частью технических вопросов, но это не значит, что ты «не аналитик».

Это значит, что у тебя **большие пробелы в базовой теории**, что абсолютно нормально для человека, который много работал в «бизнесовой» роли и меньше - в технической.

Чтобы реально выйти на уровень уверенного кандидата, тебе нужно закрыть блоки:

- SDLC
- требования (BRD/SRS, классификация)
- основы архитектуры
- реляционные БД + SQL

- REST / API
- HTTP
- очереди (Kafka, RabbitMQ - частично знаешь, но нужно системно)
- BPMN / UML
- безопасность
- ERD моделирование
