Основные компоненты архитектуры информационной системы и их взаимодействие

Что такое архитектура информационной системы?

Определение

Архитектура ИС — это фундаментальная организация системы, включая её компоненты, их взаимосвязи, а также принципы проектирования и развития.

Основные цели

Обеспечение надежности, масштабируемости, управляемости и соответствия бизнес-требованиям, закладывая основу для эффективного функционирования и развития системы.

Ключевые компоненты архитектуры ИС

Слой представления **(Frontend)**

Интерфейс, через который пользователи взаимодействуют с системой. Отвечает за отображение данных и пользовательский опыт.

Слой бизнес-логики **(Backend)**

Сердце системы, где происходит обработка данных, реализация бизнес-правил и управление операциями.

База данных

«хранилище» информации, организованное так, чтобы её легко искать, менять и использовать в программах.

Интеграционные компоненты

Механизмы (API, шины данных, брокеры сообщений) для связи с внешними и внутренними системами.

Визуализация архитектуры: диаграмма взаимодействия компонентов

Пользователь

Инициирует запросы через интерфейс.

Фронтенд

Отправляет запросы на сервер приложений.

Бэкэнд

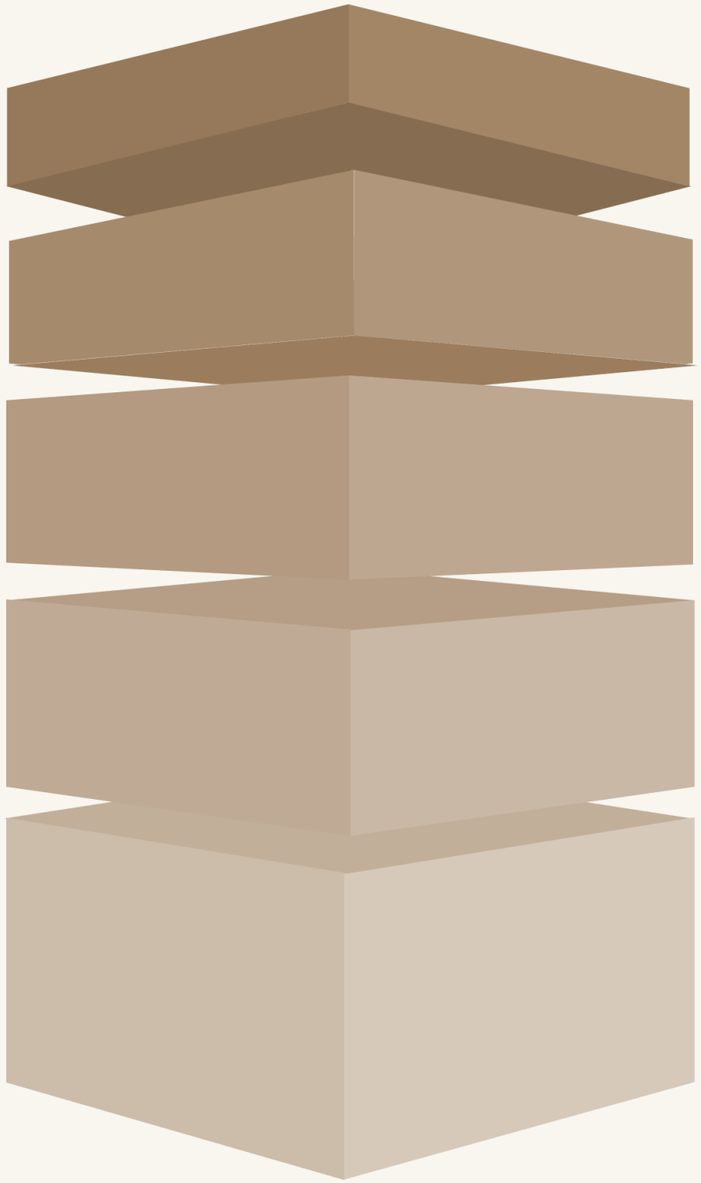
Обрабатывает логику и взаимодействует с сервисами.

База данных

Хранит и возвращает данные по запросу.

Внешние сервисы

Платежи, доставка и уведомления интегрированы.



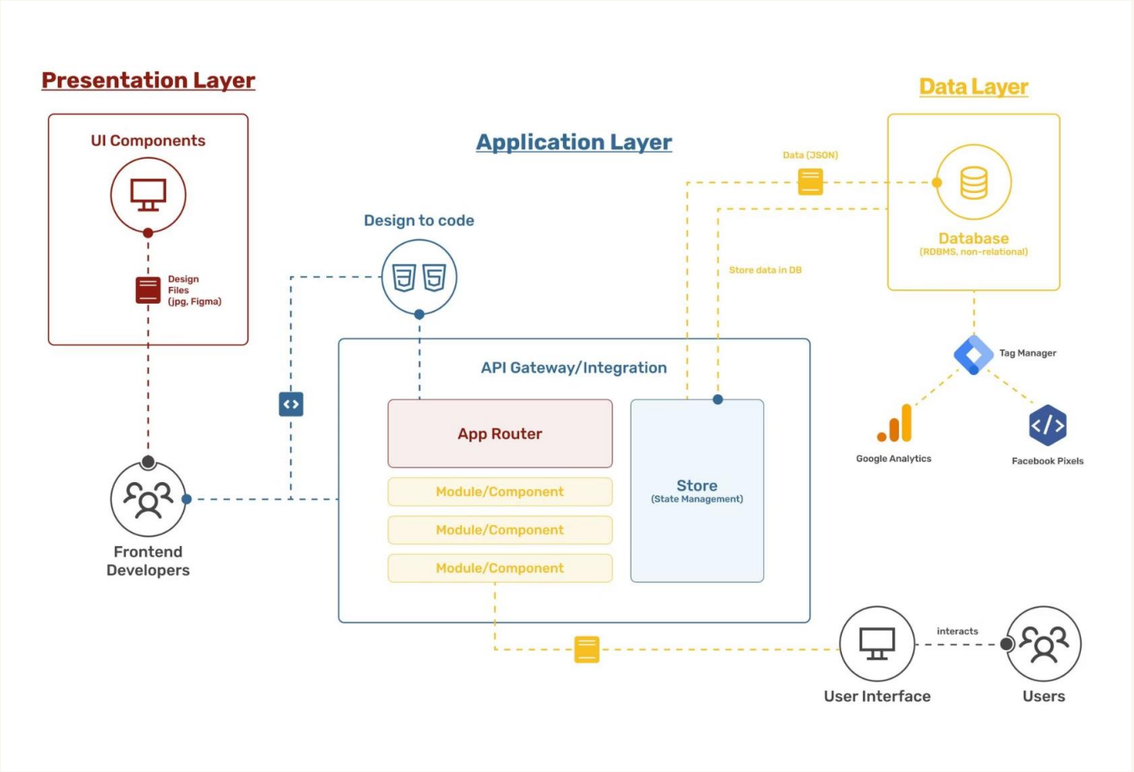
Взаимодействие компонентов: многослойная архитектура

Разделение по функциям

Каждый слой выполняет свою уникальную функцию, взаимодействуя с соседними слоями через строго определённые интерфейсы. Это обеспечивает модульность и чистоту архитектуры.

Пример взаимодействия

Пользователь отправляет запрос через Frontend. Слой бизнес-логики (Backend) обрабатывает его, при необходимости обращаясь к слою хранения данных, и возвращает результат пользователю.

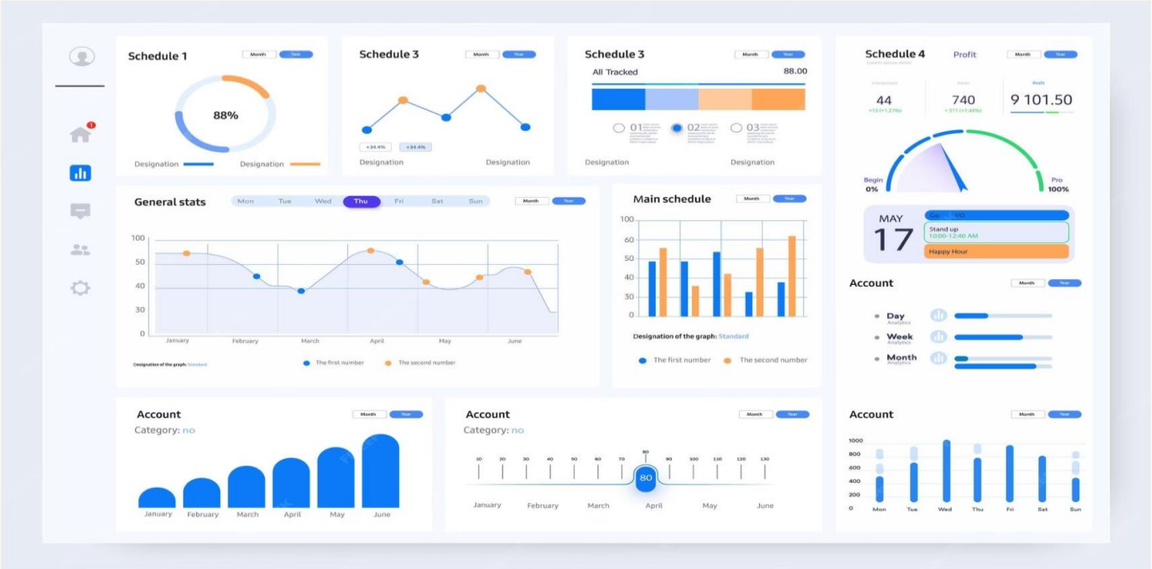


Сквозные компоненты: безопасность и мониторинг

Безопасность

Внедрение механизмов аутентификации, авторизации, шифрования и аудита на всех уровнях архитектуры для защиты данных и предотвращения угроз.

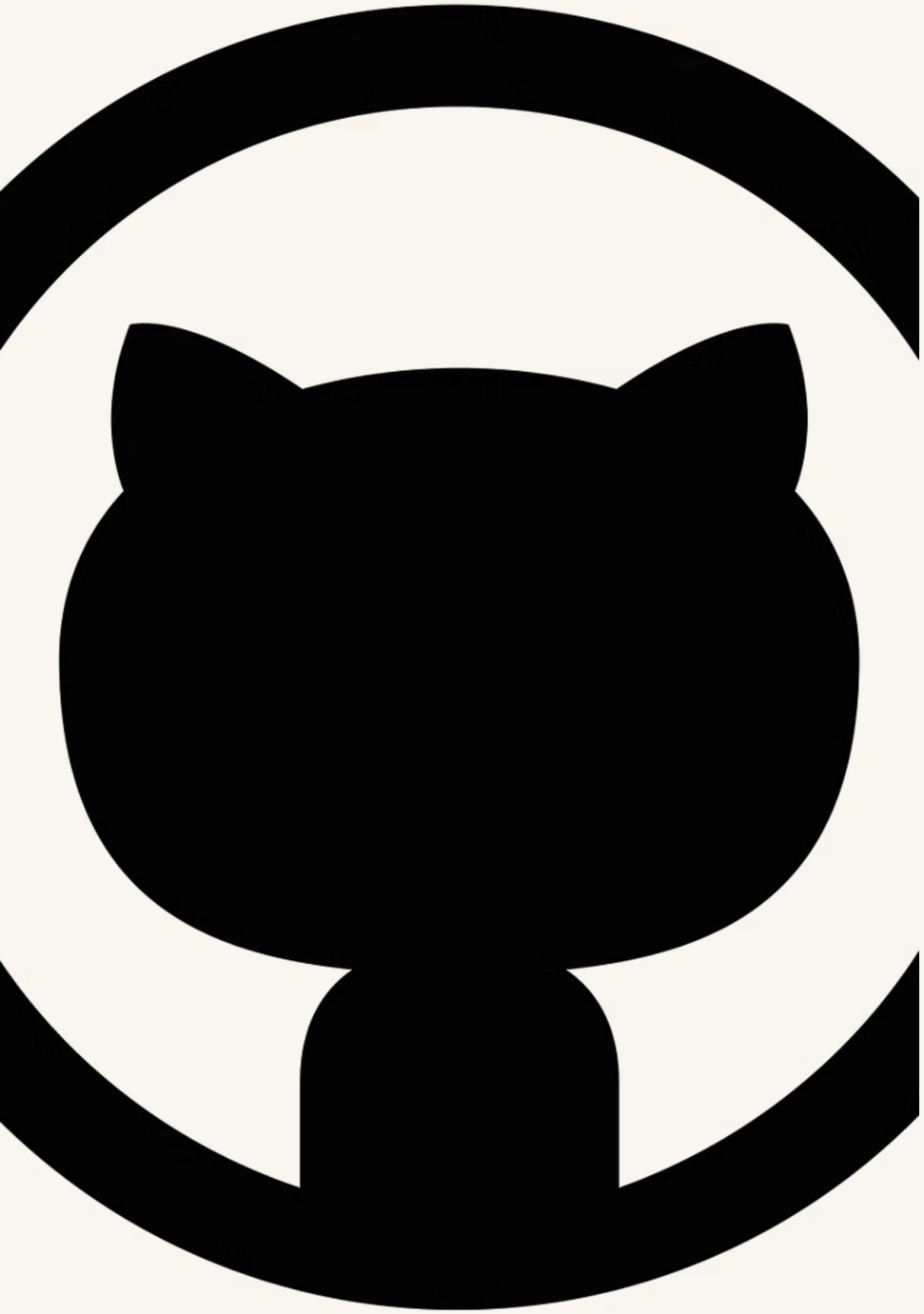
- Управление доступом
- Шифрование данных
- Регулярные аудиты безопасности



Мониторинг и логирование

Постоянный сбор метрик, логов и трассировок для отслеживания состояния системы, выявления проблем и оптимизации производительности.

- Сбор метрик производительности
- Централизованное логирование
- Системы оповещений



Архитектурные стили и их влияние на компоненты

Монолитная архитектура

Все компоненты объединены в одном приложении. Проще на начальных этапах разработки, но сложнее в масштабировании и поддержке больших систем.

Микросервисная архитектура

Система разбита на независимые, слабосвязанные сервисы, каждый из которых выполняет отдельную бизнес-функцию и взаимодействует через API.

Многослойная **(n-tier)**

Разделение системы на логические слои с чёткими границами и ролями, обеспечивающее структурированность и управляемость.

Примеры: мононлитных и микросервисных архитектур

Монолитная архитектура :

Примеры: Shopify, Github, Stack Overflow и т.д

Микросервисная архитектура:

Примеры: Amazon, Wildberries, Авито и много других

Интеграция компонентов и систем: паттерны взаимодействия

Синхронный вызов API (REST, SOAP)

Прямое взаимодействие между сервисами, когда отправитель ждет ответа от получателя. Примеры: вызовы RESTful API для получения данных.

Асинхронный обмен сообщениями

Использование брокеров сообщений (RabbitMQ, Kafka) для обмена данными без немедленного ожидания ответа. Повышает отказоустойчивость и масштабируемость системы.

Общая база данных или файловый обмен

Более простые, но менее гибкие методы интеграции, часто используемые в унаследованных системах. Могут создавать точки отказа.

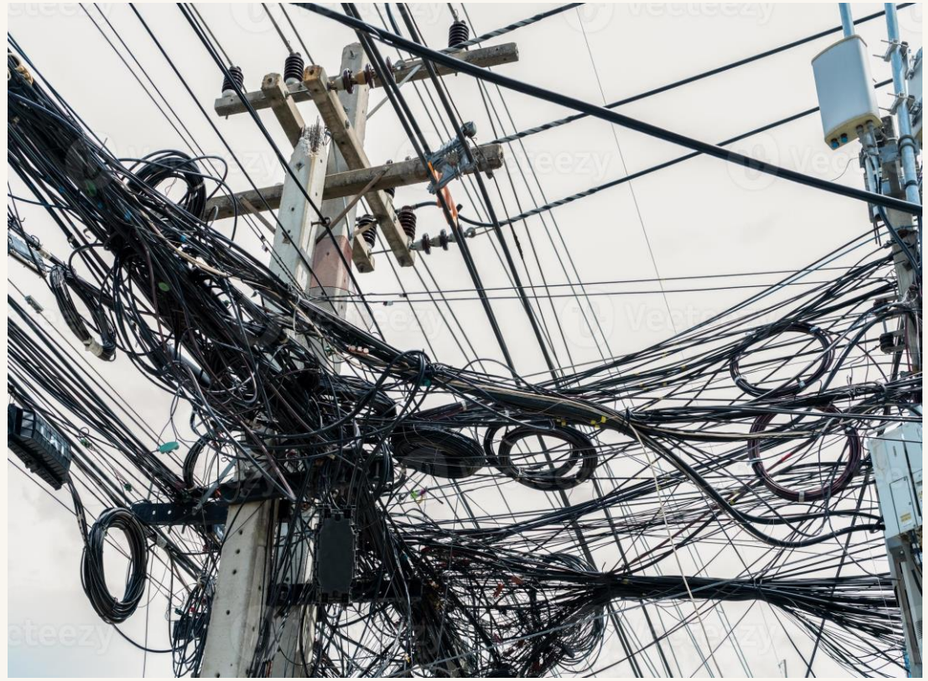
Важность правильного проектирования взаимодействия

- Гибкость и простота изменений

  Слабая связность компонентов и высокая когезия (каждый компонент хорошо выполняет свою единственную задачу) делают систему более адаптируемой к изменениям.

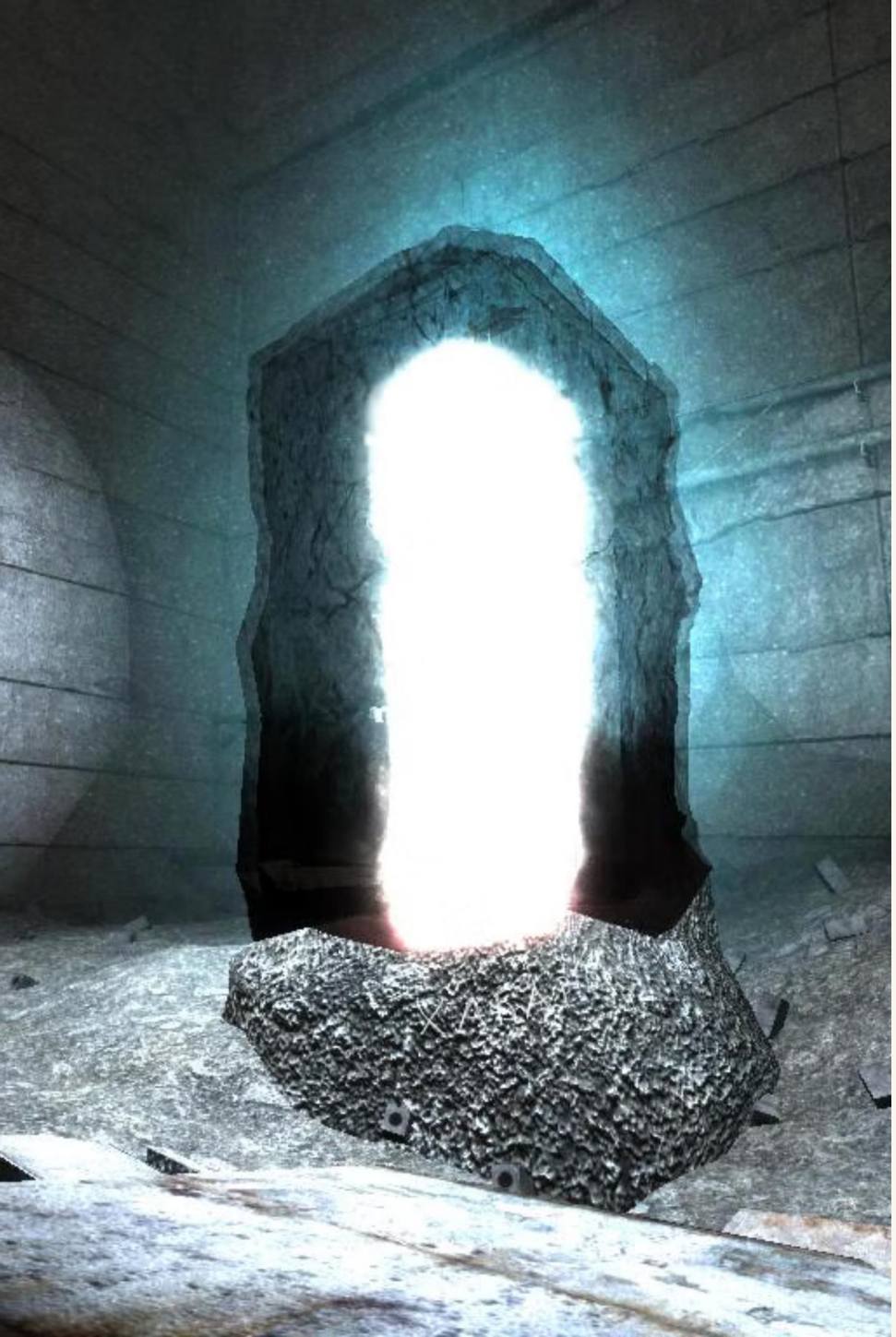
- Упрощение интеграции и поддержки

  Использование стандартизированных протоколов и форматов данных минимизирует сложности при интеграции новых функций и компонентов.



- Последствия неправильной архитектуры

  Неадекватное проектирование ведёт к системным сбоям, высоким операционным затратам, трудностям в масштабировании и негативному пользовательскому опыту.



Заключение

Архитектура — это фундамент, который позволяет создавать эффективные, конкурентоспособные и устойчивые информационные системы, минимизируя риски и затраты на их поддержку и развитие.