iOS · TECH LEAD · РФ-БИГТЕХ 2026

# Подготовка техлида iOS

## Алгоритмы и Mobile System Design

Единый учебник для опытного iOS-инженера и техлида, готовящегося к собеседованиям в Яндекс, Avito, Ozon и VK. Два курса, банк вопросов, видео-плейлисты с тайм-кодами, 10 диаграмм и обзор 26 проверенных ресурсов.

Составлено: июнь 2026 · Языки материалов: RU + EN

# Содержание

РАЗДЕЛ 0

# О комплекте

Как устроен учебник и как им пользоваться

## Два курса: Алгоритмы + Mobile System Design

> Комплект учебных материалов для опытного iOS-инженера/техлида, готовящегося в Яндекс, Avito, Ozon, VK.

> Микс авторской методички и кураторских публичных материалов (видео, статьи Medium/Habr, курсы) — с проверенными ссылками и атрибуцией.

> Подготовлено: июнь 2026.

## Структура комплекта

| Файл | Что внутри |
|---|---|
| README.md (этот файл) | Навигация, общий план, метрики готовности |
| `00_interview_questions.md` | Банк свежих вопросов с собесов: RU + Запад, iOS-блок, алгоритмы, system design, лид-секция |
| `01_resources_review.md` | Обзор и отбор курсов (алгоритмы + SD) + доп. источники (каналы, репо, блоги) |
| `02_video_playlists.md` | Кураторские видео-плейлисты (34 проверенных ролика, сгруппированы по темам) |
| `COURSE_A_Algorithms.md` | Курс A — Алгоритмы: 13 модулей, паттерны, задачи, протокол интервью, план на 8 недель |
| `COURSE_B_SystemDesign.md` | Курс B — Mobile System Design: фреймворк, 7 кейсов, чек-листы, план на 5–6 недель |

## Почему два отдельных курса

Это два разных навыка с разной механикой подготовки:

- Алгоритмы — лайвкодинг под таймер, тренируются повторением задач по паттернам. Длинный разгон (8 нед).
- System Design — открытая дискуссия и трейд-оффы, тренируется кейсами и моками. Быстрее набирается (5–6 нед), но для лида важнее.

Можно проходить последовательно (сначала алгоритмы, потом SD) или параллельно (если до собеса ≥10 недель).

## Рекомендуемый порядок прохождения

1. Прочитать `00_interview_questions.md` — понять, что именно спрашивают (калибровка ожиданий).

2. Просмотреть `01_resources_review.md` — выбрать 1–2 платных ресурса при желании; остальное бесплатно.
3. Идти по `COURSE_A` и `COURSE_B`, подтягивая видео из `02_video_playlists.md`.
4. С 3-й недели каждого курса — еженедельные мок-интервью (партнёр из Telegram-чатов или платформы из обзора).

## Общий график (вариант «12 недель до собеса»)

| Недели | Фокус |
|---|---|
| 1–8 | Курс A (Алгоритмы) — основной поток, 2 задачи/день |
| 4–9 | Курс B (System Design) — параллельно, 2–3 вечера/нед (стартует с 4-й недели) |
| 10 | Микс-ревью: слабые темы по обоим курсам |
| 11 | Мок-марафон: 3+ алго-мока + 3+ SD-мока |
| 12 | Полировка, отдых перед собесом, повтор дневника ошибок |

Вариант «спринт 6 недель» (сильная база): Курс A модули M0–M12 в ускоренном темпе (3 задачи/день) + Курс B параллельно с 1-й недели; моки с 3-й недели.

## Метрики готовности (пройти оба чек-листа)

Алгоритмы: medium-задача за ≤25 мин с верной Big-O; знаю паттерн на каждый модуль; пишу Swift без IDE; ≥5 моков. System Design: фреймворк (5 шагов) за ≤45 мин на любом кейсе; вскрываю скрытые требования; обсуждаю кэш/офлайн/модуляризацию; staff-уровень трейд-оффов; ≥4 моков.

## Принципы подготовки (общие для обоих курсов)

- Учить темы/паттерны, не количество задач (Habr).
- Интервальное повторение слабых тем.
- Дневник ошибок (шаблон в Курсе A).
- Думать вслух на интервью — интервьюер ведёт по подсказкам.
- Мок-интервью — главный множитель результата.

## Источники и легальность

Комплект объединяет публичные материалы (YouTube, Medium, Habr, открытые курсы) как кураторские ссылки с атрибуцией — без перезаливки и нарезки контента в отдельные файлы. Полные списки источников с рабочими URL — внутри каждого файла. Доступ к Medium — через ваш Google-аккаунт в локальном браузере.

РАЗДЕЛ 1

# Банк вопросов с собеседований

Что реально спрашивают в РФ-бигтех: iOS, алгоритмы, system design, лид-секция

> Свод актуальных вопросов и тем с собеседований для опытных iOS-инженеров и техлидов.

> Источники: RU-сегмент (Habr, Яндекс Практикум, ios-interview.ru) + западный сегмент (System Design Newsletter, Jacob's Tech Tavern, Medium, Exponent).

> Помечено: RU RU-источник, EN EN-источник, уровень сложности (★ junior/mid · ★★ senior · ★★★ staff/lead).

## 1. Структура собеседования в РФ-бигтех (Яндекс / Avito / Ozon / VK)

Типичная воронка для senior/lead iOS:

1. Скрининг (HR + краткий технический).
2. Алгоритмическая секция — лайвкодинг на платформе `code.yandex` / `code.avito` (онлайн-редактор без автодополнения и подсветки ошибок) (Habr: подготовка к алго-собесу в Яндексе).
3. Технический iOS-блок — язык Swift, рантайм, многопоточность, память, архитектура.
4. System Design / Mobile Architecture — проектирование клиента и end-to-end флоу.
5. Управленческая / лид-секция — для техлида: команда, делегирование, продукт (Яндекс Практикум: что спрашивают у тимлида).

Где исторически гоняют алгоритмы в РФ: Тинькофф Контест, Яндекс Контест, Ozon Route 256 и отборы в другие крупные IT-компании (Habr).

## 2. Алгоритмическая секция

### 2.1. Рекомендованный протокол ответа на алго-задачу RU ★★

Из разбора реального собеса в Яндексе (Habr):

1. Внимательно прочитать условие.
2. Задать уточняющие вопросы: отсортирован ли массив; только ASCII; возможны ли дубликаты; можно ли менять массив in-place или нужно вернуть новый.
3. Подробно описать алгоритм словами (думать вслух).
4. Проверить на тест-кейсах, оценить по времени и памяти (Big-O).
5. Сообщить о готовности → писать код.
6. После — повторно пройтись, исправить синтаксис/логику, мысленный дебаг.
7. Не молчать ни на одном этапе.

### 2.2. Ключевые паттерны (готовиться по темам, не по числу задач) ★★

- Sliding Window (плавающее окно) — пример: Permutation in String, Max Consecutive Ones III.
- Two Pointers.
- Бинарный поиск (оценить сложность — O(log n)).
- Деревья: обход бинарного дерева, Find Duplicate Subtrees.
- Hashing / коллизии хеш-функций (часто всплывает и в теории).
- BFS/DFS, графы.
- Динамическое программирование.
- Heap / приоритетная очередь.
- Стек/очередь, intervals, prefix sum.

Прямо упомянутые типы задач: плавающее окно, бинарное дерево (Habr).

> Топ-50 паттернов LeetCode для iOS-собесов и подборки см. в разделе ресурсов (`01_resources_review.md`).

## 3. Технический iOS-блок (Senior)

Из разбора senior-вопросов (ios-interview.ru, ч.1):

| # | Вопрос | Тема | Ур. |
|---|---|---|---|
| 1 | Сколько потоков может выполняться одновременно без программного ограничения? | Многопоточность, CPU, параллелизм | ★★ |
| 2 | Перечислите состояния `Operation` | Operation, жизненный цикл, state machine | ★★ |
| 3 | Чему будет равен `count` после выполнения? | Data Race, атомарность инкремента | ★★ |
| 4 | Что выведет `MemoryLayout`? | Размер структуры, выравнивание памяти | ★★ |
| 5 | Зачем нужен `@inline`? | Оптимизация компилятора, `@inline(never/__always)` | ★★ |
| 6 | Что выведет программа? | Диспетчеризация методов, extension протокола, SR-103 | ★★ |
| 7 | Что такое `CALayer`, отличие от `UIView`? | Слои, обработка событий, рисование | ★ |
| 8 | Что такое `RunLoop`? | Цикл событий, источники ввода | ★★ |
| 9 | Что такое коллизия хеш-функции? | Хеширование, коллизии | ★ |
| 10 | Оцените сложность алгоритма поиска | Бинарный поиск, O(log n) | ★ |

Дополнительные частые темы для senior+ (по совокупности RU/EN-источников):

- ARC, retain cycles, `weak`/`unowned`, утечки памяти и их детект.

- GCD vs `Operation` vs Swift Concurrency (`async/await`, `actor`, `Task`, `TaskGroup`, `@MainActor`, structured concurrency).
- `value` vs `reference` типы, copy-on-write.
- Method dispatch: static / dynamic / witness table / message dispatch.
- SwiftUI vs UIKit, жизненный цикл, `@State/@Binding/@Observable`, Observation framework.
- Архитектуры: MVC, MVVM, MVVM+C, VIPER, RIBs, TCA — плюсы/минусы.
- Сетевой слой, кэширование, `URLSession`, фоновые задачи.

---

# 4. System Design / Mobile Architecture

## 4.1. Фреймворк ответа (универсальный) EN ★★

Из System Design Newsletter и Jacob's Tech Tavern (2026):

1. Requirements & scope (5–10 мин): уточнить, что проектируем и ограничения.
2. API & data needs (5–10 мин): что приложению нужно от сервера; data model.
3. Architecture (10–15 мин): клиентские компоненты и поток данных.
4. Deep dive (15–20 мин): один критичный флоу или адаптация под новое требование.
5. Wrap-up (1–5 мин): резюме решений и трейд-оффов.

Шаги по Jacob Bartlett: Scoping → Functional reqs → Non-functional reqs → Data model → API design → High-level design → Drill-down. Главный принцип: «измерь дважды, отрежь один раз»; интервью больше про data flow и структуру, чем про UI/фреймворки. Не бросаться сразу в SwiftUI vs UIKit / MVVM vs VIPER.

## 4.2. Уточняющие вопросы в начале EN ★★

- Что такое успех для фичи? (скорость / надёжность / простота)
- Это всё приложение или одна часть? Какие платформы (iOS only / cross-platform)?
- Сетевые условия: всегда онлайн / нестабильно / полный офлайн?
- Кто пользователи и на каких устройствах (флагманы / low-end)?
- Технические ограничения: интеграция с легаси, существующая авторизация?

## 4.3. Частые задачи на проектирование EN ★★/★★★

- Design Instagram feed / media-heavy лента с офлайн-поддержкой.
- Design chat / messaging (с офлайн и end-to-end шифрованием) — уточнять: WhatsApp-like (E2E, real-time, по номеру), Slack-like (треды, поиск, интеграции) или Twitter DMs (простой, текст, соц-граф).
- Design ride-sharing app.
- Share photos with friends (фоновая загрузка, чанкинг для 50MB видео).
- Design analytics SDK для крупной компании (Jacob).
- Design news feed на соцприложении.

## 4.4. Что ожидают от уровней EN

| Уровень | Ожидания |
|---|---|
| Junior/Mid | Уточняющие вопросы, базовая диаграмма, объяснить кэширование/фон/lifecycle, простые трейд-оффы. |
| Senior ★★ | Проактивно вскрывать скрытые требования («а что с лимитом трафика?»), обосновывать выбор (repository pattern для offline-first), глубокое знание mobile-specific (lifecycle, memory pressure, battery, offline sync), адаптация при смене требований, вести разговор. |
| Staff/Lead ★★★ | Фреймить проблему («а нужен ли вообще офлайн?»), думать о структуре команды и владении модулями, долгосрочно (дизайн под будущие фичи), системные трейд-оффы: модуляризация, build time, деплой, обратная совместимость; мыслить как архитект/техлид. |

Источники: System Design Newsletter, Jacob's Tech Tavern.

## 5. Лид / управленческая секция (для техлида) RU ★★★

Из Яндекс Практикум:

| Блок | Что проверяют | Примеры вопросов |
|---|---|---|
| Технический | Понимание устройства системы, архитектурные решения, баланс архитектуры/бизнеса/команды | «Как выбрать стек, если команда на нём не работала?»; «Как спроектировать систему, выдерживающую рост нагрузки ×10 за год?» |
| Управленческий | Мотивация, разрешение конфликтов, планирование, распределение | «Команда выгорела, сроки поджимают — действия?»; «Два сеньора спорят про ORM — как решите?» |
| Продуктовый | Логика бизнеса, компромисс «идеальный код vs рабочая фича» | «Как убедить заказчика отказаться от ненужной фичи?»; «Три задачи, ресурс на одну — как выбираете?» |

Ожидаемые компетенции лида: техническая экспертиза (быть авторитетом, а не самым быстрым кодером), проектное мышление, делегирование, работа с людьми, принятие решений в неопределённости, сильные soft skills.

Признаки готовности к роли: ведёте проекты от планирования до релиза, менторите младших, участвуете в продуктовых обсуждениях, подменяете лида.

## 6. Расширенный банк iOS-вопросов с опорными ответами (senior → staff)

Ниже — ключевые вопросы технической секции с каркасами сильных ответов. ★ — базовый senior, ★★ — уверенный senior, ★★★ — staff/лид. Цель опорного ответа — структура и глубина, а не заученный текст: проговаривайте trade-offs вслух.

### 6.1. Память и ARC ★★

Что такое retain cycle и как его разорвать? ARC считает сильные ссылки; цикл возникает, когда два объекта (или замыкание и его владелец) держат друг друга сильно — счётчик не доходит

до нуля, память течёт. Разрыв: одну из ссылок делаем `weak` или `unowned`. Классика — `self` внутри escaping-замыкания: `[weak self] in guard let self else { return }`.

`weak` vs `unowned` — когда что? `weak` — опциональна, автоматически становится `nil` при деаллокации цели; брать, когда время жизни цели короче или независимо (делегаты, родитель↔поток). `unowned` — не опциональна, не зануляется; брать, только когда цель гарантированно жива не меньше владельца (обращение после деаллокации = краш). По умолчанию выбирайте `weak` — безопаснее.

Как искать утечки? Instruments (Leaks, Allocations), Xcode Memory Graph Debugger (видно, кто держит объект), счётчик `deinit` в логах (не вызвался — объект жив). На уровне staff упомяните системные последствия: рост резидентной памяти → Jetsam/OOM-kill на устройстве, отдельный класс крашей без стектрейса.

Copy-on-write у value-типов? `Array`/`Dictionary`/`String` хранят буфер по ссылке и копируют его только при мутации, если на буфер >1 владельца — отсюда дешёвое присваивание. В своих типах воспроизводится через `isKnownUniquelyReferenced(&storage)`.

## 6.2. Concurrency ★★ / ★★★

GCD vs Operation vs Swift Concurrency? GCD — низкоуровневые очереди (serial/concurrent), просто и быстро, но ручное управление зависимостями и отменой. `Operation`/`OperationQueue` — объектная модель: зависимости, отмена, приоритеты, KVO. Swift Concurrency (`async/await`, акторы) — структурированная конкурентность с проверкой на этапе компиляции, автоматической отменой и защитой от гонок. Тренд — миграция на Swift Concurrency.

Что такое data race и как защититься? Гонка данных — одновременный доступ к общему состоянию, где хотя бы один доступ на запись, без синхронизации → UB. Защита: serial-очередь, барьер на concurrent-очереди для записи (`.barrier`), `NSLock`/`os_unfair_lock`, либо актор (изолирует состояние на уровне языка). Ищется Thread Sanitizer (TSan).

Актор и `@MainActor`? Актор сериализует доступ к своему изолированному состоянию — снаружи только через `await`, поэтому гонок по этому состоянию нет by design. `@MainActor` — глобальный актор главного потока; помечаем им UI-код, чтобы компилятор гарантировал обращения к UIKit/SwiftUI с главного потока.

`async let` / `TaskGroup` vs ручной GCD-fan-out? `async let` запускает дочерние задачи параллельно и ждёт их в точке `await`; `TaskGroup` — динамическое число задач. Преимущество над GCD: структурированность (дочерние задачи не переживают родителя), автоматическая проброска отмены и ошибок.

`Sendable` и strict concurrency (Swift 6)? `Sendable` — маркер типа, безопасного для передачи через границы изоляции. В Swift 6 strict concurrency компилятор требует доказательства потокобезопасности при пересечении актор-границ — несоблюдение = ошибка компиляции, а не краш в проде. Для staff: уметь рассказать стратегию поэтапной миграции (warnings → errors по модулям).

## 6.3. Swift и рантайм ★★

Виды диспетчеризации методов? Static (прямой вызов, известен на компиляции — `final`, `struct`, глобальные функции), virtual table (наследование классов), witness table (методы протокола), dynamic (`@objc dynamic`, Obj-C runtime). Static — быстрее, допускает инлайнинг; dynamic — нужен для KVO/swizzling.

Value vs reference типы? `struct`/`enum` — семантика значения, копируются, живут на стеке/инлайн, потокобезопаснее. `class` — ссылка, общее изменяемое состояние, ARC. Совет: предпочитать value-типы, ссылочные — когда нужна идентичность или общее состояние.

Ловушка диспетчеризации в extension протокола (SR-103)? Метод, объявленный только в extension протокола (не в самом протоколе), вызывается статически по типу переменной, а не динамически по реальному типу — переопределение в конформере «не срабатывает» при вызове через тип протокола. Чтобы получить динамику — объявлять требование в самом протоколе.

`some` vs `any`? `some P` — opaque type: конкретный, но скрытый тип, известный компилятору, без боксинга, со static-диспетчеризацией. `any P` — existential: стирание типа, боксинг, динамическая диспетчеризация, дороже. Для возвращаемых значений и SwiftUI-вью предпочтителен `some`.

## 6.4. UI и жизненный цикл ★ / ★★

`CALayer` vs `UIView`? `UIView` — обработка событий, иерархия, Auto Layout; рисованием занимается его backing-слой `CALayer` (геометрия, контент, анимация на GPU). Тяжёлую графику/анимацию делают на уровне слоёв.

RunLoop — зачем? Цикл обработки событий потока: источники ввода, таймеры, перерисовка. Главный RunLoop крутит UI; понимание режимов (`.common` vs `.default`) важно, например, чтобы таймер не «замирал» при скролле.

`@State` / `@Binding` / `@Observable` (Observation, iOS 17+)? `@State` — локальное владение состоянием вью; `@Binding` — двусторонняя ссылка на чужое состояние; макрос `@Observable` (заменяет `ObservableObject`/`@Published`) даёт точечную инвалидцию только реально использованных свойств → меньше лишних перерисовок.

SwiftUI vs UIKit — что выбрать? SwiftUI — декларативно, быстрее прототипировать, единый код под платформы Apple, но меньше тонкого контроля и есть ограничения на старых iOS. UIKit — зрелый, полный контроль, сложные кастомные интерфейсы. На практике — гибрид (`UIHostingController`/`UIViewRepresentable`); ответ staff: выбор по требованиям, команде и минимальной поддерживаемой версии.

## 6.5. Архитектура и масштаб (staff) ★★★

MVVM vs VIPER vs TCA? MVVM — простой, тестируемый ViewModel, но риск «massive ViewModel». VIPER — строгое разделение (View/Interactor/Presenter/Entity/Router), хорош для больших команд, но многословен. TCA (The Composable Architecture) — однонаправленный поток, композиция, отличная тестируемость, но кривая входа и зависимость от фреймворка. Выбор — по размеру команды/проекта, а не по моде.

Модуляризация — trade-offs? Разбиение на SPM-модули: параллельная сборка, изоляция, переиспользование, явные границы. Минусы: рост времени сборки при плохом графе зависимостей, накладные расходы на инфраструктуру, циклы зависимостей. Staff-ответ: разрезать по доменам, выделять интерфейсные таргеты, мерить время сборки.

Dependency Injection? Явная передача зависимостей (init/property/контейнер) вместо синглтонов → тестируемость и заменяемость. Composition root — единая точка сборки графа на старте. Для лида важно объяснить компромисс между удобством и связанностью.

Dynamic feature delivery / feature flags? Фиче-флаги (remote config) — включение функциональности без релиза, A/B, поэтапный rollout, kill-switch. Trade-off: рост комбинаторной

сложности и техдолга «мёртвых» флагов — нужна гигиена удаления.

Версионирование API и обратная совместимость? Клиент на руках у пользователей и обновляется не сразу → сервер обязан держать старые версии. Стратегии: версия в пути/хедере, аддитивные изменения, graceful-деградация фич, форс-апдейт как крайняя мера. Это любимая зона mobile system design.

Источники опорных ответов: ios-interview.ru — Senior, ч.1, ios-interview.ru — Senior, ч.2, Habr — iOS-собеседования 2025.

## 7. Специфика компаний РФ-бигтеха

Форматы отличаются: где-то жёсткий лайвкодинг в своём редакторе, где-то упор на продуктовый разрез и архитектуру. Узнавайте формат у рекрутёра заранее.

| Компания | Алгоритмы | System Design | Особенности |
|---|---|---|---|
| Яндекс | Лайвкодинг в собственном редакторе (code.yandex), задачи уровня Контеста: sliding window, деревья, графы, бинарный поиск | Mobile/общий SD на старших грейдах | Отдельная лид-секция (управленческий + продуктовый блоки); ценят рассуждение вслух и чистоту кода |
| Avito | Задачи medium с упором на паттерны и структуры данных; свой редактор | Архитектура мобильного приложения, модуляризация | Сильный продуктовый разрез — почему такое решение, метрики, влияние на бизнес |
| Ozon | Контесты (Route 256 и аналоги), классические алго-задачи | Проектирование под высокую нагрузку | Акцент на нагрузку и отказоустойчивость, бэкенд-мышление даже у мобильщиков |
| VK | Алгоритмы + структуры данных, классический формат | Масштаб, real-time, ленты/мессенджеры | Акцент на масштаб и real-time, продуктовая логика крупных соцсервисов |

Общий совет. В алго-секции всегда проговаривайте Big-O по времени и памяти и идите от наивного решения к оптимальному. В System Design начинайте со сбора требований (функциональные/нефункциональные, масштаб) и обязательно поднимайте mobile-специфику: офлайн, синхронизация, экономия батареи/трафика, версионирование API, размер приложения.

## 8. Поведенческая секция и метод STAR

Лид-секция почти всегда включает поведенческие вопросы. Сильный ответ строится по STAR: Situation (контекст) → Task (задача/ответственность) → Action (что лично сделали) → Result (измеримый итог + вывод). Заранее подготовьте 5–6 историй из опыта, которые можно переиспользовать под разные вопросы.

| Вопрос | Что проверяют | Каркас сильного ответа |
|---|---|---|
| Конфликт в команде (два сеньора спорят о решении) | Разрешение конфликтов, лидерство без давления | S: спор о подходе → T: разблокировать команду → A: свёл к данным/прототипу, дал решающий критерий → R: выбрали вариант, отношения сохранили |
| Команда выгорела, сроки горят | Забота о людях + управление ожиданиями | S: перегруз → T: сдать без потери людей → A: пересмотрел скоуп со стейкхолдерами, перераспределил, защитил команду → R: сдали ядро, выгорание сняли |
| Сложное решение в неопределённости | Принятие решений при нехватке данных | S: мало данных, дедлайн → T: выбрать направление → A: зафиксировал допущения, обратимое решение, метрика для проверки → R: подтвердили/скорректировали по факту |
| Ваша неудача / факап | Рефлексия, ответственность, рост | S: что пошло не так → T: ваша зона ответственности → A: признали, починили, ввели процесс/тест → R: что изменили навсегда |
| Менторинг и рост команды | Развитие людей, делегирование | S: джун/отстающий → T: вырастить → A: план роста, ревью, делегирование с возрастающей сложностью → R: повышение грейда/самостоятельность |
| Убедить заказчика отказаться от фичи | Продуктовое мышление, влияние | S: фича дорогая/вредная → T: защитить продукт и сроки → A: данные, альтернатива, разговор на языке бизнеса → R: отказались/упростили, выиграли ресурс |

Что демонстрировать лиду. Технический авторитет (а не «самый быстрый кодер»), делегирование вместо «сделаю сам», заботу о людях и продукте, решения в неопределённости, сильные soft skills. Источник по структуре лид-секции: Яндекс Практикум — что спросят тимлида.

## 9. Дополнительные подборки вопросов (для расширения банка)

- Medium — Advanced iOS architecture interview questions (Part 4), Anuj Kumar — system-level: модуляризация, dynamic feature delivery, CoreML/AI в архитектуре, версионирование API. (доступ через локальный браузер)
- Medium — 70 iOS Interview Questions (Apple, Meta, Amazon)
- Medium — iOS Developer Interview God List (850 вопросов)
- Habr — Object Oriented Design: подготовка к сложному интервью
- ios-interview.ru — Senior, часть 2
- ENIGMA AI — iOS собеседование 2026 (Swift, SwiftUI, архитектура)
- GitHub: weeeBox/mobile-system-design — фреймворк mobile SD интервью

РАЗДЕЛ 2

# Курс А — Алгоритмы

13 модулей по паттернам, протокол интервью, дневник ошибок, план на 8 недель

### Трек для опытного iOS-инженера / техлида

> Цель: уверенно проходить алгоритмическую секцию (лайвкодинг на `code.yandex` / `code.avito` / CodeRun) в Яндекс, Avito, Ozon, VK.

> Длительность: 8 недель (можно сжать до 6 при высокой базе).

> Принцип: учим паттерны, а не количество задач. Освоив паттерн, решаешь десятки похожих задач (Habr: подготовка к алго-собесу в Яндексе).

> Язык решений: Swift (с заметками про подводные камни онлайн-редактора без автодополнения).

## Как устроен курс

Каждый модуль = теория → паттерн → 1 разобранный пример → набор задач для закрепления → видео/материалы. В конце — протокол интервью, дневник ошибок и недельный план.

Ключевая методика подготовки (из разбора реального собеса в Яндексе, Habr):

- Готовиться по темам/паттернам, не прорешивать всё подряд.
- На задачу — до 30 минут самостоятельно, потом подсказка/идея (без готового кода).
- Интервальное повторение: вернуться к теме через несколько дней.
- Вести дневник ошибок.
- «Компилятор в голове» — критически перечитать код перед сдачей.
- Мок-интервью с партнёром (искать в Telegram-чатах по алгоритмам).

## Модуль 0. Фундамент: сложность и модель вычислений

Теория: асимптотика Big-O / Ω / Θ, амортизированный анализ, оценка по времени и по памяти, типичные классы сложности (O(1), O(log n), O(n), O(n log n), O($n^2$), O($2^n$)). Зачем: интервьюер ВСЕГДА просит оценить решение; неверная оценка → критика и переоценка (Habr).
Материалы:

- ▶ Сложность и модели вычислений — М. А. Бабенко (Yandex for ML) и плейлист курса (12 видео).
- ▶ Алгоритмы и структуры данных. Введение. Массивы (VK Team).
- Stepik: Алгоритмы — теория и практика. Методы (CS Center).

Чек-практика: для 10 произвольных решённых задач напишите оценку времени и памяти и обоснуйте.

## Модуль 1. Массивы, строки, два указателя (Two Pointers)

Паттерн: два указателя (встречные / попутные), in-place преобразования. Уточняющие вопросы (всегда задавать): отсортирован ли массив; только ASCII; возможны ли дубликаты; можно ли менять массив in-place (Habr). Разобранный пример — Two Sum (отсортированный массив):

```
func twoSum(_ nums: [Int], _ target: Int) -> (Int, Int)? {
    var l = 0, r = nums.count - 1            // O(1) память
    while l < r {                            // O(n) время
        let s = nums[l] + nums[r]
        if s == target { return (l, r) }
        else if s < target { l += 1 }
        else { r -= 1 }
    }
    return nil
}
```

Задачи: Two Sum / Two Sum II, Valid Palindrome, Container With Most Water, 3Sum, Remove Duplicates, Trapping Rain Water (★★★).

## Модуль 2. Sliding Window (плавающее окно)

Паттерн: окно фиксированной/переменной длины; явно упомянут как тема собесов в Яндексе (Habr). Задачи: Permutation in String, Max Consecutive Ones III, Longest Substring Without Repeating Characters, Minimum Window Substring (★★★), Best Time to Buy and Sell Stock.

## Модуль 3. Хеш-таблицы и множества

Теория: хеш-функции, коллизии (частый теоретический вопрос — см. банк вопросов), разрешение коллизий (цепочки/открытая адресация), сложность операций. В Swift: `Dictionary`, `Set`, протокол `Hashable`. Задачи: Group Anagrams, Top K Frequent Elements, Subarray Sum Equals K, Longest Consecutive Sequence.

## Модуль 4. Стек, очередь, дек

Паттерн: монотонный стек, парные скобки, очередь через два стека. Задачи: Valid Parentheses, Min Stack, Daily Temperatures, Largest Rectangle in Histogram (★★★), Sliding Window Maximum (дек).

## Модуль 5. Связные списки

Паттерн: fast/slow указатели, разворот, dummy-узел. Задачи: Reverse Linked List, Linked List Cycle, Merge Two Sorted Lists, Reorder List, Remove Nth Node, LRU Cache (★★★ — пересекается с System Design!).

## Модуль 6. Бинарный поиск

Паттерн: поиск по значению и по ответу (binary search on answer); оценить сложность O(log n) (ios-interview.ru senior). Задачи: Binary Search, Search in Rotated Sorted Array, Find Minimum in Rotated Array, Koko Eating Bananas, Median of Two Sorted Arrays (★★★).

## Модуль 7. Деревья (бинарные, BST)

Паттерн: DFS (pre/in/post), BFS по уровням, рекурсия; явно упомянуты «бинарное дерево» и Find Duplicate Subtrees (Habr). Задачи: Invert Binary Tree, Max Depth, Diameter, Level Order Traversal, Validate BST, Lowest Common Ancestor, Find Duplicate Subtrees, Serialize/Deserialize (★★★).

## Модуль 8. Графы (BFS/DFS, топосортировка, Union-Find)

Паттерн: обход матрицы/списка смежности, поиск компонент, обнаружение циклов, кратчайшие пути (BFS/Dijkstra). Задачи: Number of Islands, Clone Graph, Course Schedule (топосорт), Pacific Atlantic Water Flow, Word Ladder, Number of Connected Components (Union-Find).

## Модуль 9. Куча / приоритетная очередь

Теория: binary heap, k-й элемент, слияние k списков. В Swift нет встроенной heap — уметь реализовать или использовать сортировку/`Heap` из swift-collections. Задачи: Kth Largest Element, Top K Frequent, Merge k Sorted Lists, Find Median from Data Stream (★★★), Task Scheduler.

## Модуль 10. Backtracking

Паттерн: перебор с возвратом, дерево решений, отсечения. Задачи: Subsets, Permutations, Combination Sum, Word Search, N-Queens, Palindrome Partitioning.

## Модуль 11. Динамическое программирование

Теория: оптимальная подструктура, перекрывающиеся подзадачи, мемоизация ↔ табуляция, 1D/2D DP. Отдельный сильный разбор:

- ▶ Dynamic Programming — freeCodeCamp (5 ч, EN).

Задачи: Climbing Stairs, House Robber, Coin Change, Longest Increasing Subsequence, Longest Common Subsequence, Edit Distance, 0/1 Knapsack, Unique Paths, Word Break.

## Модуль 12. Жадные алгоритмы и интервалы

Паттерн: сортировка + жадный выбор, слияние интервалов. Задачи: Merge Intervals, Insert Interval, Non-overlapping Intervals, Jump Game, Gas Station, Meeting Rooms II.

## Модуль 13 (опционально, для отборов/контестов). Продвинутые темы

Дерево отрезков, битовые операции, B-деревья, коды Хэмминга — встречаются в Тренировках Яндекса и отборах (Тинькофф Контест, Яндекс Контест, Ozon Route 256) (Habr).

---

## Протокол ответа на алго-интервью (распечатать и отрепетировать)

1. Прочитать условие внимательно.
2. Задать уточняющие вопросы (сортировка, дубликаты, диапазон, ASCII, in-place?).

3. Проговорить алгоритм словами — думать вслух.
4. Привести тест-кейсы, оценить время/память (Big-O).
5. Сообщить о готовности → писать код.
6. Перечитать, исправить синтаксис/логику, мысленный дебаг, граничные случаи.
7. Сдать и проговорить итог.

> Никогда не молчать: интервьюер ведёт по подсказкам, если виден ход мысли (Habr).

Особенности онлайн-редактора (`code.yandex`/`code.avito`/CodeRun): нет автодополнения и подсветки ошибок — тренируйтесь писать Swift «на чистом листе», знайте сигнатуры стандартной библиотеки наизусть.

---

## Дневник ошибок (шаблон)

| Дата | Задача | Тип ошибки (синтаксис/логика/Big-O/edge case) | Что понял | Повтор через |
|---|---|---|---|---|

---

## Недельный план (8 недель)

| Неделя | Модули | Задач/нед | Мок |
|---|---|---|---|
| 1 | M0–M1 (сложность, two pointers) | 12–15 | — |
| 2 | M2–M3 (sliding window, хеши) | 15 | — |
| 3 | M4–M5 (стек/очередь, списки) | 15 | 1 |
| 4 | M6–M7 (бин. поиск, деревья) | 15 | 1 |
| 5 | M8–M9 (графы, куча) | 15 | 1 |
| 6 | M10–M11 (backtracking, DP) | 15 | 2 |
| 7 | M12 + слабые места | 15 | 2 |
| 8 | Mixed review + мок-марафон | 10 | 3+ |

> Темп ~2 задачи/день, до 30 мин на задачу, интервальное повторение слабых тем.

---

## Рекомендованные ресурсы (детали — в `01_resources_review.md`)

Бесплатный старт: Тренировки Яндекса → NeetCode 150 → Яндекс Практикум: подготовка к алго-собесу → AlgoMaster patterns + BackToBackSWE. Платно (по желанию): balun.courses (под РФ-бигтех) + algocode.io (мок-интервью) + Grokking Coding Interview (Educative).

## Критерии готовности

- [ ] Решаю medium-задачу за ≤25 мин с верной Big-O оценкой.
- [ ] Знаю по 1 паттерну на каждый модуль и могу его проговорить.
- [ ] Пишу Swift без IDE-подсказок.
- [ ] Прошёл ≥5 мок-интервью, веду дневник ошибок.

РАЗДЕЛ 3

# Курс B — Mobile System Design

Фреймворк ответа, кэш/офлайн, архитектура iOS, 7 кейсов, план на 5–6 недель

### Трек для опытного iOS-инженера / техлида

> Цель: уверенно проходить секцию System Design / Mobile Architecture в Яндекс, Avito, Ozon, VK — на уровне senior и техлид/staff.

> Длительность: 5–6 недель.

> Акцент: мобильная специфика (lifecycle, память, батарея, офлайн, сеть) и data flow, а не UI/фреймворки.

> Главный принцип: «измерь дважды, отрежь один раз» — понять задачу до проектирования (Jacob's Tech Tavern, 2026).

## Модуль 0. Что оценивают на уровне техлида

В отличие от IC, от лида/staff ждут системного мышления (System Design Newsletter):

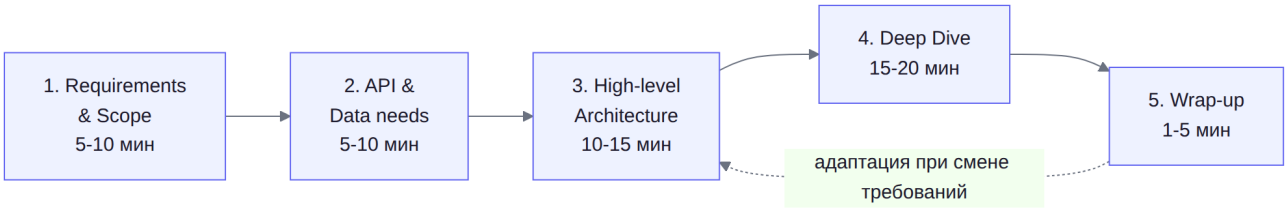
- Senior: проактивно вскрывать скрытые требования, обосновывать архитектуру (repository pattern для offline-first), глубокое знание mobile-specific, адаптация при смене требований, вести разговор.
- Staff/Lead: фреймить проблему («а нужен ли офлайн вообще?»), думать о структуре команды и владении модулями, проектировать под будущее, системные трейд-оффы (модуляризация, build time, деплой, обратная совместимость), мыслить как архитект.

В РФ для лида добавляется управленческо-продуктовый разрез (см. `00_interview_questions.md`, раздел 5): «как спроектировать систему, выдерживающую ×10 нагрузки за год?», «два сеньора спорят про подход — как решите?» (Яндекс Практикум).

Материалы модуля:

- ▶ iOS System Design: чем уникален мобильный систем-дизайн — FaangTalk 62 RU
- ▶ Как проходит интервью Mobile System Design & PM — Yandex for Mobile RU

## Модуль 1. Универсальный фреймворк ответа




D1 · Фреймворк System Design (5 шагов)

Структура, проверенная на реальных интервью (System Design Newsletter, Jacob Bartlett):

1. Requirements & Scope (5–10 мин) — что проектируем + ограничения.
2. API & Data needs (5–10 мин) — что приложению нужно от сервера; data model.
3. High-level architecture (10–15 мин) — клиентские компоненты + поток данных.
4. Deep dive (15–20 мин) — один критичный флоу / адаптация под новое требование.
5. Wrap-up (1–5 мин) — резюме решений и трейд-оффов.

Альтернативная детализация (Jacob): Scoping → Functional → Non-functional → Data model → API design → High-level → Drill-down.

Reusable-паттерн на каждом шаге: задать 2–3 высокоэффективных вопроса → переформулировать и получить «да» → записать решения/ограничения → рано подсветить red flags → не обязательно решать всё, показать осознанный трейд-офф.

Материалы:

- ▶ Как пройти System Design Интервью? 4 этапа, 2 качества — В. Невзоров RU
- ▶ Почему все проваливают собеседование по System Design? — В. Балун RU
- GitHub: weeeBox/mobile-system-design — фреймворк EN

---

## Модуль 2. Сбор требований (Requirements Gathering)

Уточняющие вопросы (System Design Newsletter):

- Что = успех фичи? (скорость/надёжность/простота) Что одно важнее всего?
- Всё приложение или часть? Платформы (iOS only / cross-platform)?
- Сетевые условия (всегда онлайн / нестабильно / офлайн)?
- Кто пользователи, какие устройства (флагманы / low-end)?
- Технические ограничения (легаси, существующая авторизация)?

Red flags, которые нужно подсветить рано: размытые требования («сделай быстро»), противоречия («просто, но кастомизируемо»), отсутствие данных об офлайне/авторизации/источнике данных.

Практика: возьмите промпт «Design a messaging feature» и определите, что это — WhatsApp-like (E2E, real-time, по номеру) / Slack-like (треды, поиск) / Twitter DMs (текст, соц-граф).

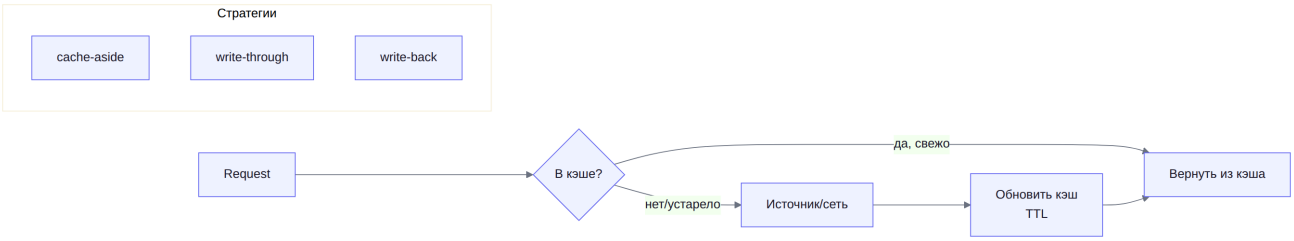
---

## Модуль 3. Базовые концепты систем (общая база)

Клиент-сервер, REST/GraphQL/gRPC/WebSockets, БД (SQL/NoSQL), индексы, репликация/шардирование, очереди, балансировка, CAP, согласованность. Материалы:

- ▶ 8 Most Important System Design Concepts — ByteByteGo EN
- ▶ 20 концептов System Design за 10 минут — AlexTech IO RU
- ▶ System Design RoadMap — В. Невзоров RU
- Высоконагруженные приложения (DDIA): разбор гл.1 — В. Невзоров RU

## Модуль 4. Кэширование и офлайн (mobile-critical)

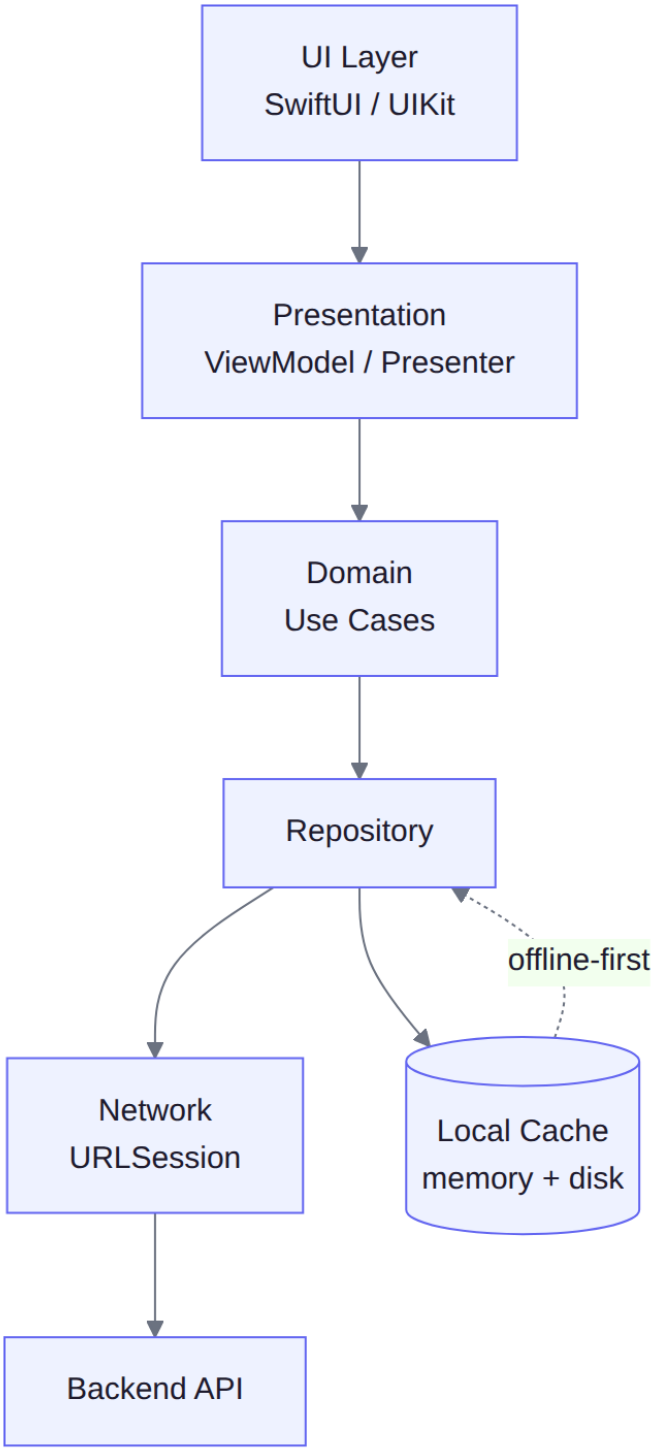



D7 · Стратегии кэширования

Уровни кэша (memory/disk/URLCache), стратегии (cache-aside, write-through, write-back), инвалидация, TTL; офлайн-first, синхронизация и разрешение конфликтов (last-write-wins → CRDT), пре-фетч. Материалы:

- ▶ Как кэшировать данные — теория кэширования — В. Балун RU

## Модуль 5. Клиентская архитектура iOS




D2 · Слои клиентской архитектуры iOS

Слои (UI / Presentation / Domain / Data / Network), repository pattern, DI, модуляризация (SPM-модули, владение командами), feature flags / dynamic delivery, навигация/координаторы. Mobile-specific: app lifecycle, memory pressure, battery, background tasks, фоновые загрузки с чанкингом (напр. 50MB видео). Материалы:

- ▶ Руководство по проектированию мобильных приложений (+советы) — Philipp Lackner EN
- ▶ 4 Common Mistakes in Mobile System Design — Andrey Tech EN
- ▶ Дизайн-система в iOS и Android — Охэхэнные Истории RU

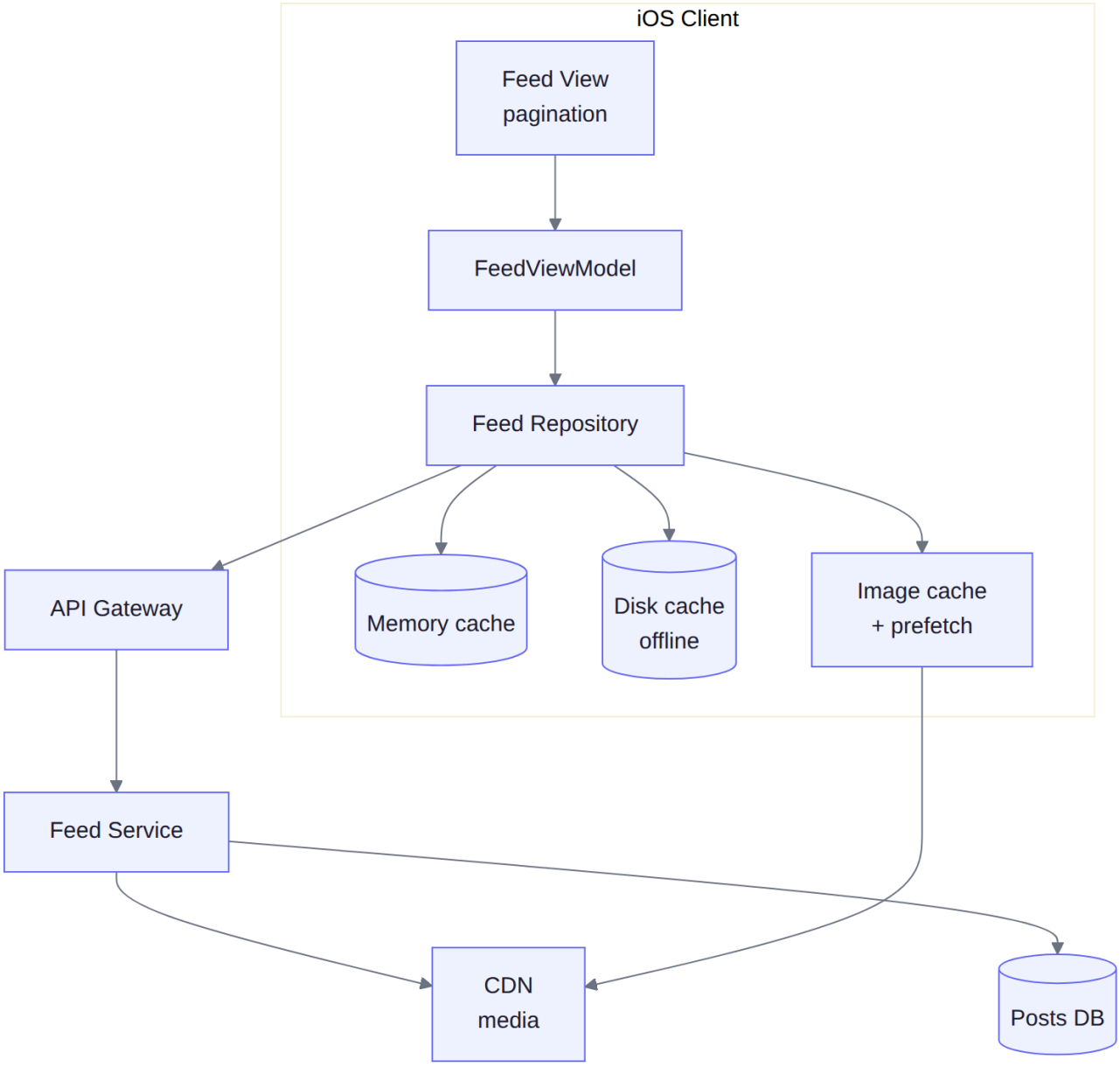
- Medium (доступ через ваш Google-аккаунт): Advanced iOS architecture interview questions Part 4 — Anuj Kumar — dynamic feature delivery, AI/ML в архитектуре, SwiftData в проде, версионирование API, multi-platform (macOS/visionOS/watchOS), App Clips/Widgets/Siri, логирование/мониторинг, CI/CD модульной архитектуры, secure storage.

---

## Модуль 6. РАЗБОР КЕЙСОВ (ядро курса)

Для каждого кейса проходим фреймворк целиком и фиксируем mobile-специфику.

### Кейс 6.1 — Лента (Feed) ENRU ★★★ — полный разбор




D4 · Кейс «Лента» — поток данных

1. Требования. Функциональные: бесконечная лента постов (текст + медиа), пагинация, pull-to-refresh, лайк/коммент, просмотр офлайн ранее загруженного. Нефункциональные: плавный скролл 60 fps, экономия трафика и батареи, работа на low-end устройствах, быстрый «холодный старт» (показать что-то мгновенно из кэша). Скрытые требования, которые надо

вскрыть самому: лимит трафика у пользователя, поведение при потере сети, ранжирование на сервере vs клиенте.

2. API / контракт. `GET /feed?cursor=<opaque>&limit=20` → массив постов + `next_cursor`. Курсорная пагинация, а не offset — offset «плывёт» при вставке новых постов сверху. Медиа отдаём ссылками на CDN с несколькими разрешениями (thumbnail/medium/full), клиент выбирает по размеру экрана и сети. ETag/`If-None-Match` для дешёвого refresh.

3. Архитектура клиента. Repository как единый источник правды: сначала отдаёт данные из локальной БД (мгновенный показ), параллельно идёт сетевой запрос, результат мёржится и обновляет UI. Слои: Network → Repository (+ кэш-политика) → ViewModel → View. Медиа грузит отдельный image-loader с дисковым+memory-кэшем (NSCache + диск), отменой запросов при переиспользовании ячейки и пре-фетчем (`UICollectionViewDataSourcePrefetching`).

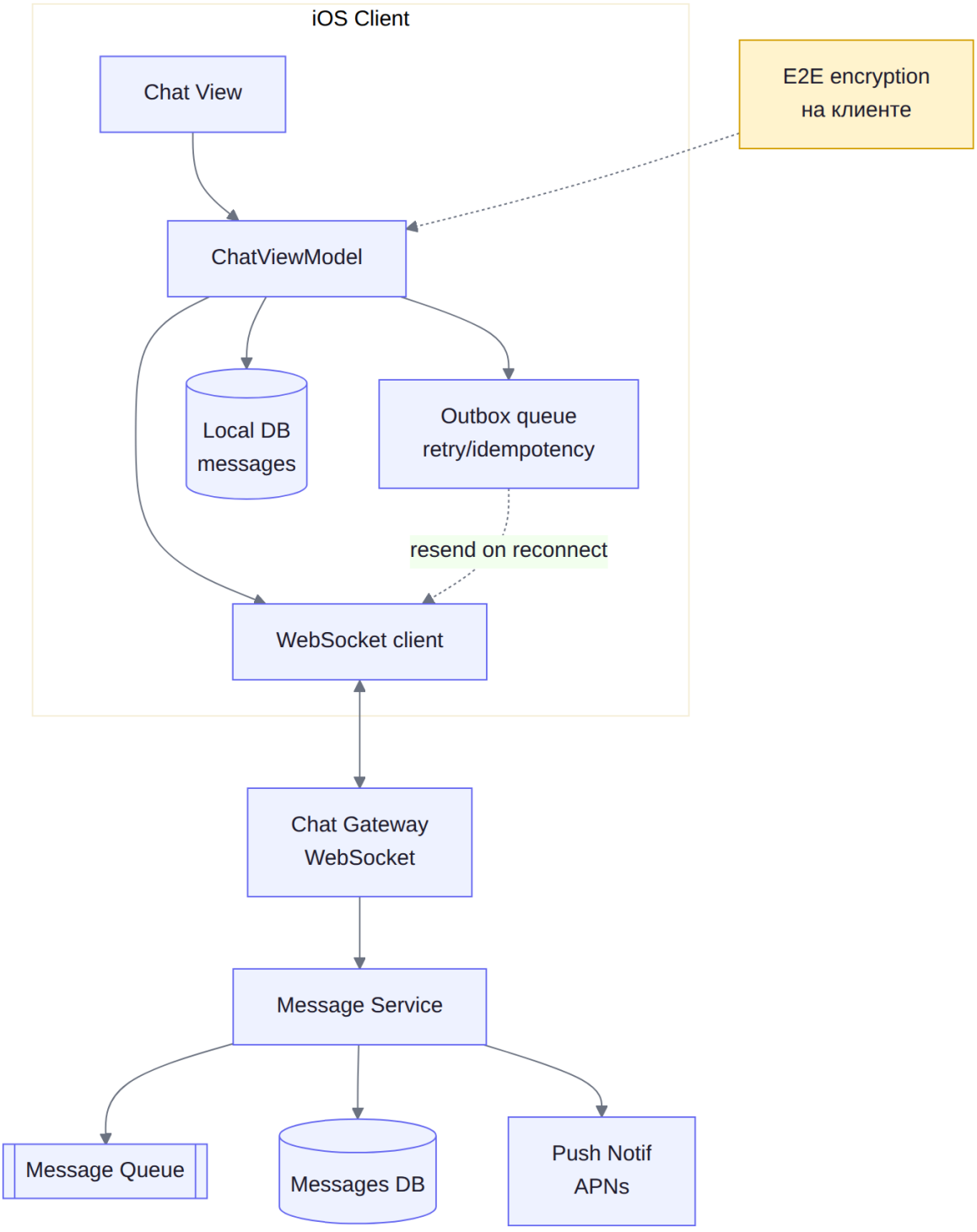
4. Deep dive — производительность скролла. Переиспользование ячеек, асинхронная декодировка изображений вне главного потока, фиксированные/предрасчитанные высоты ячеек, отмена загрузки при быстром скролле, downsampling больших картинок до размера ячейки (иначе memory pressure → Jetsam). Пре-фетч следующей страницы при достижении порога (например, за 5 ячеек до конца).

5. Офлайн и синхронизация. Кэшируем N последних страниц в БД + медиа на диск с LRU-эвикцией по лимиту размера. Лайки офлайн — оптимистичный апдейт + очередь исходящих действий с повтором при появлении сети.

6. Trade-offs (staff-разрез). Push (fan-out на запись, лента предсобрана — быстрый GET, дорогая запись для звёзд с млн подписчиков) vs Pull (fan-out на чтение — дёшево писать, дорого читать) vs гибрид (для «звёзд» pull, для остальных push). На клиенте: агрессивный пре-фетч экономит время, но жжёт трафик/батарею — баланс по типу сети. Где ранжировать: сервер (единая логика, A/B) vs клиент (персонализация офлайн).

- ▶ МОК-интервью: проектируем ленту Twitter — Назаров + Балун RU
- ByteByteGo: Design a News Feed System.

## Кейс 6.2 — Чат / Мессенджер ★★★ — полный разбор




D5 · Кейс «Чат / Мессенджер»

1. Требования. Функциональные: 1:1 и групповые чаты, доставка в реальном времени, статусы (отправлено/доставлено/прочитано), «печатает», медиа-вложения, история и поиск. Нефункциональные: низкая задержка, гарантия доставки и порядка, работа при нестабильной сети, масштаб до ≈ 1 млрд пользователей. Скрытые: E2E-шифрование, мульти-девайс синхронизация, поведение в фоне/push.

2. Транспорт. Постоянный WebSocket для двустороннего обмена (fallback long-poll), APNs для доставки, когда сокет закрыт (фон/убитое приложение). При возврате онлайн — переподключение с backoff и delta-sync по курсору.

3. Модель доставки и порядок. Каждое сообщение получает клиентский `clientMessageId` (UUID) для идемпотентности (ретраи не плодят дубли) и серверный монотонный `seq`/timestamp для порядка. Порядок определяет сервер, а не часы устройства (они расходятся).

4. Архитектура клиента и очередь исходящих. Локальная БД (SQLite/Core Data/GRDB) — источник правды для UI. Отправка: сообщение сразу пишется в БД со статусом `pending` (оптимистичный UI), затем фоновый «отправитель» берёт из очереди и шлёт в сокет; при ACK → `sent`, при ошибке → ретрай с backoff. Входящие дедуплицируются по id и вставляются по `seq`.

5. Синхронизация и мульти-девайс. При подключении клиент посылает последний известный `seq` по каждому чату → сервер отдаёт только дельту. Непрочитанные и статусы хранятся на сервере, чтобы совпадали на всех устройствах.

6. Trade-offs (staff-разрез). at-least-once + дедупликация по id даёт эффект exactly-once без дорогих распределённых транзакций. E2E (Signal-протокол) усиливает приватность, но ломает серверный поиск/антиспам и усложняет мульти-девайс (раздача ключей). Постоянные WebSocket-соединения дороги по памяти на сервере — баланс с push-only для неактивных пользователей.

- ▶ Мессенджер на 1 млрд пользователей RU
- ByteByteGo: Design a Chat System.

### Кейс 6.3 — Музыкальный / медиа-стриминг ★★★

Стриминг, буферизация, офлайн-загрузки, плейлисты, фоновое воспроизведение, кодеки.

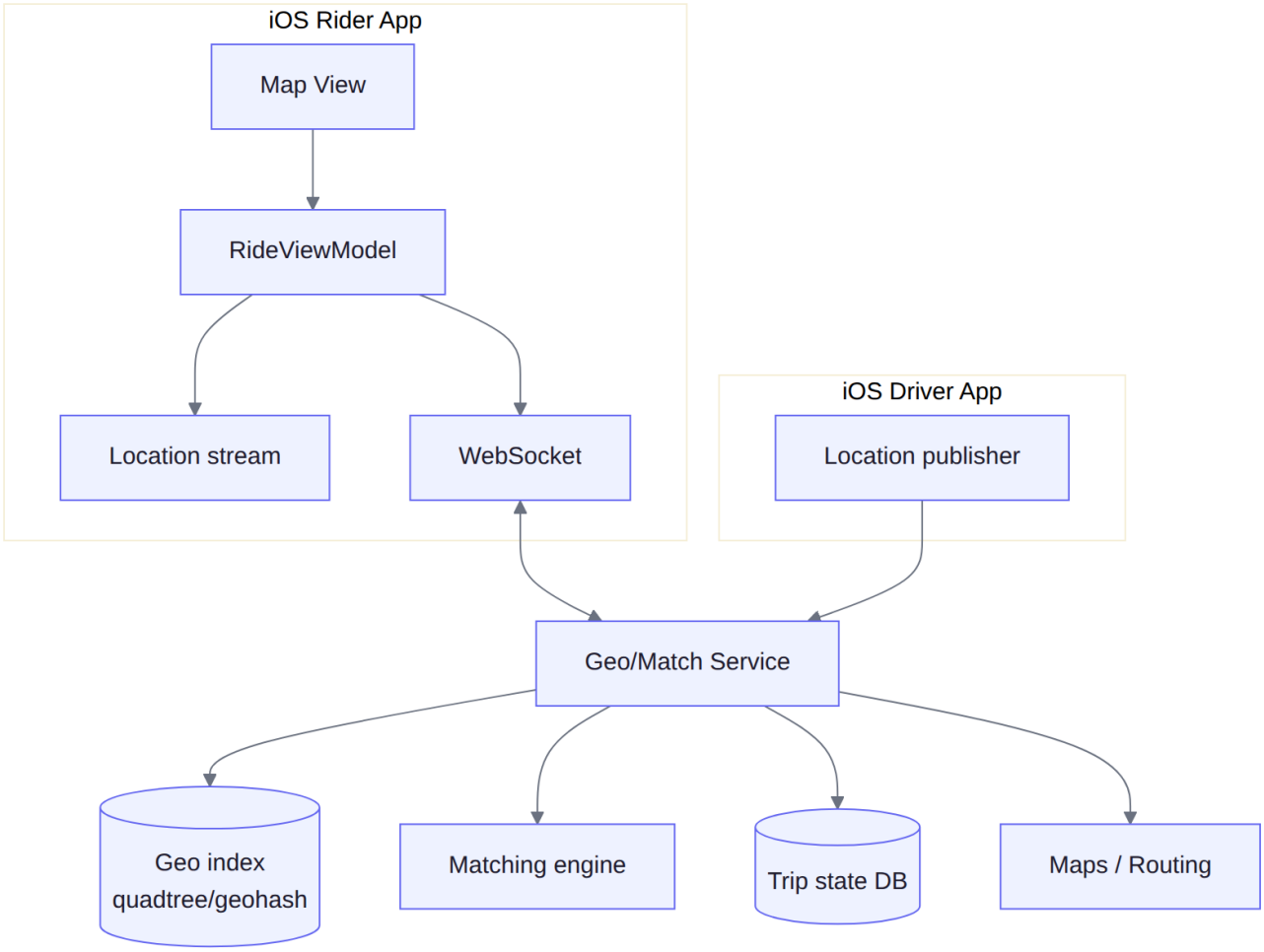
- ▶ System Design: проектируем Yandex Music — { между скобок } RU

### Кейс 6.4 — Видеохостинг (YouTube-like) ★★★

Загрузка/транскодинг, CDN, адаптивный битрейт, лента рекомендаций.

- ▶ Проектируем YouTube — System Design Notes RU

### Кейс 6.5 — Ride-sharing (Uber-like) ★★★ — полный разбор




D6 · Кейс «Ride-sharing (Uber)»

1. Требования. Функциональные: запрос поездки, матчинг с ближайшим водителем, real-time трекинг машины на карте, расчёт ETA/цены, состояния поездки (поиск → принята → в пути → завершена), оплата. Нефункциональные: низкая задержка матчинга, точность гео, отказоустойчивость при потере сети в движении, экономия батареи (постоянный GPS — главный потребитель). Скрытые: surge-ценообразование, отмена с обеих сторон, поведение в туннеле/без сигнала.

2. API / контракт. Гео-апдейты водителя — частый, но компактный поток (батчинг локаций, бинарный формат). `POST /rides` (идемпотентно по request-id), затем подписка на статус поездки. Клиент пассажира получает позицию водителя через push по сокету; интерполяция движения между апдейтами для плавности.

3. Архитектура клиента. Конечный автомат поездки (explicit state machine) — единственный источник истины состояния, переходы только по событиям с сервера. Слой локации абстрагирован: разная частота GPS по состоянию (редко в ожидании, часто в поездке) для экономии батареи. Карта рендерит позиции через слой аннотаций с анимацией перемещения.

4. Deep dive — гео и матчинг. На сервере геоиндекс (geohash / S2 / quadtree) для запроса «ближайшие водители в радиусе». Матчинг учитывает дистанцию, ETA, рейтинг, направление. На клиенте: throttling гео-апдейтов, отправка только при значимом смещении, сжатие трека. ETA считается на сервере (актуальный трафик), клиент показывает кэш до обновления.

5. Офлайн и надёжность. Состояние поездки кэшируется локально: при кратком пропадании сети UI остаётся консистентным, апдейты буферизуются и досылаются. Запрос поездки идемпотентен (request-id) — повтор при таймауте не создаёт две поездки. Оплата — отдельный идемпотентный шаг с подтверждением.

6. Trade-offs (staff-разрез). Частота GPS: точность трекинга против расхода батареи — адаптивная частота по состоянию и скорости. Матчинг на сервере (глобальная оптимизация, surge) против гибридного (клиент предлагает кандидатов). Геоиндекс: точность против стоимости пересчёта при высокой плотности апдейтов. Push через сокет (мгновенно, дорого держать) против polling (проще, выше задержка).

- ▶ Секреты успешного SD Interview: Design Uber — IT Enduro RU
- ByteByteGo: Design Nearby Friends / Proximity Service.

### Кейс 6.6 — Финтех / банковское моб. приложение ★★★

Безопасность, secure storage, авторизация, идемпотентность платежей, офлайн-ограничения.

- ▶ System Design в Revolut на $9000 (Android/iOS) — Фабрика Офферов RU

### Кейс 6.7 — Дизайн-система / SDK ★★★

Версионирование, обратная совместимость, распространение (SPM), API-контракты — частый вопрос для staff (Jacob).

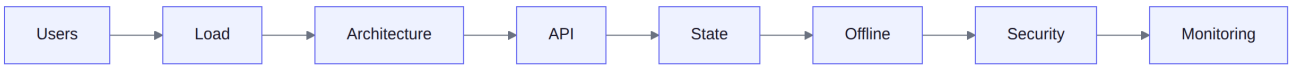
- ▶ Дизайн системы: Google Material You — Disarto RU

## Модуль 7. Мок-интервью (репетиция)

Смотрите чужие моки, затем проводите свои с партнёром.

- ▶ Mock-собеседование по System Design (ex-Team Lead Яндекс) — balun.courses RU
- ▶ Публичное собеседование по System Design — { между скобок } RU
- ▶ Проводим и проходим Mobile System Design-интервью — Слава Слуцкер, Mobius RU
- ▶ Разбор Android System Design за 450k — IT Crew RU (методология применима к iOS)

## Чек-лист на интервью (mental checklist)




D3 · Mental checklist на интервью

Users → Load → Architecture → API → State → Offline → Security → Monitoring. Mobile-болевые точки, которые отличают сильного кандидата: нет сигнала? поворот экрана в середине запроса? фоновый режим и session drift? memory pressure? батарея? (System Design Handbook).

## Недельный план (5–6 недель)

| Неделя | Модули | Кейсы | Мок |
|---|---|---|---|
| 1 | M0–M2 (уровни, фреймворк, требования) | — | — |

| Неделя | Модули | Кейсы | Мок |
|---|---|---|---|
| 2 | M3–M4 (концепты, кэш/офлайн) | 6.1 Лента | — |
| 3 | M5 (клиентская архитектура iOS) | 6.2 Чат | 1 |
| 4 | M6 кейсы | 6.3, 6.4 | 1 |
| 5 | M6 кейсы | 6.5, 6.6, 6.7 | 2 |
| 6 | M7 мок-марафон + слабые места | повтор 2 кейсов | 3+ |

## Рекомендованные ресурсы (детали — в `01_resources_review.md`)

Бесплатный старт: weeeBox/mobile-system-design → themobileinterview.com → RU-фреймворк levabond → FaangTalk (YouTube). Платно (по желанию): «Building Mobile Apps at Scale» (G. Orosz) + книга Tjeerd in 't Veen + подписка Educative «Grokking the Mobile System Design Interview».

## Критерии готовности

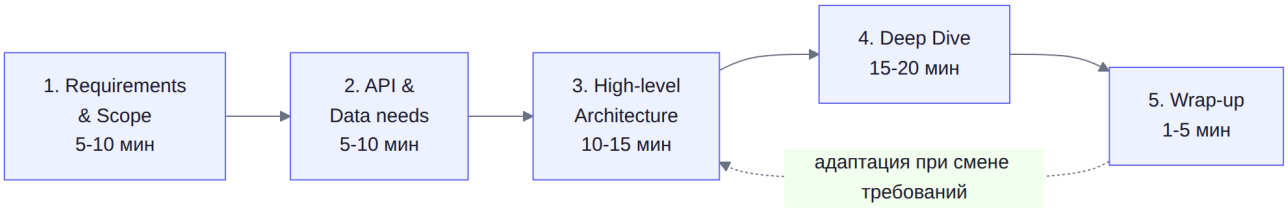
- [ ] Прохожу фреймворк (5 шагов) за ≤45 мин на любом кейсе.
- [ ] Проактивно вскрываю скрытые требования и подсвечиваю red flags.
- [ ] Свободно обсуждаю кэш/офлайн/конфликты и модуляризацию.
- [ ] Аргументирую трейд-оффы на уровне команды/орга (staff-разрез).
- [ ] Прошёл ≥4 мок-интервью по разным кейсам.

РАЗДЕЛ 4

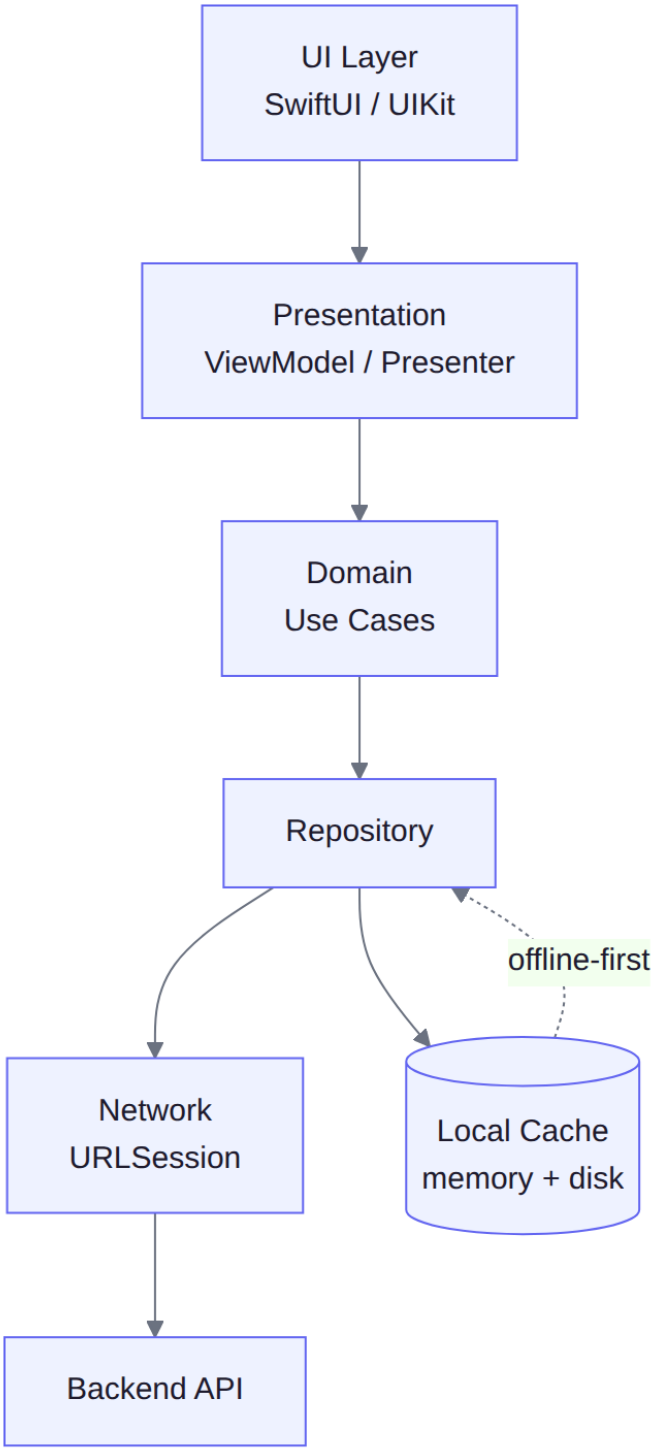
# Диаграммы

Все 10 схем в одном месте — для повторения перед собеседованием

## D1 · Фреймворк System Design интервью (5 шагов)

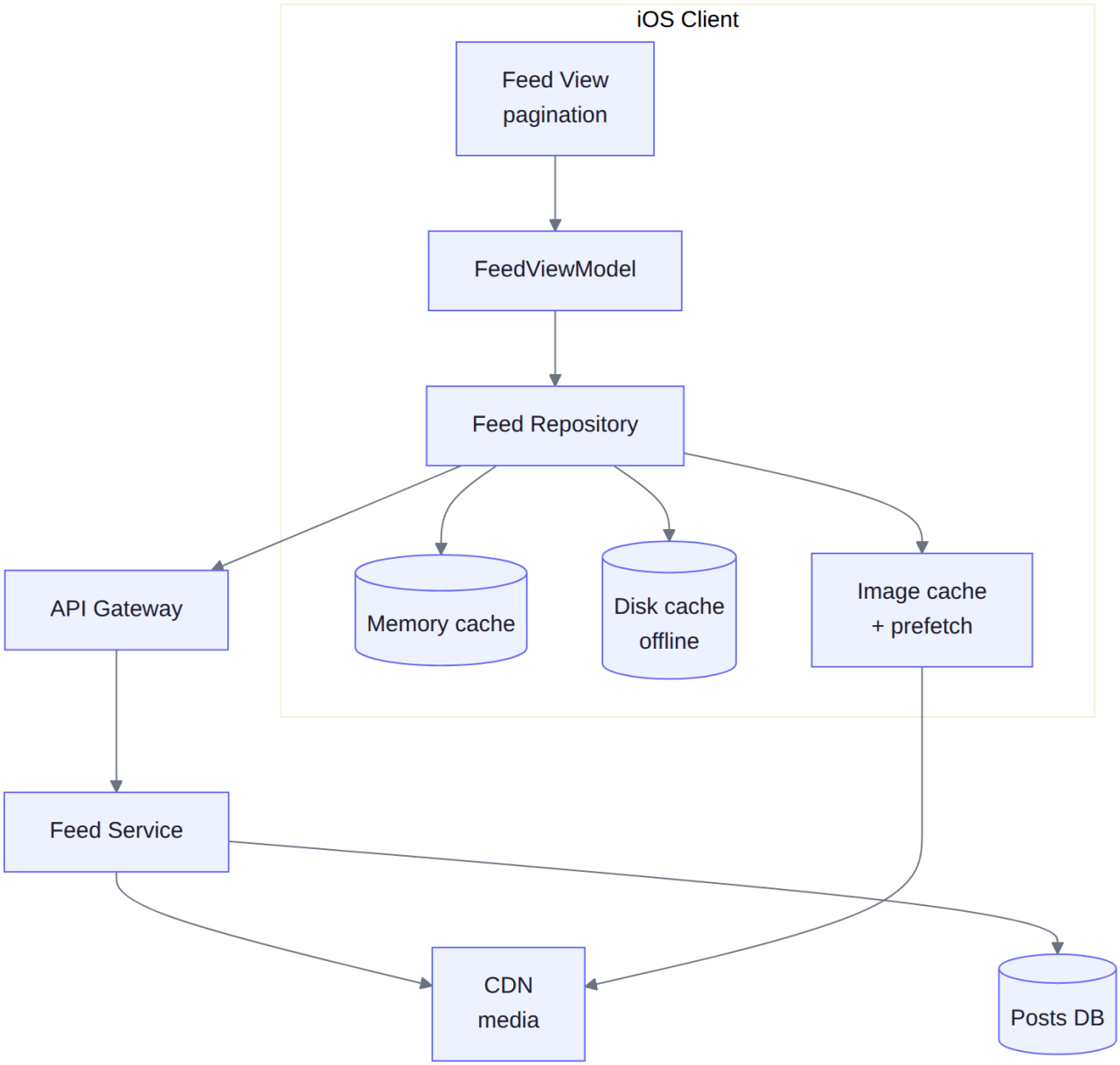



## D2 · Слои клиентской архитектуры iOS

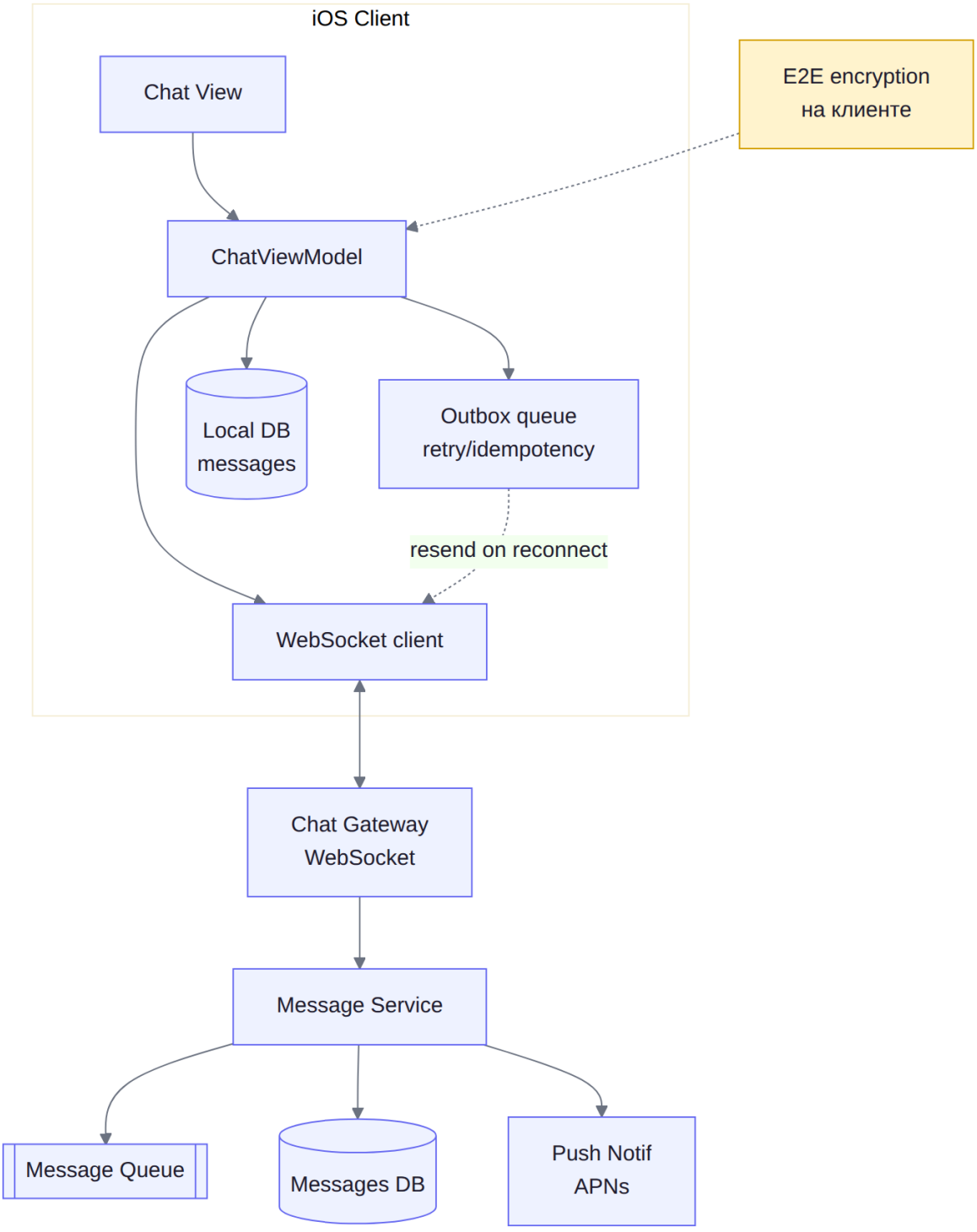



## D3 · Mental checklist на интервью

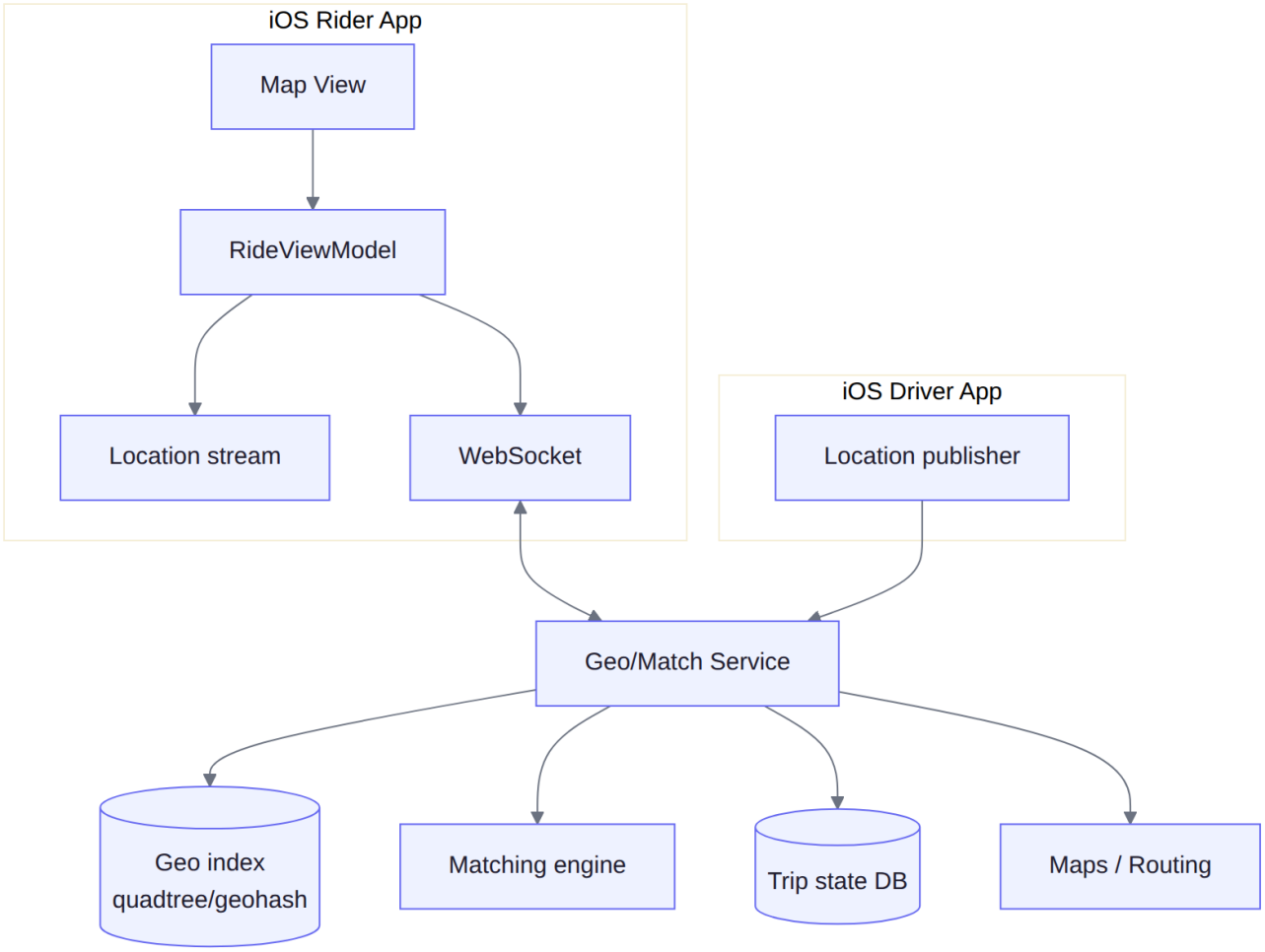

## D4 · Кейс «Лента» (Feed) — поток данных



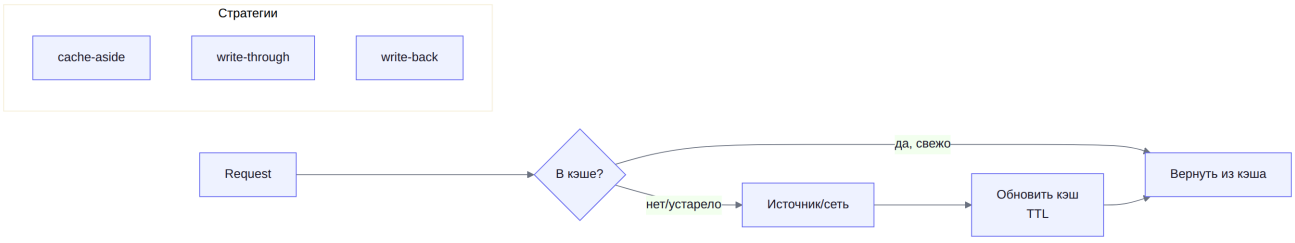

## D5 · Кейс «Чат / Мессенджер» — архитектура



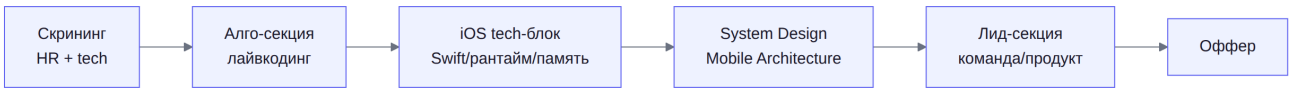

## D6 · Кейс «Ride-sharing (Uber)» — real-time tracking



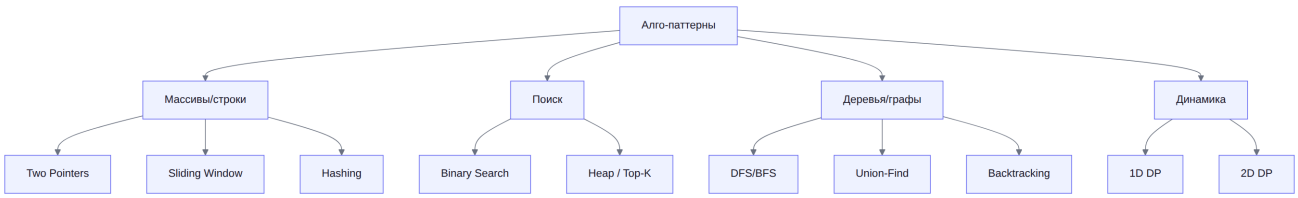

## D7 · Кэширование: стратегии



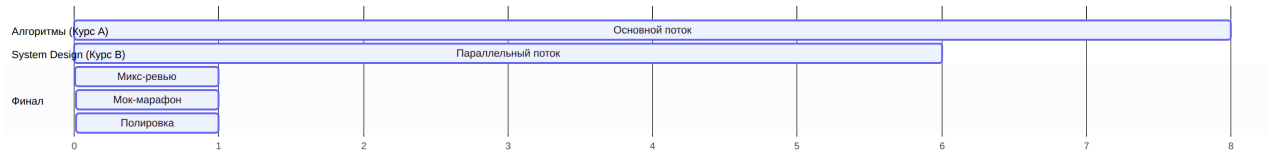

## D8 · Воронка собеседования в РФ-бигтех (техлид iOS)




## D9 · Дерево паттернов алгоритмов



# D10 · Общий план подготовки (12 недель)



РАЗДЕЛ 5

# Видео-плейлисты

34 ролика по темам с уровнем сложности, языком и тайм-кодами ключевых видео

> Все ролики из вашей подборки проверены по названию и каналу, сгруппированы по темам, снабжены аннотациями и пометками уровня (★ база · ★★ middle+ · ★★★ senior/lead) и языка (RU/EN).

> Рекомендуемый порядок просмотра указан внутри каждого блока. Видео используются как ссылки с атрибуцией — смотрите на YouTube.

## ЧАСТЬ B — System Design (31 видео)

### B0. Старт: фреймворк и общая база ★

| # | Видео | Канал | EN |
|---|---|---|---|
| 1 | System Design RoadMap — Твоя база | Владимир Невзоров | RU |
| 2 | Основы системного дизайна за 30 минут | Eugene Suleimanov | RU |
| 3 | Что нужно знать новичку о системном дизайне | Максим Добрынин (ex. Jetbulb) | RU |
| 4 | 20 концептов System Design за 10 минут | AlexTech IO | RU |
| 5 | 8 Most Important System Design Concepts | ByteByteGo | EN |

### B1. Методология интервью: как проходить и не заваливать ★★

| # | Видео | Канал | EN |
|---|---|---|---|
| 6 | Почему все проваливают собеседование по System Design? | Владимир Балун | RU |
| 7 | Как пройти System Design Интервью? 4 этапа, 2 качества кандидата | Владимир Невзоров | RU |
| 8 | Как подготовиться и пройти System Design Interview — А. Поломодов | ArchDays | RU |
| 9 | Интервью по System Design — А. Поломодов (Тинькофф) | ArchDays | RU |

| # | Видео | Канал | EN |
|---|---|---|---|
| 10 | System Design-интервью для практиков — Д. Волыхин | JUG ru | RU |
| 11 | Первая часть гайда по System Design (Middle+) | Антон Назаров (Осознанная Меркантильность) | RU |
| 12 | System Design. Как построить распределённую систему — В. Маслов | JUG ru | RU |

## B2. ★ Mobile / iOS System Design (ключевой блок для вас) ★★★

| # | Видео | Канал | EN |
|---|---|---|---|
| 13 | Как проходит интервью Mobile System Design & PM | Yandex for Mobile | RU |
| 14 | iOS System Design: чем уникален мобильный систем-дизайн (#FaangTalk 62) | FaangTalk | RU |
| 15 | Проводим и проходим Mobile System Design-интервью — Слава Слуцкер | Mobius | RU |
| 16 | 4 Common Mistakes in Mobile System Design Interviews | Andrey Tech | EN |
| 17 | Руководство по проектированию мобильных приложений (+ советы для собесов) | Philipp Lackner | EN |
| 18 | Дизайн-система в iOS и Android | Охэхэнные Истории | RU |
| 19 | Дизайн системы: Google Material You (Material Design 3) | Disarto | RU |

## B3. Разбор кейсов и мок-интервью ★★★

| # | Видео | Канал | Кейс |
|---|---|---|---|
| 20 | Mock-собеседование по System Design (ex-Team Lead Яндекс) | balun.courses | Общий мок RU |
| 21 | Публичное собеседование по System Design — Вахмистров/Иванов | { между скобок } | Мок RU |

| # | Видео | Канал | Кейс |
|---|---|---|---|
| 22 | System Design: проектируем Yandex Music — Антонов/Скобелев | { между скобок } | Музыкальный сервис RU |
| 23 | МОК-интервью: проектируем ленту Twitter | Назаров + Балун | Лента/Feed RU |
| 24 | Проектируем YouTube — Введение в System Design | System Design Notes | Видеохостинг RU |
| 25 | Мессенджер на 1 млрд пользователей | Распределённые системы и SD | Чат/мессенджер RU |
| 26 | Секреты успешного SD Interview: Design Uber | IT Enduro | Ride-sharing RU |
| 27 | Как я прошёл System Design интервью (оффер 550к) | Булат Якубов | Личный опыт RU |
| 28 | System Design в Revolut на $9000 (Android/iOS) | Фабрика Офферов | Финтех-моб. RU |
| 29 | Разбор Android System Design за 450k | IT Crew | Моб. кейс RU |

### B4. Углублённые темы ★★★

| # | Видео | Канал | Тема |
|---|---|---|---|
| 30 | Как кэшировать данные — теория кэширования | Владимир Балун | Кэширование RU |
| 31 | System Design. Разбор «Высоконагруженные приложения», гл.1 | Владимир Невзоров | DDIA RU |

## ЧАСТЬ A — Алгоритмы (3 видео + плейлист)

| # | Видео | Канал | EN |
|---|---|---|---|
| 1 | Сложность и модели вычислений. Анализ учётных стоимостей — М. А. Бабенко | Yandex for ML | RU ★★ |
| → | Плейлист целиком: Курс «Алгоритмы и структуры данных поиска» (12 видео) | Yandex for ML | RU |
| 2 | Алгоритмы и структуры данных. Введение. Массивы | VK Team | RU ★ |
| 3 | Dynamic Programming — Learn to Solve Algorithmic Problems | freeCodeCamp.org | EN ★★ |

> Эти три ролика — фундамент: оценка сложности (Бабенко), базовые структуры (VK), и отдельный глубокий разбор динамического программирования (freeCodeCamp). Расширить можно ресурсами из `01_resources_review.md` (Тренировки Яндекса, NeetCode, algocode.io).

---

## Заметка по легальности и компоновке

Материалы используются как кураторские ссылки с указанием автора и канала. Видео не перезаливаются и не нарезаются в отдельные файлы — в учебнике даются тезисы/тайм-коды и ссылка на оригинал. Это корректный и легальный способ реюза публичного контента.

РАЗДЕЛ 6

# Обзор ресурсов

26 курсов и источников с вердиктами, цены и форматы, каналы, репозитории, блоги

> Аудитория: опытный iOS-инженер / техлид, уровень Senior+.

> Цель: Яндекс, Avito, Ozon, VK — алгоритмическая секция + Mobile/iOS System Design.

> Языки: RU + EN.

> Дата обзора: июнь 2026.

## A. Алгоритмы и структуры данных

### Сводная таблица

| # | Название | Провайдер | Ссылка | Цена | Язык | Формат | Уровень |
|---|---|---|---|---|---|---|---|
| 1 | Тренировки по алгоритмам (сезон 7.0+) | Яндекс | yandex.ru | Бесплатно | RU | Лекции + задачи + разборы (live) | Middle–Senior |
| 2 | Подготовка к алгоритмическому собеседованию | Яндекс Практикум | practicum.yandex.com | Бесплатно | RU | Текст + задачи (без ментора) | Any |
| 3 | Алгоритмы и структуры данных | Яндекс Практикум | practicum.yandex.com | Частично бесплатно (1 модуль), далее платно (~≈30 000–40 000 ₽) | RU | Текст + задачи + код-ревью наставника | Middle |
| 4 | Подготовка к алгоритмическому собеседованию | balun.courses | balun.courses | Платно ( интенсивы по темам, покупаются отдельно или комплектом; есть бесплатный курс по СД) | RU | Видео + задачи + самопроверка | Middle–Senior |

| # | Название | Провайдер | Ссылка | Цена | Язык | Формат | Уровень |
|---|---|---|---|---|---|---|---|
| 5 | Алгоритмы и структуры данных | algocode.io | algocode.io | Подписка ~2 650 ₽/мес или ~1 855 ₽/мес при оплате на 300 дней | RU | Видео + задачи + мок-интервью + чат | Middle–Senior |
| 6 | Алгоритмы с нуля | Влад Тен (Boosty) | boosty.to/vladtenishe | ~11 000 ₽ за месяц (доступ навсегда после оплаты) | RU | Видео (50+ ч) + 100+ задач + Live Q&A | Middle–Senior |
| 7 | Алгоритмы: теория и практика. Методы | Stepik / Computer Science Center | stepik.org/course/217 | Бесплатно | RU | Видео + задачи | Junior–Middle |
| 8 | Алгоритмы: теория и практика. Структуры данных | Stepik / CS Center | stepik.org/course/1547 | Бесплатно | RU | Видео + задачи | Junior–Middle |
| 9 | LeetCode (задачник) + LeetCode Crash Course | LeetCode | leetcode.com | Бесплатно (большинство задач) / Premium ~$35/мес | EN | Интерактивные задачи + курсы | Any |
| 10 | NeetCode 150 / Roadmap | NeetCode | neetcode.io | Бесплатно (150 задач + видео) / Pro платно | EN | Видео-разборы + задачи | Middle–Senior |
| 11 | Grokking the Coding Interview: Patterns | Educative.io | educative.io | Подписка ~$14.9/мес или единоразово ~$80 | EN | Текст + интерактивный код в браузере | Middle–Senior |
| 12 | AlgoMaster Newsletter (15/50 LeetCode patterns) | AlgoMaster | blog.algomaster.io | Бесплатно (статьи) | EN | Текст + задачи | Senior |

| # | Название | Провайдер | Ссылка | Цена | Язык | Формат | Уровень |
|---|---|---|---|---|---|---|---|
| 13 | BackToBackSWE (YouTube канал) | BackToBackSWE | youtube.com/@BackToBackSWE | Бесплатно | EN | Видео-разборы | Middle–Senior |

## Краткие разборы

### 1. Тренировки по алгоритмам от Яндекса

Провайдер: Яндекс (yandex.ru/yaintern/training/algorithm-training) Стоимость: Бесплатно Формат: Онлайн-интенсив ~5 недель; лекции в прямом эфире (записи остаются), 4 контеста по 10 задач, разборы, вебинары с рекрутерами. Задачи сдаются в CodeRun. Темп: Сезонный (новый раз в год, записи предыдущих сезонов доступны публично на YouTube). Темы сезона 7.0: жадные алгоритмы, динамическое программирование, дерево отрезков, битовые операции, коды Хэмминга, B-деревья, двусвязные списки. Плюсы:

- Задачи максимально близки к реальным задачам на собеседованиях в Яндекс.
- 300 лучших участников автоматически проходят этап отбора на стажировку/в штат.
- Записи лекций от ведущих специалистов (преподаватель сезона 7.0 — М. Густокашин, тренер чемпионов ICPC).
- Готовые разборы задач на YouTube-канале Яндекса.

Минусы:

- Сезонный формат (нет гибкого доступа в любое время).
- Ориентирован на Middle и выше, без плавного входа для новичков.
- Задачи ближе к уровню Яндекса, не охватывают весь спектр РФ-бигтеха.

Вердикт: ★★★★★ — обязательный ресурс для тех, кто целится именно в Яндекс. Для всех остальных — отличный бесплатный тренажёр уровня Medium/Hard.

### 2. Подготовка к алгоритмическому собеседованию (бесплатный курс)

Провайдер: Яндекс Практикум (practicum.yandex.com/algorithms-interview) Стоимость: Полностью бесплатно Формат: 5 блоков; теория, тесты, задачи (~10 часов теории, практика зависит от уровня). Нет дедлайнов, нет ментора, нет ревью кода. Темы: структура собеседований, базовые АиСД, реальные задачи с интервью. Код — C++/Python, сдача задач — также на Java/JS. Плюсы:

- Полностью бесплатно, навсегда.
- Хорошая точка старта для оценки своего уровня и понимания структуры алго-секции.
- Реальные задачи с прошедших собеседований.

Минусы:

- Поверхностный охват теории (не заменяет полноценный курс).
- Нет обратной связи от наставника.

Вердикт: ★★★★☆ — отличный бесплатный «чекап» перед началом интенсивной подготовки.

### 3. Алгоритмы и структуры данных (полный курс)

Провайдер: Яндекс Практикум (practicum.yandex.com/algorithms) Стоимость: 1 модуль бесплатно (7 ч), платная часть — ~30 000–40 000 ₽ Формат: 4 месяца, ~10 ч/нед; 23 темы, 27 часов, 100+ задач с код-ревью наставника, пробное алгоритмическое собеседование. Темы: двухнедельные спринты по классическим АиСД (сортировки, деревья, графы, ДП и т.д.). Плюсы:

- Систематический подход с наставником и код-ревью.
- Пробное собеседование в финале.
- Работодатель может оплатить.

Минусы:

- Дорого (при самостоятельной оплате).
- Темп жёстко привязан к двухнедельным спринтам.
- Для опытного инженера часть материала — избыточна.

Вердикт: ★★★☆☆ — подходит тем, кому нужна структура и поддержка наставника, а не самостоятельная работа. Для Senior-специалиста с опытом — вероятно, избыточно по цене.

---

### 4. Подготовка к алгоритмическому собеседованию — balun.courses

Провайдер: Владимир Балун, ex-TeamLead Яндекс (balun.courses/courses/algorithmic_interview) Стоимость: Есть бесплатный курс по структурам данных (3 ч); платные интенсивы по темам покупаются отдельно или комплектом (скидки 10–30%); рассрочка доступна. Формат: Видеозаписи; по каждой теме: быстрая теория → разбор типовых задач easy/medium → 5–7 задач для самопрактики → эталонные решения (на Go, переводимые через AI). Общий чат, вопросы по урокам. Темы (доступны или анонсированы): два указателя, матрицы, хеш-таблицы, битовые манипуляции, префиксные суммы, бинарный поиск, сортировки, DFS/BFS, плавающие окна, стеки/очереди, связные списки, деревья, кучи. Плюсы:

- Курс полностью «заточен» под РФ-бигтех (Яндекс, VK, Avito, Ozon) — задачи реальных секций.
- Автор прошёл путь от «провала всех алгосекций» до успешных офферов в BigTech.
- Компактные интенсивы по темам (1–2 ч + практика) — удобно для повторения пробелов.
- Бесплатный курс по СД — хорошая точка входа.

Минусы:

- Курс на Go (хотя логика решений универсальна).
- Небольшой объём задач на тему (5–7 штук) — достаточно для понимания паттерна, но мало для глубокой нарешки.
- Нет видео-разборов каждой задачи (только эталонное решение).

Вердикт: ★★★★★ — лучший российский курс, заточенный под реальные задачи РФ-бигтеха. Хорошо сочетается с LeetCode для практики в объёме.

---

### 5. Алгоритмы и структуры данных — algocode.io

Провайдер: algocode.io (algocode.io/algorithms) Стоимость: Подписка ~2 650 ₽/мес или ~1 855 ₽/мес при оплате на 300 дней. Включает все курсы платформы (АиСД, графы, System Design, Go-собеседование). Формат: 12 тем, 100+ задач, 80+ видео-разборов, еженедельные мок-интервью с партнёром, чат с куратором. Задачи в IDE, метки компаний (Яндекс, Avito, VK, Ozon, Сбер). Темы: два указателя, хеш-таблицы, массивы/матрицы, скользящее окно,

бинарный поиск, структуры данных, backtracking, односвязный список, деревья, графы.
Плюсы:

- Задачи помечены компаниями, что удобно для целевой подготовки.
- Еженедельные мок-интервью с партнёром — уникальная функция в RU-сегменте.
- Подписка даёт доступ к System Design курсу — два в одном.
- Многоязычная поддержка (Python, Java, Go, JS, C++, C#).

Минусы:

- Подписочная модель (нет разового доступа навсегда).
- Нет разбора по теме «динамическое программирование» в базовом курсе.
- Некоторые пользователи жалуются на недостаточную глубину сложных тем.

Вердикт: ★★★★☆ — сильная платформа для подготовки к РФ-бигтеху с акцентом на практику и мок-интервью.

---

### 6. Алгоритмы с нуля — Влад Тен

Провайдер: Влад Тен (Boosty: boosty.to/vladtenishe) Стоимость: ~11 000 ₽ за месяц (доступ к материалам навсегда после оплаты) Формат: 50+ часов видео, 100+ задач, Live Q&A-сессии, Telegram-канал. Паттерны: Two Pointers, Sliding Window, Monotonic Stack, Greedy. Структуры данных: Linked List, Stack/Queue, Binary Tree, Heap, LRU, Graphs, Dynamic Programming. Плюсы:

- Очень большой объём видео-материала — один из самых насыщенных RU-курсов.
- Влад Тен активен как ментор и публично разбирает LeetCode-задачи (публичные видео на YouTube).
- Охватывает DP и продвинутые темы.
- Бессрочный доступ после оплаты.

Минусы:

- Цена высокая относительно конкурентов.
- Платформа Boosty — менее удобна, чем специализированные образовательные платформы.
- Курс без встроенного тренажёра задач (задачи с LeetCode).

Вердикт: ★★★★☆ — хороший выбор для тех, кто учится через видео и хочет большой объём контента с разбором паттернов. Конкурент balun.courses в чуть более видео-ориентированном формате.

---

### 7 & 8. Алгоритмы: теория и практика — Stepik / CS Center

Провайдер: Computer Science Center (stepik.org/course/217 и stepik.org/course/1547) Стоимость: Бесплатно Формат: Академические видео-лекции + практические задачи на C++/Java/Python/Haskell/Octave. Курс «Методы»: жадные алгоритмы, «разделяй и властвуй», динамическое программирование — с математическими доказательствами. Курс «Структуры данных»: массивы, списки, очереди, стеки, кучи, DSU, хеш-таблицы, сбалансированные деревья. Плюсы:

- Академически строгий подход с доказательствами — помогает понять «почему», а не только «как».
- Полностью бесплатно.
- Авторы — сотрудники CS Center/ВШЭ.

Минусы:

- Рассчитан на Junior–Middle или студентов; для Senior это «повторение базы».
- Задачи менее ориентированы на формат BigTech-интервью.
- Нет разборов задач с реальных собеседований.

Вердикт: ★★★★☆ — отличный бесплатный академический фундамент. Для Senior-инженера полезен как быстрое повторение теории (особенно «Методы»), а не как основной тренажёр.

---

### 9. LeetCode (задачник + Crash Course)

Провайдер: LeetCode (leetcode.com) Стоимость: Большинство задач бесплатно; Premium ~$35/мес (company-specific задачи, подсказки, mock interviews). Формат: 3000+ задач; встроенный IDE; структурированные курсы (Crash Course по паттернам); company-таги (Яндекс, VK и т.д. через Premium). Плюсы:

- Де-факто стандарт подготовки. Без него подготовка к интервью в 2026 немыслима.
- Встроенный курс «LeetCode 75» + «Top Interview 150» — отличные кюрейтед-листы.
- Contests для поддержания формы.

Минусы:

- Отдельный ресурс, нужна своя стратегия (риск «бездумной нарешки»).
- Premium-фичи необязательны, если использовать внешние листы (NeetCode 150).
- Интерфейс и пояснения не всегда удобны без внешних видео-разборов.

Вердикт: ★★★★★ — must-have. Основная площадка для практики; комбинируется с любым теоретическим ресурсом.

---

### 10. NeetCode 150 / Roadmap

Провайдер: NeetCode (neetcode.io) Стоимость: Бесплатно (150 задач + видео-разборы); Pro — платно (дополнительные курсы). Формат: 16 категорий, 150 задач с видео-разборами. Наглядный roadmap (neetcode.io/roadmap). Плюсы:

- Лучший бесплатный структурированный план подготовки по алгоритмам.
- Каждая задача — видео с объяснением мышления, а не только кода.
- 150 задач покрывают ~95% паттернов, встречающихся на интервью.
- Регулярно обновляется.

Минусы:

- Нет System Design.
- Нет компании-специфичных задач (нужен LeetCode Premium).
- Оптимально для знания паттернов, не для глубины в сложных темах (Segment Tree, Trie и т.д.).

Вердикт: ★★★★★ — лучший бесплатный старт/структурирование по EN. В идеале: NeetCode 150 как скелет, далее тематическая нарешка на LeetCode.

---

### 11. Grokking the Coding Interview: Patterns for Coding Questions

Провайдер: Educative.io (educative.io) Стоимость: Подписка ~$14.9/мес (все курсы Educative) или ~$80 за курс единоразово; есть бесплатный trial (25+ уроков). Формат: Текстовый интерактивный курс, 28 паттернов, 215 задач, 17 мок-интервью, coding playground прямо в

браузере. Языки: Python, Java, JS, Go, C++. Темы: Sliding Window, Two Pointers, Fast & Slow Pointers, Merge Intervals, Cyclic Sort, In-place Reversal, BFS/DFS, Two Heaps, Subsets, Backtracking, DP и др. Плюсы:

- Паттерн-ориентированный подход — лучшая система для понимания «типа задачи».
- Интерактивная среда без установки IDE.
- 28 паттернов охватывают практически всё, что встречается на FAANG и РФ-бигтех.
- Регулярно обновляется (версия 2026 активна).

Минусы:

- Только текст, нет видео.
- Подписочная модель.
- Дороговато при единоразовой оплате.

Вердикт: ★★★★☆ — лучший EN-курс для систематизации паттернов. Пара с NeetCode: Grokking даёт паттерновую базу, NeetCode — видео-разборы.

---

### 12. AlgoMaster Newsletter (15/50 LeetCode Patterns)

Провайдер: AlgoMaster (blog.algomaster.io) Стоимость: Бесплатно (статьи/рассылка) Формат: Серия статей и рассылки о паттернах LeetCode; 15 ключевых паттернов (Two Pointers, Sliding Window, BFS/DFS, DP, Backtracking и т.д.) с объяснением и примерами задач. Плюсы:

- Быстрый обзор ключевых паттернов — хорошо для повторения перед интервью.
- Бесплатно.
- Автор решил 1500+ задач — практический взгляд.

Минусы:

- Не заменяет полноценный курс; только справочник паттернов.

Вердикт: ★★★★☆ — отличный бесплатный cheat sheet для Senior-инженера, чтобы быстро освежить паттерны.

---

### 13. BackToBackSWE (YouTube-канал)

Провайдер: BackToBackSWE (youtube.com/@BackToBackSWE) Стоимость: Бесплатно (YouTube) Формат: 140+ видео (30–60 мин каждое); глубокий разбор «почему» за алгоритмом, а не только «как». Упор на интуицию и рассуждение. Плюсы:

- Лучший канал для понимания логики алгоритмов на продвинутом уровне.
- Разбирает edge cases и вопросы типа «почему не X?» — именно то, что спрашивают на Senior-интервью.
- Регулярно цитируется в дискуссиях про подготовку к FAANG.

Минусы:

- Контент обновляется редко.
- Нет структурированного курса / roadmap.

Вердикт: ★★★★☆ — дополняет NeetCode: там где NeetCode даёт чистый ответ, BackToBackSWE объясняет «грязный путь» к нему — именно это ценят интервьюеры.

---

## ТОП-рекомендации для алгоритмов

+ С чего начать бесплатно

1. Тренировки по алгоритмам от Яндекса — записи предыдущих сезонов (1.0–6.0) на YouTube + текущий сезон. Дают понимание точного уровня и формата задач Яндекса.
2. Бесплатный курс Яндекс Практикум — 10 часов для оценки уровня и структуры интервью.
3. NeetCode 150 — бесплатный roadmap из 150 задач с видео. Закрывает 95% паттернов.
4. AlgoMaster Newsletter + BackToBackSWE — бесплатные справочники по паттернам.
5. Stepik АиСД (если нужно академически закрыть пробелы в ДП / жадных алгоритмах) — бесплатно.

Что докупить

1. balun.courses — лучший RU-ресурс под РФ-бигтех: покупать интенсивы по проблемным темам (бесплатный курс по СД — первый шаг). Практика на реальных задачах с российских интервью.
2. algocode.io (~2 650 ₽/мес) — если нужны мок-интервью с партнёром и теги компаний. Также даёт доступ к System Design курсу.
3. Grokking the Coding Interview (~$14.9/мес) — если хочется паттерн-ориентированного EN-курса с интерактивом.
4. LeetCode Premium (~$35/мес) — перед конкретными собеседованиями для company-specific задач.
5. Влад Тен (11 000 ₽, бессрочно) — если предпочитаете обучение через видео, хотите большой объём разборов и доступ к DP/Graphs в RU-формате.

---

# B. Mobile / iOS System Design

## Сводная таблица

| # | Название | Провайдер | Ссылка | Цена | Язык | Формат | Уровень |
|---|---|---|---|---|---|---|---|
| 1 | GitHub: mobile-system-design (weeeBox) | Alex Lementuev (ex-Booking/Meta) | github.com/weeeBox | Бесплатно | EN | Текст + схемы + упражнения | Senior |
| 2 | Cracking the Mobile System Design Interview | themobileinterview.com | themobileinterview.com | Бесплатно (статья) | EN | Текст / статья | Senior |
| 3 | Grokking the Mobile System Design Interview | Educative.io | educative.io | Подписка ~$14.9/мес | EN | Текст + интерактив | Senior |

| # | Название | Провайдер | Ссылка | Цена | Язык | Формат | Уровень |
|---|---|---|---|---|---|---|---|
| 4 | Grokking the Modern System Design Interview | Educative.io | educative.io | Подписка ~$14.9/мес | EN | Текст + интерактив | Senior |
| 5 | Mobile System Design Book (Tjeerd in 't Veen) | mobilesystemdesign.com | mobilesystemdesign.com | Платно (книга / серия томов; без $300+ курсов) | EN | Книга (PDF/EPUB) | Senior–Staff |
| 6 | Mobile System Design Book (Manuel Vivo) | manuelvivo.dev | manuelvivo.dev/mobilesystemdesign | Платно (книга) | EN | Книга | Senior–Staff |
| 7 | System Design Interview Course | Exponent (tryexponent.com) | tryexponent.com | Подписка ~$12/мес (годовой план) | EN | Видео (53 урока, 35 видео) + мок-интервью | Senior |
| 8 | ByteByteGo (книги + рассылка) | Alex Xu / ByteByteGo | bytebytego.com | Частично бесплатно (рассылка); книги System Design Interview Vol.1/2 — платно (~$40 каждая) | EN | Текст + диаграммы | Senior–Staff |
| 9 | System Design Handbook (бесплатный гайд) | systemdesignhandbook.com | systemdesignhandbook.com | Бесплатно (гайд) | EN | Текст + схемы | Senior |
| 10 | Building Mobile Apps at Scale | Gergely Orosz (The Pragmatic Engineer) | mobileatscale.com | Платно (~$30 книга) | EN | Книга | Senior–Staff |
| 11 | iOS System Design (Ievabond/ios-mobile-system-design) | Ievabond (GitHub, RU) | github.com/levabond | Бесплатно | RU/EN | Текст + схемы | Senior |

| # | Название | Провайдер | Ссылка | Цена | Язык | Формат | Уровень |
|---|---|---|---|---|---|---|---|
| 12 | Avito System Design (статья на Habr) | Avito Tech | habr.com/avito | Бесплатно | RU | Статья | Senior |
| 13 | FaangTalk iOS System Design (YouTube) | FaangTalk | youtube.com FaangTalk | Бесплатно | RU | Видео | Senior |

## Краткие разборы

### 1. GitHub: weeeBox/mobile-system-design — обязательный ресурс

Провайдер: Alex Lementuev (ex-Booking, ex-Meta; @weeeBox) (github.com/weeeBox/mobile-system-design) Стоимость: Бесплатно (открытый репозиторий) Формат: Текстовый фреймворк с разборами задач (Chat App, Image Library, File Downloader, Analytics Service и ~20 других). Список инженерных блогов Uber, Airbnb, Instagram, LinkedIn. Ключевые темы: фреймворк интервью (требования → HLD → компоненты → deep dive), API-дизайн, pagination, real-time updates, offline support, DI, Image Loading, Caching. Плюсы:

- Золотой стандарт для Mobile System Design интервью — на него ссылаются практически во всех обсуждениях (Blind, Reddit, инженерные блоги).
- Охватывает и iOS, и Android; примеры с iOS-специфичными библиотеками (SDWebImage, Kingfisher).
- Список задач (exercises) — отличный список вопросов для мок-интервью.
- Бесплатно.

Минусы:

- Обновляется нечасто (основная часть создана в 2021–2022).
- Нет структурированного «курса» — нужна самодисциплина.
- Нет RU-версии.

Вердикт: ★★★★★ — первый ресурс, с которого нужно начинать Mobile System Design.

### 2. Cracking the Mobile System Design Interview — themobileinterview.com

Провайдер: themobileinterview.com (themobileinterview.com) Стоимость: Базовая статья — бесплатно; сайт предлагает дополнительные платные материалы. Формат: Подробная статья-фреймворк из 6 шагов: (1) Understand, (2) Scope, (3) Technical Requirements, (4) High-Level Design, (5) Deep Dive, (6) Wrap Up. Плюсы:

- Самый структурированный публичный фреймворк для mobile system design.
- Детально описывает каждый шаг с примерами (архитектуры MVVM/VIPER, wireframing, data entities, API design).
- Бесплатно.
- Упоминается в большинстве гайдов по iOS-собеседованиям (Blind, StackOverflow).

Минусы:

- Одна статья, не курс.
- Материал 2021 года — некоторые примеры немного устарели (RxSwift focus).

Вердикт: ★★★★★ — обязательное чтение (30–40 минут). Даёт чёткий скелет для любого mobile SD-вопроса.

---

### 3. Grokking the Mobile System Design Interview — Educative.io

Провайдер: Educative.io (educative.io/courses/mobile-system-design) Стоимость: Подписка ~$14.9/мес Формат: 81 урок, 3 мок-интервью, ~14 часов; обновлён 2026. Курс от industry engineers. Темы: мобильная архитектура, масштабируемость, resilience, user-centric дизайн; AI-powered practice. Плюсы:

- Единственный специализированный mobile system design курс на крупной обучающей платформе.
- AI-powered практика и интерактивные элементы.
- Охватывает специфику мобильной архитектуры, а не просто backend SD.

Минусы:

- Нет имени конкретного автора (только «industry engineers»).
- Отзывы смешанные: хорошо для старта, недостаточно глубоко для Staff-уровня.
- Нет видео.

Вердикт: ★★★☆☆ — полезен как структурированный курс; не заменяет weeeBox + реальные кейсы. Имеет смысл в рамках общей Educative-подписки.

---

### 4. Grokking the Modern System Design Interview — Educative.io

Провайдер: Educative.io (educative.io/courses/grokking-modern-system-design-interview) Стоимость: Подписка ~$14.9/мес Формат: Текстовый интерактивный курс; кейсы на Slack, Uber, YouTube; архитектурные паттерны (кэширование, шардинг, pub/sub, rate limiting). Плюсы:

- Глубокое погружение в distributed systems паттерны.
- Полезен для понимания backend-части, когда интервьюер просит «end-to-end» дизайн.
- Современные кейсы (Slack, Uber).

Минусы:

- Backend-ориентирован, мало iOS-специфики.
- Для mobile SD важен как дополнение, а не основа.

Вердикт: ★★★★☆ — если нужно уверенно говорить о backend-стороне системы (API, БД, CDN, push notifications) — обязателен. Для tech lead это особенно важно.

---

### 5. Mobile System Design Book — Tjeerd in 't Veen

Провайдер: mobilesystemdesign.com (mobilesystemdesign.com) Автор: Tjeerd in 't Veen, ex-Staff Engineer at X (Twitter), Mobile Tech Lead at ING Стоимость: Серия книг (PDF/EPUB); позиционируется как дешевле курсов за $300+ Формат: 3 тома: Fundamentals, UI, Apps at Scale; Quick Reference Cheat Sheet; ~175 тем; 7 полностью решённых вопросов; 5-шаговый playbook для интервью. Плюсы:

- Написан iOS tech lead'ом — полностью мобильная перспектива.
- Реальный опыт, а не академические примеры.

- Quick Reference — удобный cheat sheet накануне интервью.
- Вышел финальный вариант в январе 2026.

Минусы:

- Некоторые рецензенты отмечают поверхностность примеров (особенно для сложных тем).
- Англоязычный.
- Не заменяет практику мок-интервью.

Вердикт: ★★★★☆ — хорошее дополнение к weeeBox и themobileinterview.com для системного покрытия темы.

---

### 6. Mobile System Design Book — Manuel Vivo (Google)

Провайдер: manuelvivo.dev (manuelvivo.dev/mobilesystemdesign) Автор: Manuel Vivo, инженер Google (Android) Стоимость: Платно (книга) Формат: 10 глав (fundamentals → advanced); 7 полностью решённых вопросов; dual-platform (iOS + Android); 5-шаговый playbook. Плюсы:

- Один из немногих ресурсов, охватывающих обе платформы с явным iOS-контентом.
- Рецензии от инженеров Facebook, NYT.
- Чёткая структура «от требований к архитектуре».

Минусы:

- Упор на Android-примеры из-за авторского фона.
- Платно.

Вердикт: ★★★★☆ — ценен для iOS-инженера, который хочет понять cross-platform принципы.

---

### 7. System Design Interview Course — Exponent

Провайдер: Exponent (tryexponent.com/courses/system-design-interviews) Стоимость: ~$12/мес (годовой план); free preview доступен; 5 peer-мок интервью/мес бесплатно. Формат: 10 модулей, 53 урока, 35 видео; интерактивные упражнения с AI-фидбеком; 3000+ вопросов в базе данных по компаниям. Темы: fundamentals of system design + walkthrough реальных вопросов. Нет специализированного mobile-трека, но покрывает backend-паттерны SD. Плюсы:

- Видеоформат с реальными интервью (engineers + managers из Meta, eBay, Amazon).
- Community с peer-мок интервью — уникальная функция.
- Обновлён в июне 2026.
- Доступны 5 бесплатных мок-интервью с пирами в месяц.

Минусы:

- Нет mobile-специфического трека.
- Ориентирован на backend/distributed SD больше, чем на mobile.

Вердикт: ★★★☆☆ — полезен для backend-части SD и мок-интервью; мобильная составляющая слабее weeeBox.

---

### 8. ByteByteGo — Alex Xu

Провайдер: ByteByteGo (bytebytego.com) Стоимость: Еженедельная рассылка — бесплатно; книги «System Design Interview Vol.1/2» — ~$40 каждая; платная подписка ($150/год). Формат: Diagram-first подход; книги описывают 15–16 кейсов (URL shortener, YouTube, Instagram);

рассылка — архитектурные insights. Плюсы:

- Лучшие в индустрии диаграммы — помогают визуализировать архитектуру.
- Книги стали стандартом для backend SD — понимание фундамента необходимо.
- Рассылка бесплатна и очень качественна.

Минусы:

- Нет практических задач / тренажёра.
- Backend-ориентирован, мало мобильной специфики.
- Диаграммы иногда упрощают сложные distributed системы.

Вердикт: ★★★★☆ — обязательная часть backend-знаний для tech lead. Книга Vol.1 — базовое чтение. Не заменяет mobile-специфику.

---

### 9. System Design Handbook (бесплатный гайд)

Провайдер: systemdesignhandbook.com (systemdesignhandbook.com) Стоимость: Бесплатно (гайд) Формат: Статьи по ключевым концепциям SD: scalability, reliability, distributed systems, caching, load balancing; аннотированные примеры. Плюсы:

- Бесплатный, хорошо структурированный справочник.
- Удобен для быстрого освежения концепций.
- Публикует независимые обзоры других курсов SD.

Минусы:

- Не курс, а справочник.
- Нет mobile-специфики.

Вердикт: ★★★☆☆ — хороший бесплатный справочник по backend-концепциям SD.

---

### 10. Building Mobile Apps at Scale — Gergely Orosz

Провайдер: mobileatscale.com (mobileatscale.com) Автор: Gergely Orosz, ex-Uber, автор The Pragmatic Engineer Стоимость: ~$30 (книга) Формат: Книга; 39 инженерных вызовов масштабирования мобильных приложений. Темы: модульная архитектура, зависимости, навигация в больших приложениях, CI/CD, feature flags, offline режим, аналитика, производительность, app size. Плюсы:

- Уникальный контент: реальные проблемы больших мобильных команд (Uber, Instagram, Airbnb).
- Обязательное чтение для tech lead уровня.
- Напрямую применимо к System Design вопросам о масштабировании.

Минусы:

- Книга, а не интерактивный курс.
- Нет готовых «ответов на интервью» — нужна экстраполяция.

Вердикт: ★★★★★ — must-read для tech lead; даёт реальные примеры масштабирования, которые делают SD-ответы убедительными.

---

### 11. GitHub: levabond/ios-mobile-system-design (RU)

Провайдер: levabond (github.com/levabond/ios-mobile-system-design) Стоимость: Бесплатно Формат: Фреймворк на русском языке для iOS System Design интервью; структура секции, как

проводить интервью и как готовиться; пример с Twitter feed. Плюсы:

- На русском языке — единственный подробный RU-фреймворк для iOS SD.
- Описывает структуру интервью с точки зрения интервьюера (как в РФ-бигтех).
- Практические советы по инструментам (Miro, Whimsical, Excalidraw).

Минусы:

- Небольшой объём (один пример задачи).
- Не обновляется активно.

Вердикт: ★★★★☆ — ценен именно для подготовки к российскому формату SD-интервью. Используйте параллельно с weeeBox.

---

### 12. Avito: «Готовимся к System Design Interview»

Провайдер: Avito Tech (habr.com/avito) Стоимость: Бесплатно Формат: Статья на Habr от команды Avito о том, как проходит Frontend/Mobile System Design секция в Avito, что оценивается. Плюсы:

- Взгляд изнутри на то, что именно оценивают в Avito на SD-секции.
- Конкретные примеры компонентов и схем.
- Бесплатно, от первоисточника.

Минусы:

- Одна статья, не курс.
- Акцент на Frontend (но Mobile-часть применима).

Вердикт: ★★★★☆ — обязательное чтение перед интервью в Avito.

---

### 13. FaangTalk: iOS System Design (YouTube)

Провайдер: FaangTalk (youtube.com FaangTalk), канал @faangtalk Стоимость: Бесплатно Формат: Видео-разборы iOS System Design секций на русском языке; разборы «Чем уникален мобильный System Design», мок-интервью. Плюсы:

- На русском, ориентирован на FAANG и РФ-бигтех.
- Практические мок-интервью с разборами.

Минусы:

- Нерегулярный выпуск контента.

Вердикт: ★★★★☆ — полезный RU-ресурс для понимания формата и типичных вопросов.

---

## ТОП-рекомендации для Mobile/iOS System Design

### + С чего начать бесплатно

1. weeeBox/mobile-system-design — прочитать весь README + разобрать 3–5 упражнений (Chat App, Image Library, Analytics).
2. themobileinterview.com — фреймворк — изучить 6-шаговый процесс и применить к каждому упражнению weeeBox.
3. levabond/ios-mobile-system-design — русский фреймворк для адаптации к российскому формату.
4. Avito SD статья — понять, что оценивают в Avito.

5. FaangTalk YouTube — бесплатные RU-разборы iOS SD.
6. ByteByteGo рассылка — бесплатная архитектурная рассылка (подписка на newsletter).
7. System Design Handbook — бесплатный справочник по backend-концепциям.

Что докупить

1. Building Mobile Apps at Scale (~$30) — обязательная книга для tech lead. Реальные кейсы масштабирования.
2. Mobile System Design Book — Tjeerd in 't Veen — playbook с cheat sheet; удобен накануне интервью.
3. Grokking Mobile System Design (Educative ~$14.9/мес) — структурированный курс; особенно полезен в рамках общей Educative-подписки вместе с Grokking Coding Interview и Modern System Design.
4. ByteByteGo книги (Vol.1 ~$40) — для понимания backend-архитектуры, которую нужно уметь объяснять в end-to-end SD-вопросах.
5. Exponent (~$12/мес) — если нужны peer-мок интервью по SD (5 бесплатных / мес).

---

# C. Дополнительные источники для поиска

## Telegram-каналы

| Канал | Описание | Ссылка |
|---|---|---|
| iOS Dev IQ | Подборка материалов для iOS System Design, публичные разборы, Boosty с контентом | t.me/ios_prog |
| FaangTalk | System Design и алгоритмы на русском; анонсы в @faangtalk_news, чат @faangtalk | t.me/faangtalk |
| Алгоритмы — Собеседования, Олимпиады, ШАД | Задачи и разборы по алгоритмам с решениями; подготовка к собеседованиям и ШАД | t.me/algorithms_ru |
| SwiftBook (новости и чат) | Крупнейшее RU-iOS сообщество; чат + новости | t.me/swiftbook_news / t.me/swiftbook_chat |
| iOS Good Reads | Интересные статьи, видео и новости по iOS; куратор — руководитель мобильной разработки Avito | t.me/iosgr |
| iOS Heads | Интересные кейсы и новости iOS-разработки | t.me/ios_heads |
| Я.Субботник мобильная разработка | Чат митапа Яндекса по мобильной разработке | t.me/mobilesubbotnik |
| Silicon Channel | Подготовка к собеседованиям в РФ-бигтех; roadmap алгоритмов, пул вопросов | t.me/siliconchannel |
| coding_interviews | Советы по прохождению coding interviews; паттерны, ссылки | t.me/coding_interviews |

## GitHub-репозитории

| Репозиторий | Описание | Ссылка |
|---|---|---|
| weeeBox/mobile-system-design | Фреймворк и упражнения Mobile System Design; список инженерных блогов | github.com/weeeBox/mobile-system-design |
| levabond/ios-mobile-system-design | RU-версия фреймворка iOS System Design интервью | github.com/levabond/ios-mobile-system-design |
| beagreatengineer/algo-interview | Бесплатный план подготовки к алгоритмическому интервью с ресурсами | github.com/beagreatengineer/algo-interview |
| vyacheslavskiy/iOS-Interview-Questions | Вопросы на собеседованиях iOS-разработчиков с ответами и примерами кода (RU) | github.com/vyacheslavskiy/iOS-Interview-Questions |
| NeetCode Roadmap | Официальный roadmap 150 задач с видео | github.com/neetcode-gh/leetcode |

## Инженерные блоги для Mobile System Design

| Блог / Ресурс | Платформа | Ссылка |
|---|---|---|
| Яндекс Tech (мобильная разработка) | Habr | habr.com/yandex |
| Яндекс.Карты Mobile | Medium | medium.com/yandex-maps-mobile |
| Avito Tech (iOS) | Habr | habr.com/avito |
| Uber Engineering Blog | uber.com | eng.uber.com |
| Airbnb Engineering | Medium | medium.com/airbnb-engineering |
| Instagram Engineering | Medium | instagram-engineering.com |
| The Pragmatic Engineer (Gergely Orosz) | Substack | newsletter.pragmaticengineer.com |
| Jacob's Tech Tavern (iOS SD статьи) | Substack | blog.jacobstechtavern.com |
| weeeBox BLOGPOSTS.MD (подборка статей из блогов) | GitHub | github.com/weeeBox/.../BLOGPOSTS.MD |
| iOS Interview (ios-interview.ru) | Сайт | ios-interview.ru |

## Платформы для мок-интервью

| Платформа | Описание | Ссылка |
|---|---|---|
| it-interview.io | Бесплатный Telegram-бот для поиска напарника на мок-интервью (алгоритмы, SD, Go, Java и др.) | it-interview.io |
| Exponent | Peer-мок интервью (5 бесплатных/мес); platform по SD и алгоритмам | tryexponent.com |
| interviewing.io | Анонимные мок-интервью с инженерами из FAANG | interviewing.io |
| Pramp | Бесплатные peer-to-peer мок-интервью | pramp.com |

## Публичные мок-собеседования iOS (YouTube RU)

| Канал | Описание | Ссылка |
|---|---|---|
| Podlodka iOS Crew | Публичные собеседования iOS-разработчиков; обзор после каждой секции | podlodka.io/ioscrew |
| Devrush | Мок-собеседования Middle/Senior iOS с разборами | t.me/dev_rush |
| iOS Interview (ios-interview.ru) | Сборник публичных собеседований iOS Junior–Senior | ios-interview.ru/video-ios-interviews |

## Итоговая «дорожная карта» подготовки

```
АЛГОРИТМЫ (8–12 недель)
│
├── СТАРТ (бесплатно, 1–2 нед)
│   ├── Яндекс Практикум бесплатный курс (оценка уровня)
│   ├── NeetCode 150 roadmap (структура паттернов)
│   └── Записи Тренировок Яндекса 1.0–6.0 (YouTube)
│
├── ОСНОВНАЯ ПОДГОТОВКА (платно, 4–8 нед)
│   ├── balun.courses (интенсивы по проблемным темам)
│   ├── LeetCode (практика по паттернам, 3–5 задач/день)
│   └── algocode.io (мок-интервью с партнёром еженедельно)
│
└── ФИНАЛ (1–2 нед до интервью)
    ├── Тренировки Яндекса текущего сезона
    └── AlgoMaster/BackToBackSWE (повторение паттернов)

MOBILE SYSTEM DESIGN (4–6 недель параллельно)
│
├── ФРЕЙМВОРК (бесплатно, 1 нед)
│   ├── weeeBox/mobile-system-design (README + 3 упражнения)
│   └── themobileinterview.com (6-шаговый процесс)
│
├── КОНТЕКСТ РФ (бесплатно, параллельно)
│   ├── levabond/ios-mobile-system-design (RU-адаптация)
│   └── Avito Habr статья + FaangTalk YouTube
│
├── BACKEND-ЗНАНИЯ (платно, 2–3 нед)
│   ├── ByteByteGo Vol.1 (диаграммы + кейсы)
│   └── Grokking Modern System Design (Educative, подписка)
│
└── ПРАКТИКА (платно/бесплатно, ongoing)
    ├── Building Mobile Apps at Scale (книга Orosz)
    ├── Мок-интервью: it-interview.io / Exponent / peer
    └── Инженерные блоги (Avito, Яндекс, Uber, Airbnb)
```

*Ресурсы актуальны на июнь 2026. Цены и форматы могут меняться; проверяйте актуальную информацию на сайтах провайдеров.*