1. Структура курса: Архитектура информационных систем (17 лекций)

1.1. Модуль 1: Архитектурные основы (Лекции 1-3)

1.1.1. Лекция 1: Введение. Принципы построения приложения

- Оптимизация системы требует выбора критерия оптимизации. (человеческие ресурсы, масштабируемость, надежность, стоимость)
- Каждая проблема в отдельности решаема, но комбинация простых решений приводит к экспоненциальному росту сложности системы.
- SOLID принципы и Dependency Injection на практике
- Clean Architecture

1.1.2. Лекция 2: Альтернативные архитектурные паттерны

Не всегда SOLID и Clean Architecture — оптимальный выбор. Разные подходы лучше работают в разных контекстах.

- Framework-driven (Rails, Django): быстрый старт за счёт конвенций, но vendor lock-in
- Actor Model (Erlang/Elixir/Akka): изоляция состояния, встроенная отказоустойчивость и параллелизм
- Data-oriented design: максимум производительности, ориентирован на конкретное железо
- Functional Core, Imperative Shell: чистые функции в логике, побочные эффекты на границах

1.1.3. Лекция 3: API дизайн для распределённых систем

Выбор протокола взаимодействия влияет на производительность, масштабируемость и удобство разработки.

- REST: стандартный веб-протокол, простая отладка, но избыточный трафик с комплексными данными
- GraphQL: клиент запрашивает только нужные данные, снижает трафик, сложнее кэшировать
- gRPC: бинарный протокол, высокая производительность, использование в микросервисах
- WebSocket: двусторонний канал, необходим для реал-тайм, требует управления состоянием

1.2. Модуль 2: Фундаментальные структуры современных хранилищ данных (Лекции 4-6)

Связь с Модулем 1: Архитектурные решения требуют компромиссов между критериями оптимизации. Модуль 2 демонстрирует это на примере хранилищ данных, показывая, как разные структуры данных решают одну задачу под разные задачи.

1.2.1. Лекция 4: Базовые механизмы хранения данных

Рассматривается путь от простых файлов к системам с устойчивостью к сбоям.

- CSV-файлы как минимальная форма персистентности
- Append-only CSV: идея Write-Ahead Log и журналирования изменений
- Хеш-индексы как первый уровень ускорения выборки
- Ограничения и проблемы таких подходов (фрагментация, поиск, консистентность)

1.2.2. Лекция 5: Структурированные индексы

Продолжение темы индексации и оптимизации доступа. Фокус на идее упорядоченности и организации данных для масштабных систем.

- Sorted String Table (SSTable): концепция сортированных сегментов
- Log-Structured Merge-Tree (LSM-Tree): лог-ориентированная запись и фоновая компакция
- B-деревья и их модификации
- Сравнение LSM- и B-деревьев: производительность, стоимость обновлений, области применения
- Вывод: компромиссы между скоростью записи и чтения

1.2.3. Лекция 6: ACID, уровни изоляции, отличие аналитических и транзакционных хранилищ

Завершающая лекция объединяет принципы консистентности, параллелизма и аналитических нагрузок.

- ACID и уровни изоляции: практическое влияние на производительность
- OLAP и колоночные базы данных (пример — ClickHouse): принципы сжатия и эффективных агрегаций
- Сравнение подходов: CSV / Append-only / Hash Index / SSTable / LSM / B-Tree
- Итоговая таблица CRUD-сложности и применимости

1.3. Модуль 3: Нереляционные и специализированные БД (Лекции 7-10)

Модуль 3 показывает, как разные типы нереляционных и специализированных БД решают задачи, где традиционные структуры становятся ограничением — по масштабируемости, латентности или типу данных.

1.3.1. Лекция 7: Документные хранилища

- MongoDB: sharding strategies, write concerns

1.3.2. Лекция 8:

- Inverted indexes: Lucene (Elasticsearch, Solr)
- Time-series: Prometheus architecture, downsampling
- Object storage: MinIO

1.3.3. Лекция 9:

- Geospatial: Minimum Bounding Rectangles, R-Tree, QuadTree
- Графовые БД: Список смежности, Матрица смежности, B-Tree для индексации

1.3.4. Лекция 10: Векторные БД

- Вектор как единица данных: эмбеддинги текста, изображений, аудио
- Семантический поиск документов / Поиск похожих изображений
- Inverted File Indexing
- Hierarchical navigable small world

1.3.5. Лекция 11: In-memory БД

- DuckDB: встроенная аналитика и векторизованное выполнение запросов
- Redis: структуры данных и модели персистентности

1.4. Модуль 4: Компоненты распределённых систем (Лекции 12-15)

Модуль 4 показывает как компоненты распределённых систем решают фундаментальные проблемы консистентности, надёжности и производительности в масштабе.

1.4.1. Лекция 12: Основы распределенных систем

Ключевой вызов распределённых систем заключается в необходимости согласования состояния при условиях неопределённости сетевых задержек, отказов узлов и асинхронности коммуникации.

- Теорема CAP и практические примеры компромиссов
- Репликация
  - Лидер-последователь (Leader-Follower)
  - На основе кворума (quorum-based)
- Консенсусные алгоритмы
  - Raft
  - Paxos
- Окончательная консистентность (Eventual consistency) и идемпотентность операций
- Двухфазный коммит (2PC, Two-Phase Commit)

1.4.2. Лекция 13:

- Forward Proxy
- Reverse Proxy
  - Load Balancing
    - Round-Robin
    - Least Connections

    - IP Hash
- Rate Limiting
  - Fixed Window
  - Token Bucket
  - Leaky Bucket
  - Sliding Window Log / Counter

1.4.3. Лекция 14:

- Authentication / Authorization
- Управление доступом
  - На основе ролей (Role-Based Access Control, RBAC)
  - На основе аттрибутов (Attribute-Based Access Control, ABAC)
- Единый вход (Single Sign-On)

1.4.4. Лекция 15: Асинхронная коммуникация

Асинхронная коммуникация через очереди сообщений и события.

- Event Bus и архитектура, ориентированная на события (Event Driven Architecture)
- Publish-Subscribe vs Message Queue
- Saga Pattern: распределённые транзакции без двухфазного коммита, компенсирующие действия при ошибках
- Гарантии доставки сообщений: at-most-once (может потеряться), at-least-once (могут быть дубли), exactly-once (идеал, но дорого)

1.4.5. Лекция 16:

- Логгирование
  - structured logging
  - log aggregation
  - ELK stack
- Observatility & Monitoring
  - Prometheus
  - Grafana
  - alerting
- Оркестрация: Kubernetes primitives (Pods, Services, Ingress)
- Планировщики задач
  - Примитивы
    - Direct Acyclic Graphs
    - Queue / Worker Pool
  - Конкретные примеры
    - Celery
    - Airflow
    - Dagster

(остановился тут, дальше сгенерено)

1.5. Модуль 5: System Design (Лекция 17)

1.5.1. Лекции 17: разбор дизайна конкретной системы

—

2. 10 System Design задач с векторными БД (масштаб миллионов пользователей)

2.1. 1. Instagram-подобная социальная сеть с AI-powered content discovery

Масштаб: 10M пользователей, 500K DAU

Векторный компонент: Semantic feed ranking через post embeddings
- Posts → CLIP embeddings → vector search для “similar content”
- User interest profiles как векторные представления
- Hybrid retrieval: collaboration + vector similarity для Explore page
- **Расчёт**: 500K DAU × 50 posts/day = 25M posts/day
- Embedding dimension 768 × 4 bytes = 3KB per post
- Vector DB: 25M × 3KB = 75GB/day только embeddings

2.2. 2. E-commerce платформа с visual и semantic search

Масштаб: 5M products, 2M DAU

Векторный компонент: Multi-modal product search (текст + изображения)
- Product images → ResNet embeddings + text descriptions → BERT embeddings
- Unified embedding space для cross-modal search
- “Find similar items” через vector similarity
- **Расчёт**: 5M products × (512 dim image + 384 dim text) × 4 bytes = 17GB
- Queries: 2M DAU × 10 searches/day = 20M queries/day
- Latency requirement: p99 < 100ms

2.3. 3. Music streaming с AI recommendations

Масштаб: 8M users, 1M DAU, 50M tracks

Векторный компонент: Audio fingerprinting + user taste embeddings
- Spotify-style: track embeddings (audio features + collaborative signals)
- Daily Mix генерация: clustering в vector space
- “Radio” feature: nearest neighbors в embedding space
- **Расчёт**: 50M tracks × 256 dimensions × 4 bytes = 50GB track embeddings
- 1M DAU × 50 tracks/day = 50M retrievals/day
- Cold start: mean of K-nearest для новых треков

2.4. 4. Fraud detection система для финтех

Масштаб: 3M users, 10M transactions/day

Векторный компонент: Real-time anomaly detection через transaction embeddings
- Transaction features → embeddings → vector similarity поиск
- Normal behavior baseline: clustering legitimate transactions
- Fraud detection: distance from cluster centroids
- **Расчёт**: 10M transactions/day × 128 dim × 4 bytes = 5GB/day
- Real-time inference: sub-10ms latency requirement
- False positive rate < 1%, false negative < 0.1%

2.5. 5. Content moderation система для UGC платформы

Масштаб: 20M users, 5M posts/day (text + images)

Векторный компонент: Similarity search для duplicate/violating content
- Known violations → embeddings → vector index
- New content → embedding → nearest neighbor search
- Cascade: hash-based dedup → vector similarity → ML classifier
- **Расчёт**: 5M posts/day × (768 text + 512 image) × 4 bytes = 25GB/day
- Violation database: 10M known violations × 1.3KB = 13GB
- Latency: must process within 500ms для real-time moderation

2.6. 6. Enterprise knowledge management с semantic search

Масштаб: 50K employees, 10M documents

Векторный компонент: RAG для document QA + semantic search
- Document chunking: 10M docs → 100M chunks

- Embeddings + metadata filtering (department, date, author)
- Hybrid search: vector semantic + keyword для precision
- **Расчёт**: 100M chunks × 768 dim × 4 bytes = 307GB
- Queries: 50K users × 20 searches/day = 1M queries/day
- Re-ranking: retrieve top-100, rerank to top-10

2.7. 7. Real estate платформа с visual similarity search

Масштаб: 2M listings, 500K DAU

Векторный компонент: “Find similar homes” через image embeddings
- Property images ⟶ CNN embeddings ⟶ vector index
- Filters: price, location (traditional DB) + style similarity (vectors)
- Virtual staging: similar interior styles lookup
- **Расчёт**: 2M listings × 10 images × 512 dim × 4 bytes = 40GB
- Queries: 500K DAU × 15 searches/day = 7.5M queries/day
- IVF sharding: cluster by geographic region + price range

2.8. 8. Dating app с personality matching

Масштаб: 5M users, 800K DAU

Векторный компонент: User compatibility через profile embeddings
- Profile (interests, values, photos) ⟶ unified embedding
- Swipe history ⟶ collaborative filtering embeddings
- Daily matches: K-nearest neighbors в embedding space
- **Расчёт**: 5M users × 256 dim × 4 bytes = 5GB user embeddings
- Matching: 800K DAU × 50 profiles shown = 40M retrievals/day
- Geographic filtering + vector similarity для balanced matches

2.9. 9. Video streaming с content-based recommendations

Масштаб: 15M users, 3M DAU, 100K videos

Векторный компонент: Video understanding через frame embeddings
- Keyframes ⟶ CLIP embeddings ⟶ video-level aggregation
- “More like this”: vector similarity в content space
- Thumbnail optimization: similar successful thumbnails
- **Расчёт**: 100K videos × 10 keyframes × 512 dim × 4 bytes = 2GB
- Watch sessions: 3M DAU × 5 videos × 20 recommendations = 300M retrievals/day
- Two-tower: content embeddings (pre-computed) + user context (real-time)

2.10. 10. Professional networking с skill-based matching

Масштаб: 10M professionals, 1M DAU

Векторный компонент: Job-candidate matching через skill embeddings
- Skills, experience, education ⟶ unified professional embedding
- Job descriptions ⟶ requirement embeddings
- Matching: cosine similarity + traditional filters (location, salary)
- **Расчёт**: 10M profiles × 384 dim × 4 bytes = 15GB
- 500K active jobs × 384 dim × 4 bytes = 768MB
- Matching queries: 1M DAU × 10 jobs viewed = 10M queries/day
- Cold start: new profiles use skill taxonomy nearest neighbors

—

3. Ключевые метрики для векторных систем на масштабе

Общие паттерны для всех 10 задач:

1. **Latency targets**: p50 < 50ms, p99 < 200ms

2. **Recall**: ANN search 90-95% recall k=100
3. **Cost optimization**: quantization (binary/8-bit) для снижения memory footprint
4. **Sharding strategy**: geographic, categorical, или hash-based в зависимости от access patterns
5. **Replication factor**: 3x для высокой availability
6. **Index rebuild**: incremental updates vs full rebuild trade-offs

Курс фокусируется на **engineering decisions** с конкретными production числами, trade-offs и cost models, делая векторные БД равноправным storage layer наравне с SQL/NoSQL.