RuStore

Клименко Артемий

В сетях сетевого слоя

Mobius 2024

Клименко Артемий

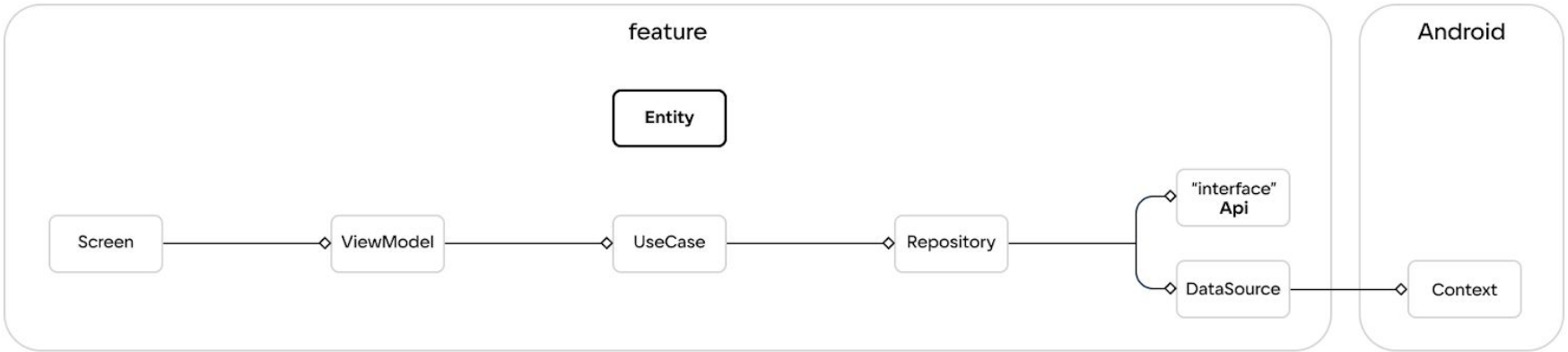
Старший инженер

Об опыте:
В андроид разработке с 2016 года

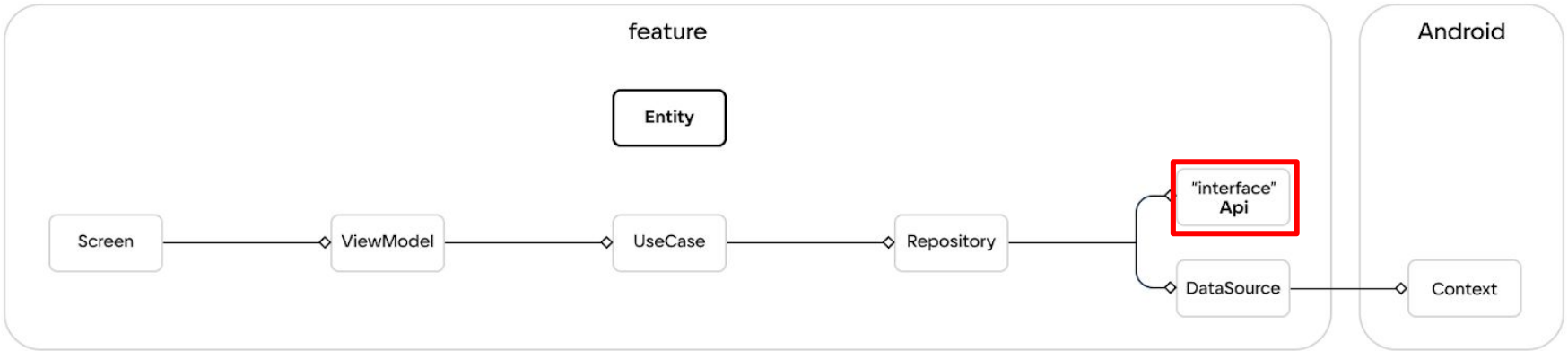
Автор статьи:
Вы за это заплатите!
Цена Чистой Архитектуры
(1 и 2 части)

Хабр - @Artilirium

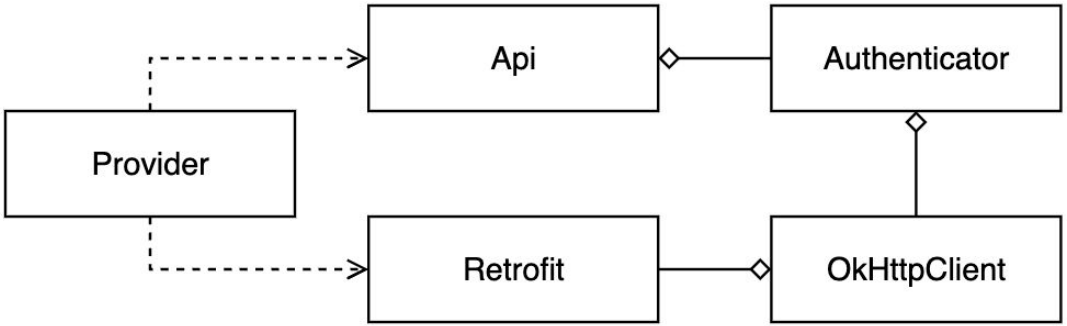
В предыдущей серии



Интерфейс есть, а сети нет



Как организуется работа с сетью?



Мало информации

- Как организуется работа с сетью?
- Где в архитектуре расположен сетевой слой?

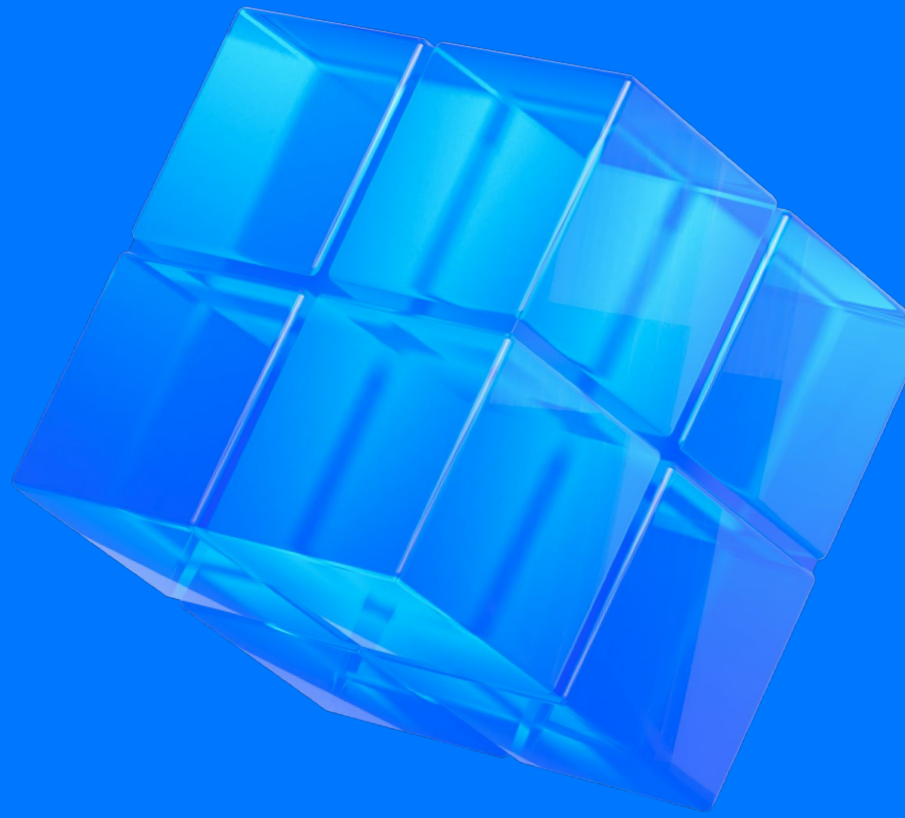
Сеть – это просто

Кодогенерация

Framework-и


Давай. Вошли и вышли, приключение
на 20 минут.

Простая сеть



Мы с друзьями делаем интернет-магазин

Список дел:

- Наверстать экранов (2 дня)
- Добавить корзину (1 день)
- Подобрали нужный цвет кнопке “**купить**” (месяц)

Не работает?

Заглушки
на запросы — это хорошо,
но не для релиза

Настраиваем работу с сетью:

Подключили Famework-и (Retrofit + OkHttp)

Создали Retrofit:

```
Retrofit.Builder()
    .client(OkHttpClient())
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

Добавляем клиент:

```
Retrofit.Builder()
    .client(OkHttpClient())
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

Добавляем базовый URL:

```
Retrofit.Builder()
    .client(OkHttpClient())
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

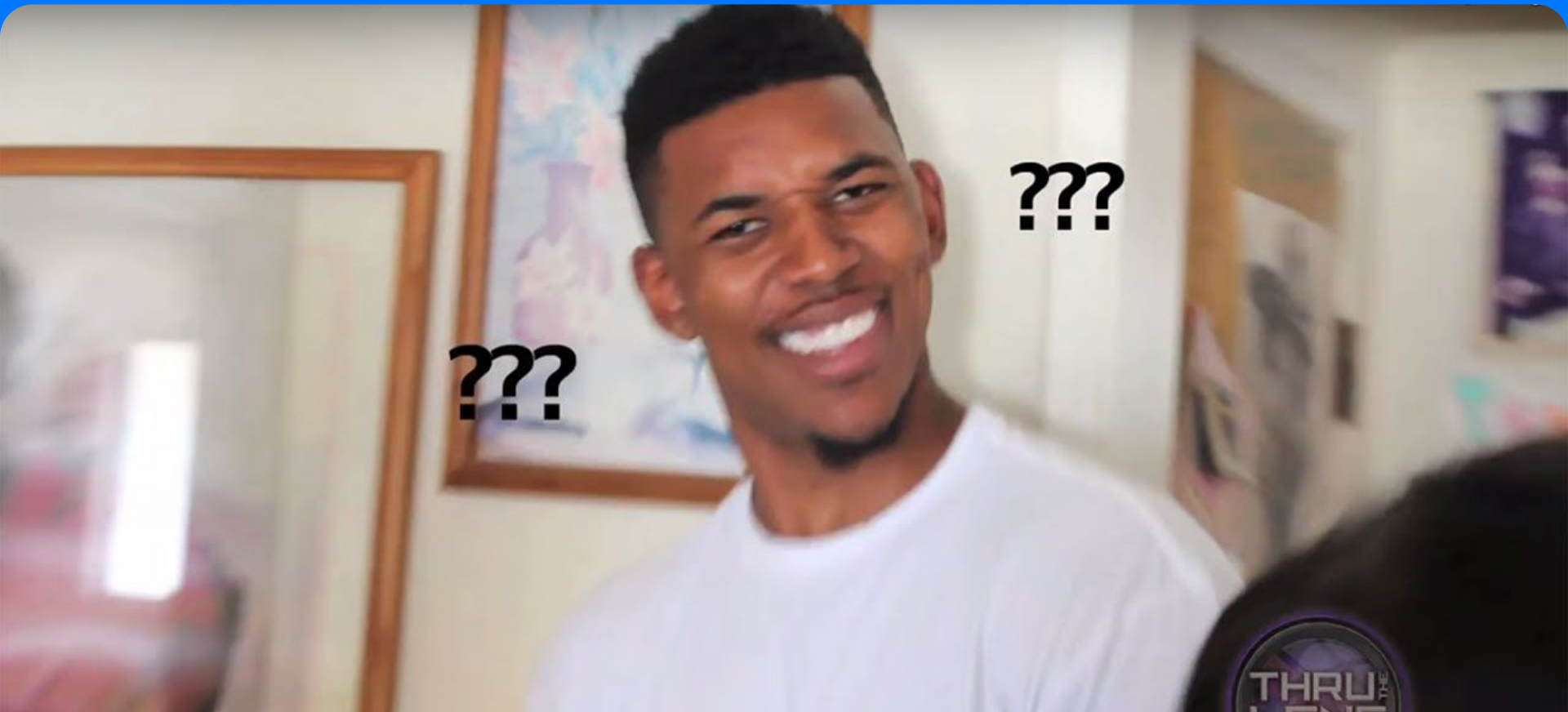
Добавляем конвертацию

```
Retrofit.Builder()
    .client(OkHttpClient())
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

Настраиваем работу с сетью:

Подключили Famework-и (Retrofit + OkHttp);
Создали Retrofit
Создали Api в feature-модулях

Действительно просто, но не заработало



Настраиваем логирование:

Подключили Famework-и (Retrofit + OkHttp);
Создали LoggingInterceptor
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Логи надо скрывать

Подключили Famework-и (Retrofit + OkHttp);
Создали LoggingInterceptor для debug
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Сертификаты

CertPathValidatorException : Trust anchor for certificate path not found

Настраиваем доверие сертификатам

Подключили Famework-и (Retrofit + OkHttp);
Создали LoggingInterceptor для debug
Создали Pinner

Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Давайте сразу обогатим запрос Header-ами

Подключили Famework-и (Retrofit + OkHttp);
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали Pinner

Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Заработало. Всё еще просто.

Появился тестовый стенд

Теперь нужно

Менять стенды
в зависимости
от сборки

Настраивать
доверие
сертификатам

Добавлять
отдельные
хедеры для debug
(опционально)

Меняем URL-ы:

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor

Создали Pinner
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Настраиваем доверие сертификатам (еще раз):

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor

Создали Pinner для release
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Настраиваем доверие сертификатам (еще раз):

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали Pinner для release

Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Отдельные Header-ы для debug:

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug
Создали Pinner для release
Создали TrustManager для debug

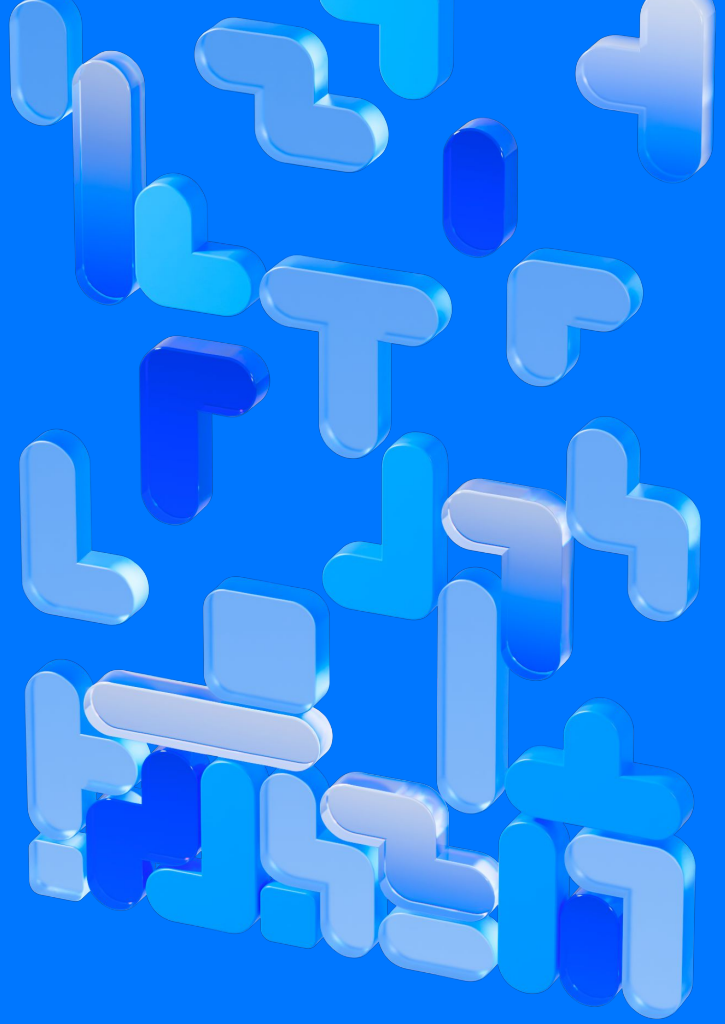
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Всё ещё просто и всё работает

только вот пользователей между собой мы не различаем

Простая сеть

Анонимная сессия



Анонимная сессия

- Не требует авторизации
- Позволяет различать пользователей между собой
- Существует ограниченное время

Проблемы выбора

Кто создает?

Сервер или клиент?

Проблемы выбора

Кто создает?

Сервер или клиент?

В какие запросы добавлять?

Определять
или слать везде?

Проблемы выбора

Кто создает?

Сервер или клиент?

В какие запросы добавлять?

Определять
или слать везде?

Где создавать и обновлять?

Определять
или в Authenticator-е?

Кто создает?

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AnonymousSessionApi
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug
Создали Pinner для release

Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

В какие запросы добавлять?

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AnonymousSessionApi
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug
Создали Pinner для release

Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Где создавать и обновлять?

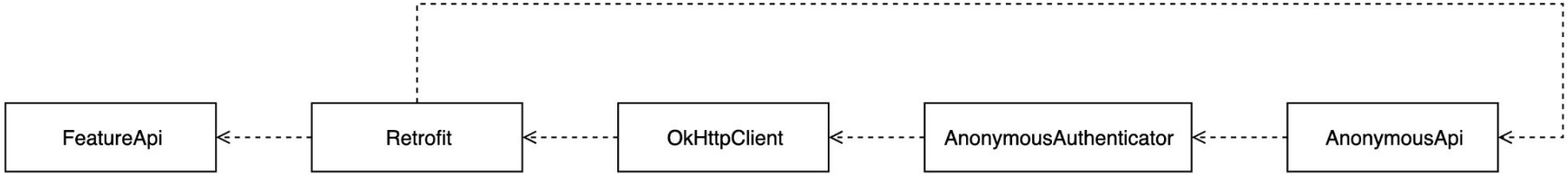
Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AnonymousSessionApi
Создали AnonymousSessionAuthenticator
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug
Создали Pinner для release

Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient
Создали Retrofit
Создали Api в feature-модулях

Всё еще просто, но не заработало

error: [Dagger/DependencyCycle] Found a dependency cycle:

Откуда?


FeatureApi
Retrofit
OkHttpClient
AnonymousAuthenticator
AnonymousApi

Что делать?

- Разный состав
  компонентов для OkHttpClient-a
- Разные
  экземпляры Retrofit-a

Добавили условия создания

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AnonymousSessionApi
Создали AnonymousSessionAuthenticator
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug
Создали Pinner для release

Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient для Anonymous и feature
Создали Retrofit для Anonymous и feature
Создали Api в feature-модулях

Да вроде просто. Главное работает.

А давайте добавим программу лояльности

- Нужно нечто большее,
  чем анонимная сессия

Простая сеть

Авторизованная сессия

Авторизованная сессия

- Позволяет различать пользователей неограниченное время
- Требует вмешательство пользователя (нужен экран авторизации)

Авторизованная сессия (JWT - стандарт)

- accessToken
- refreshToken

Добавили Авторизованную сессию в заголовки

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали Anonymous Session Api
Создали AnonymousSessionAuthenticator
Создали AuthorizedSessionInterceptor
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug

Создали Pinner для release
Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient для Anonymous и feature
Создали Retrofit для Anonymous и feature
Создали Api в feature-модулях

Добавили обновление Авторизованной сессии в Authenticator-е

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AuthorizedSessionApi
Создали AnonymousSessionApi
Создали AnonymousSessionAuthenticator
Создали AuthorizedSessionInterceptor
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug

Создали Pinner для release
Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient для Anonymous и feature
Создали Retrofit для Anonymous и feature
Создали Api в feature-модулях

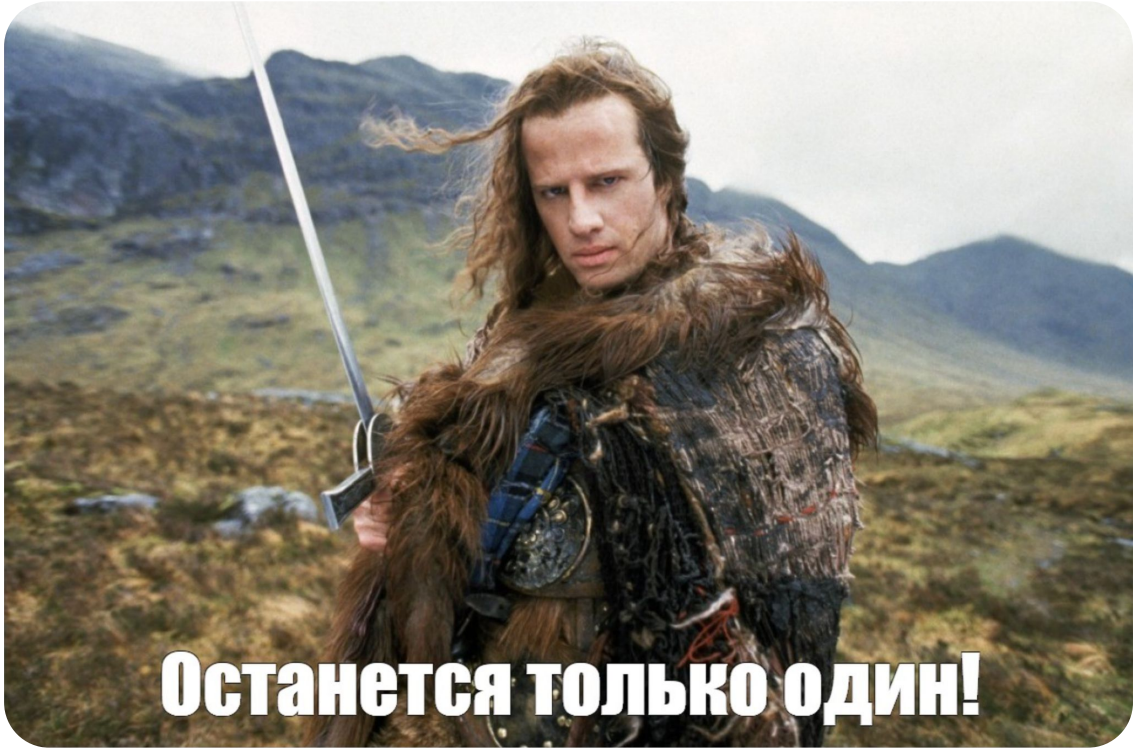
Добавили обновление
Авторизованной сессии в Authenticator-е

Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AnonymousSessionApi
Создали AuthorizedSessionApi
Создали AuthorizedSessionAuthenticator
Создали AnonymousSessionAuthenticator
Создали AuthorizedSessionInterceptor
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug

Создали DeviceInfoInterceptor
Создали DebugInterceptor для debug
Создали Pinner для release
Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient для Anonymous и feature
Создали Retrofit для Anonymous и feature
Создали Api в feature-модулях

Проблема

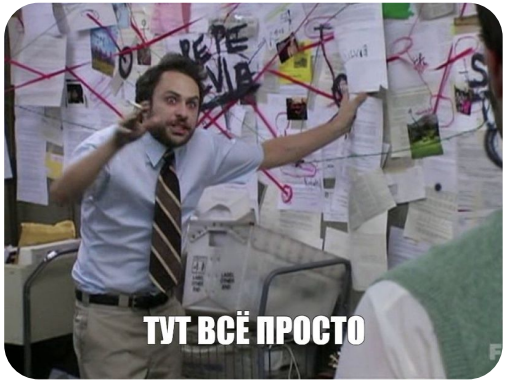
- Аутентификатор
  может быть только один



Не проще ли было отказаться от Анонимной сессии?

- В нашей ситуации авторизация не должна быть принудительной

Добавили создание OkHttpClient-а и Retrofit-а для Авторизованной сессии



Подключили Famework-и (Retrofit + OkHttp);
Выбираем URL для debug и release
Создали AnonymousSessionApi
Создали AuthorizedSessionApi
Создали AuthorizedSessionAuthenticator
Создали AnonymousSessionAuthenticator
Создали AuthorizedSessionInterceptor
Создали AnonymousSessionInterceptor
Создали LoggingInterceptor для debug
Создали DeviceInfoInterceptor

Создали DebugInterceptor для debug
Создали Pinner для release
Создали TrustManager для debug
Создали SSLSocketFactory для debug
Создали HostNameVerifire для debug
Создали OkHttpClient для Anonymous, Authorized и feature
Создали Retrofit для Anonymous, Authorized и feature
Создали Api в feature-модулях

Работу с сетью закончили?

Хотим добавить рекомендации

- Одно неловкое движение
  и авторизация говорит нам

Неправильных решений много

Общие черты

- Нелегко воспроизвести
- Тяжело поддержать
- Невозможно забыть

Проблемы

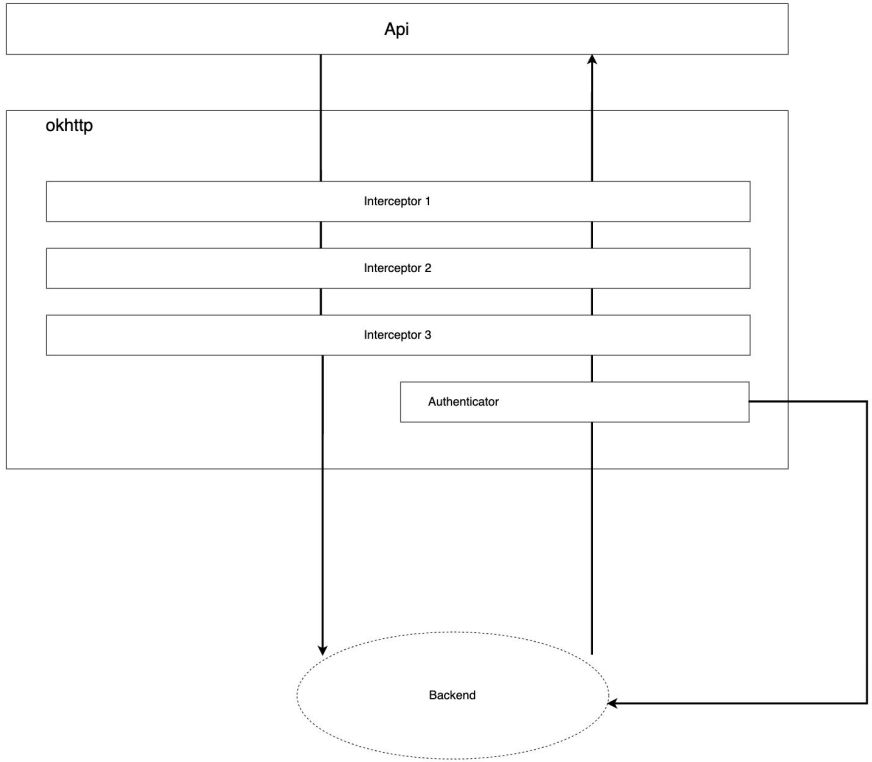
- Дебажный код в релизных сборках
- Лишние запросы
- Лишние Header-ы
- Сложность обеспечение уникальных header-ов
- Непредумышленная DoS-атака сервера
- Сложность обеспечения логирования
- Потенциал на другие проблемы

А что не так с логированием?

- Дебажный код в релизных сборках
- Лишние запросы
- Лишние Header-ы
- Сложность обеспечение уникальных header-ов
- Непредумышленная DoS-атака сервера
- **Сложность обеспечения логирования**
- Потенциал на другие проблемы

Обеспечение полного логирования

- Порядок Interceptor-ов
- Authenticator



Как решить наши проблемы?

Причины проблем

Большое количество условий

- Разные сборки
- Разные способы авторизации

Решение

Безусловная сборка

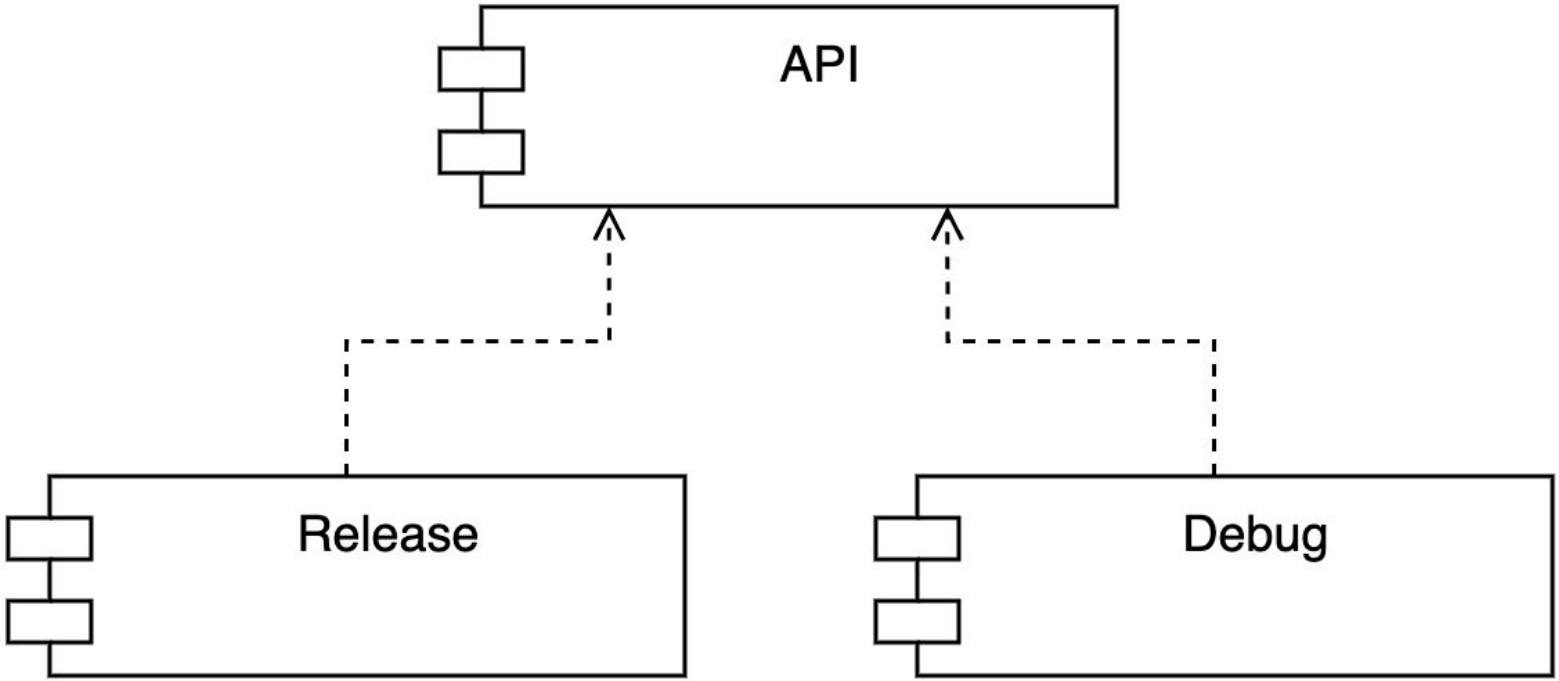
Безусловная сборка

- **implementation**

Безусловная сборка

- implementation
- **debug**Implementation
- **release**Implementation

Безусловная сборка



Решение

Зоны авторизации

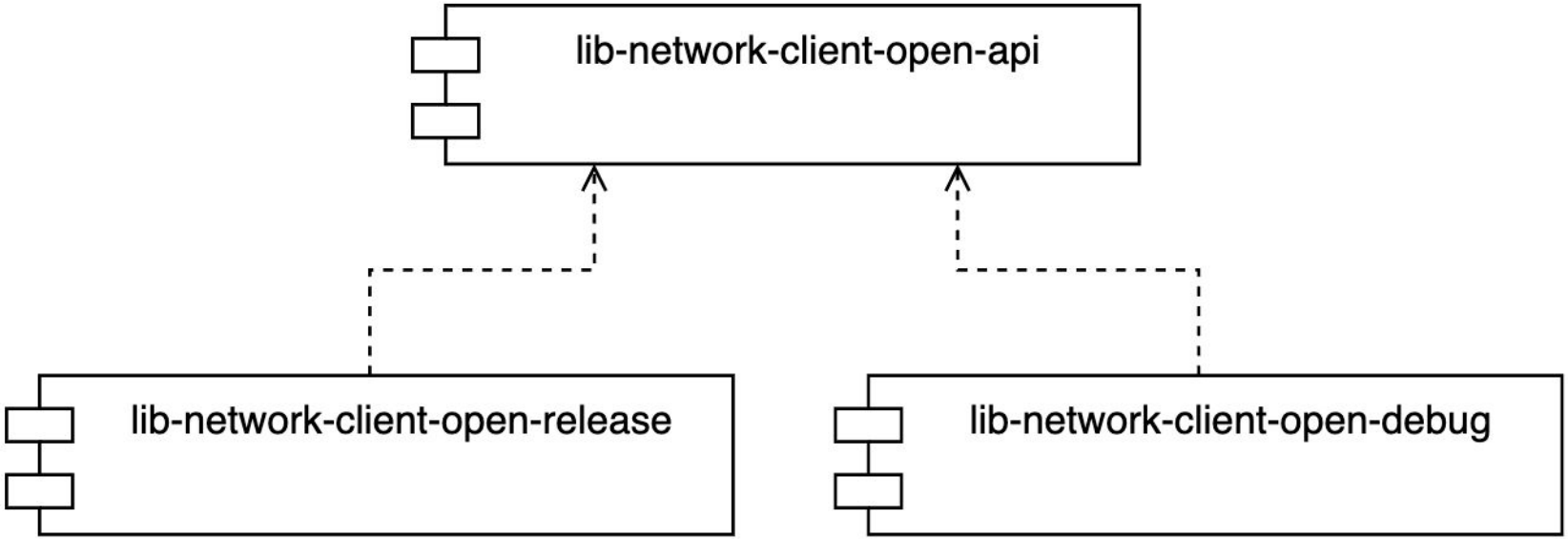
Зона авторизации

Пространство запросов,
права на которые предоставляются
в рамках сессии

Зоны авторизации

- Открытая
- Анонимная
- Авторизованная
- Адаптивная

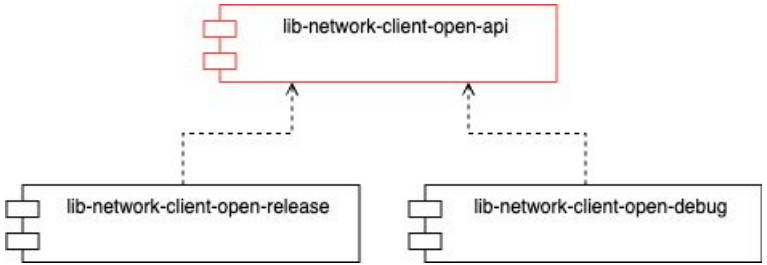
Открытый OkHttpClient (безусловный)



Открытый OkHttpClient (безусловный)

```
interface OpenClientProvider {

    fun get(): OkHttpClient
}
```

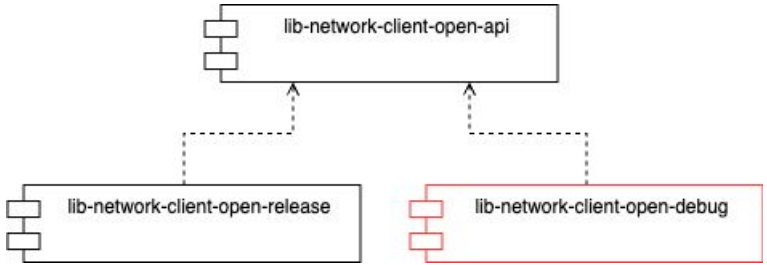


Открытый OkHttpClient (безусловный)

```
internal class OpenClientProviderDebug(
    loggingInterceptor: HttpLoggingInterceptor,
    deviceInfoInterceptor: DeviceInfoInterceptor,
    debugHeaderInterceptor: DebugHeaderInterceptor,
    trustManager: X509TrustManager,
    sslSocketFactory: SSLSocketFactory,
    hostNameVerifier: HostnameVerifier,
) : OpenClientProvider {

    private val client = OkHttpClient.Builder()
        .addInterceptor(deviceInfoInterceptor)
        .addInterceptor(debugHeaderInterceptor)
        .addInterceptor(loggingInterceptor)
        .sslSocketFactory(sslSocketFactory, trustManager)
        .hostnameVerifier(hostNameVerifier)
        .build()

    override fun get(): OkHttpClient =
        client
}
```

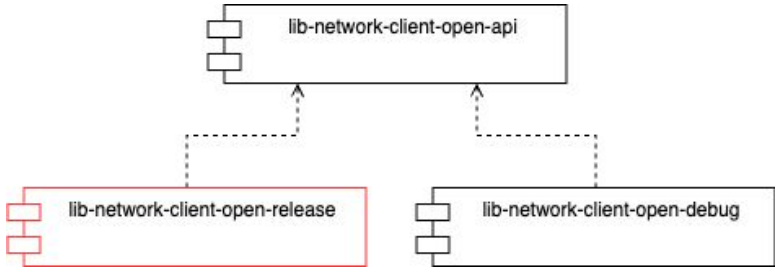


Открытый OkHttpClient (безусловный)

```
internal class OpenClientProviderRelease(
    deviceInfoInterceptor: DeviceInfoInterceptor,
    certificatePinner: CertificatePinner,
) : OpenClientProvider {

    private val client = OkHttpClient.Builder()
        .addInterceptor(deviceInfoInterceptor)
        .certificatePinner(certificatePinner)
        .build()

    override fun get(): OkHttpClient =
        client
}
```



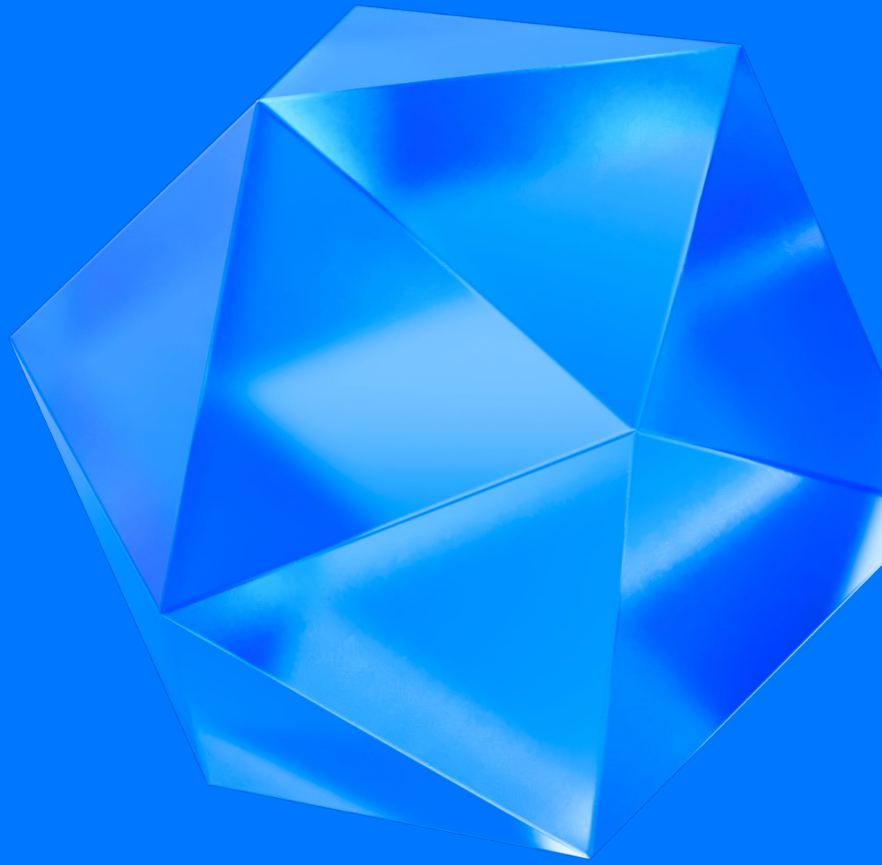
Повторять до готовности

Всего 12 модулей

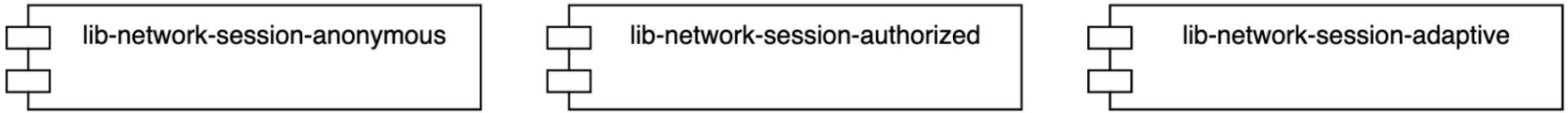
- 4 реализации
  для release
- 4 реализации
  для debug
- 4 api

Решение

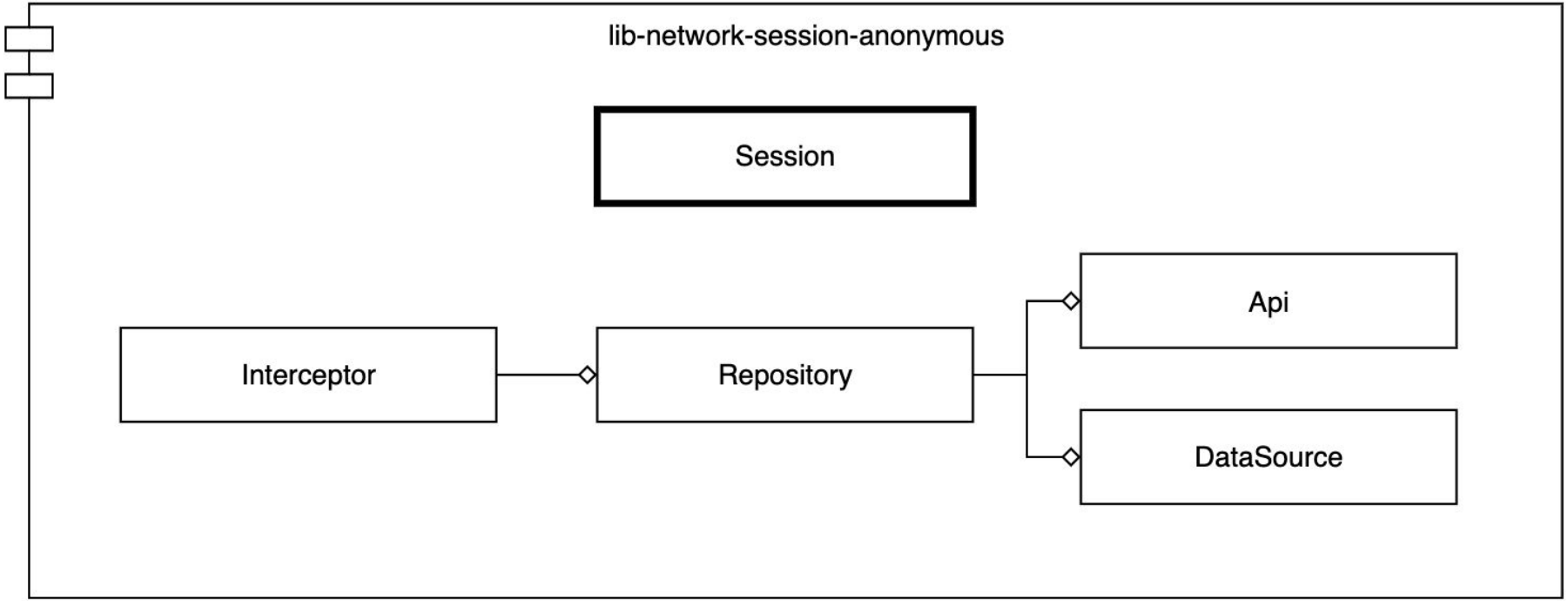
Самостоятельные сессии



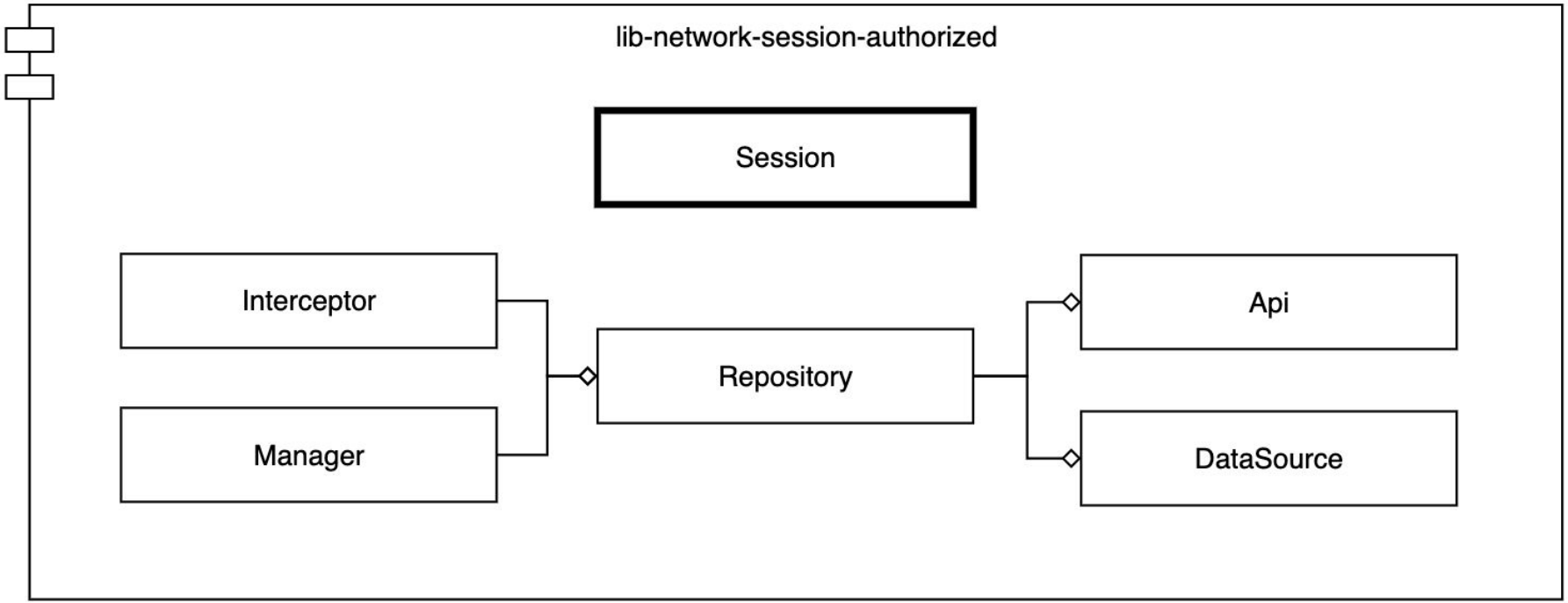
Сессии


lib-network-session-anonymous
lib-network-session-authorized
lib-network-session-adaptive

Анонимная сессия



Авторизованная сессия



AuthorizedSessionManager

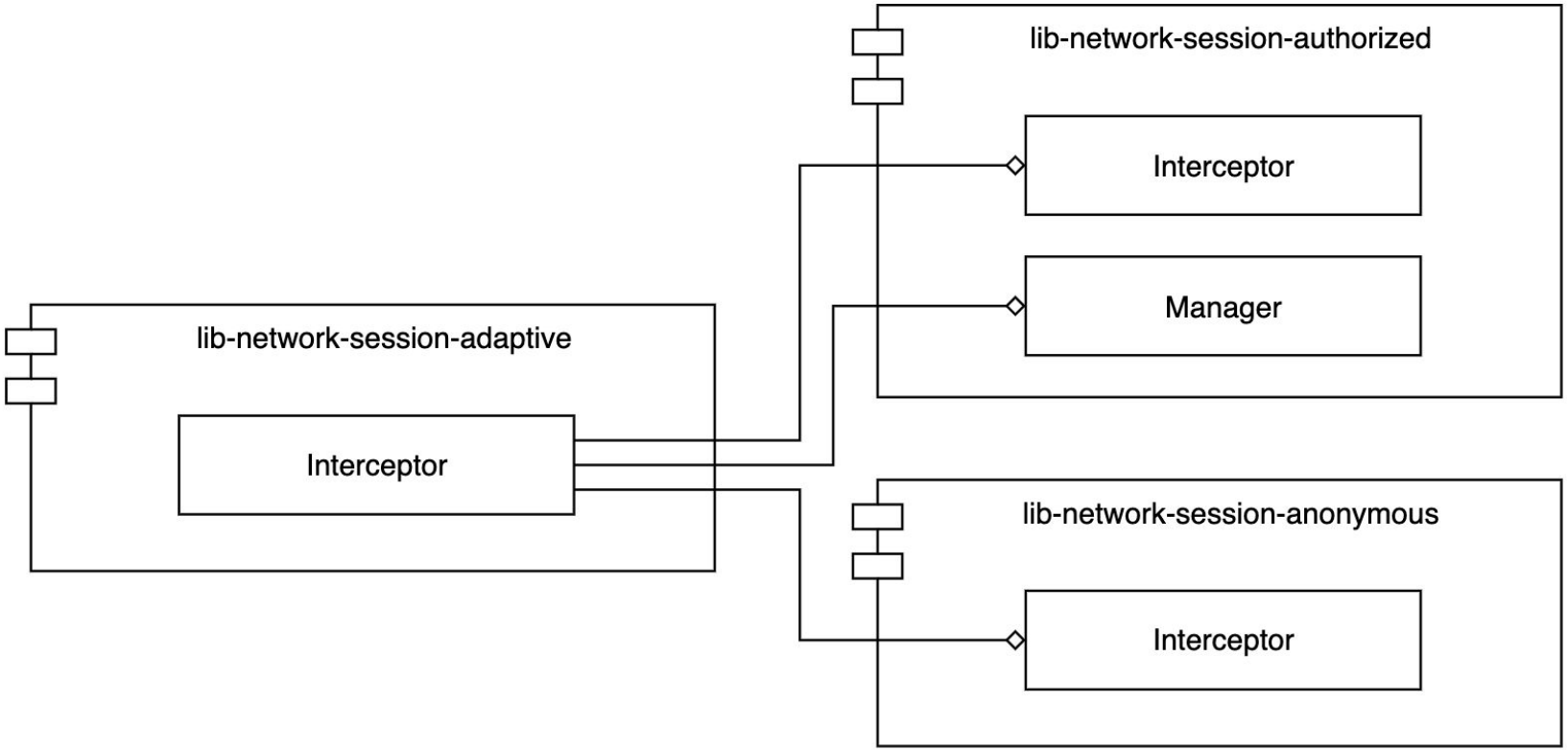
```
class AuthorizedSessionManager @Inject internal constructor(
    private val repository: AuthorizedSessionRepository,
) {

    fun isAuthorized(): Boolean =
        repository.get() != null

    fun set(session: AuthorizedSession) {
        repository.set(session)
    }

    fun clear() {
        repository.clear()
    }
}
```

Адаптивная сессия



Адаптивная сессия

```
class AdaptiveSessionInterceptor @Inject constructor(
    private val authorizedSessionManager: AuthorizedSessionManager,
    private val authorizedInterceptor: AuthorizedSessionInterceptor,
    private val anonymousInterceptor: AnonymousSessionInterceptor,
) : Interceptor {

    override fun intercept(chain: Interceptor.Chain): Response =
        if (authorizedSessionManager.isAuthorized()) {
            authorizedInterceptor.intercept(chain)
        } else {
            anonymousInterceptor.intercept(chain)
        }
}
```

Адаптивная сессия

```
class AdaptiveSessionInterceptor @Inject constructor(
    private val authorizedSessionManager: AuthorizedSessionManager,
    private val authorizedInterceptor: AuthorizedSessionInterceptor,
    private val anonymousInterceptor: AnonymousSessionInterceptor,
) : Interceptor {

    override fun intercept(chain: Interceptor.Chain): Response =
        if (authorizedSessionManager.isAuthorized()) {
            authorizedInterceptor.intercept(chain)
        } else {
            anonymousInterceptor.intercept(chain)
        }
}
```

Адаптивная сессия

```
class AdaptiveSessionInterceptor @Inject constructor(
    private val authorizedSessionManager: AuthorizedSessionManager,
    private val authorizedInterceptor: AuthorizedSessionInterceptor,
    private val anonymousInterceptor: AnonymousSessionInterceptor,
) : Interceptor {

    override fun intercept(chain: Interceptor.Chain): Response =
        if (authorizedSessionManager.isAuthorized()) {
            authorizedInterceptor.intercept(chain)
        } else {
            anonymousInterceptor.intercept(chain)
        }
}
```

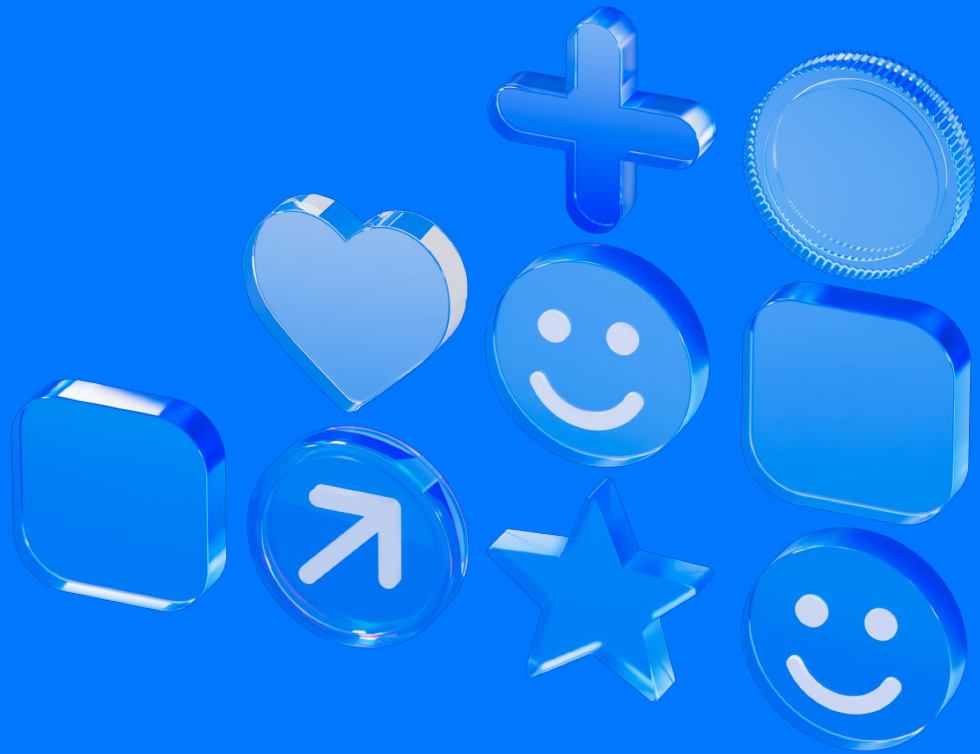
Применение

```
@Singleton
@Provides
fun provideFeatureApi(
    clientProvider: AnonymousClientProvider,
    urlProvider: UrlProvider,
    converter: Converter.Factory,
): FeatureApi =
    Retrofit.Builder()
        .client(clientProvider.get())
        .baseUrl(urlProvider.getBaseUrl())
        .addConverterFactory(converter)
        .build()
        .create()
```

Применение

```
@Singleton
@Provides
fun provideFeatureApi(
    clientProvider: AnonymousClientProvider,
    urlProvider: UrlProvider,
    converter: Converter.Factory,
): FeatureApi =
    Retrofit.Builder()
        .client(clientProvider.get())
        .baseUrl(urlProvider.getBaseUrl())
        .addConverterFactory(converter)
        .build()
        .create()
```

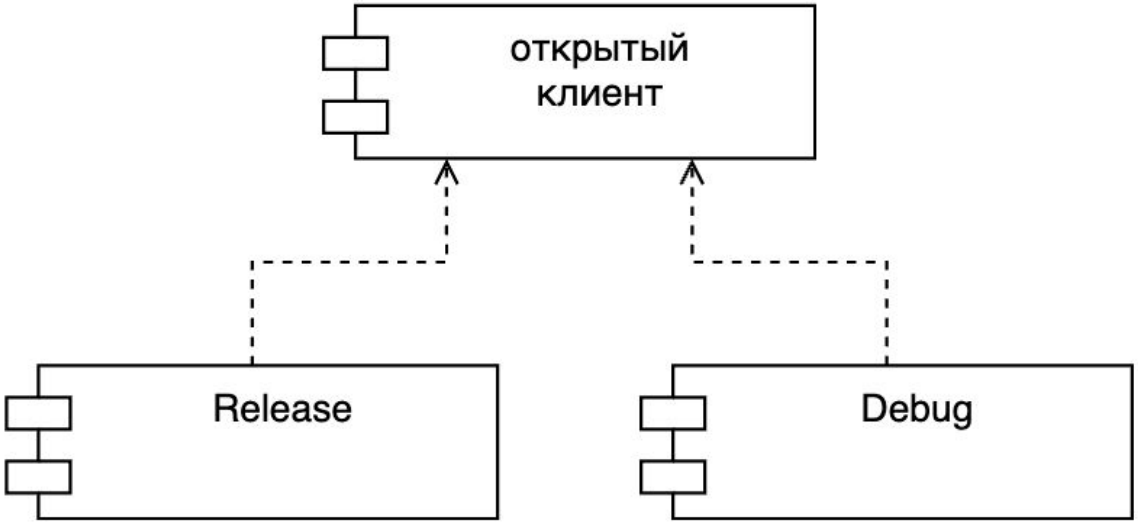
Архитектура сетевого слоя



Сетевой слой

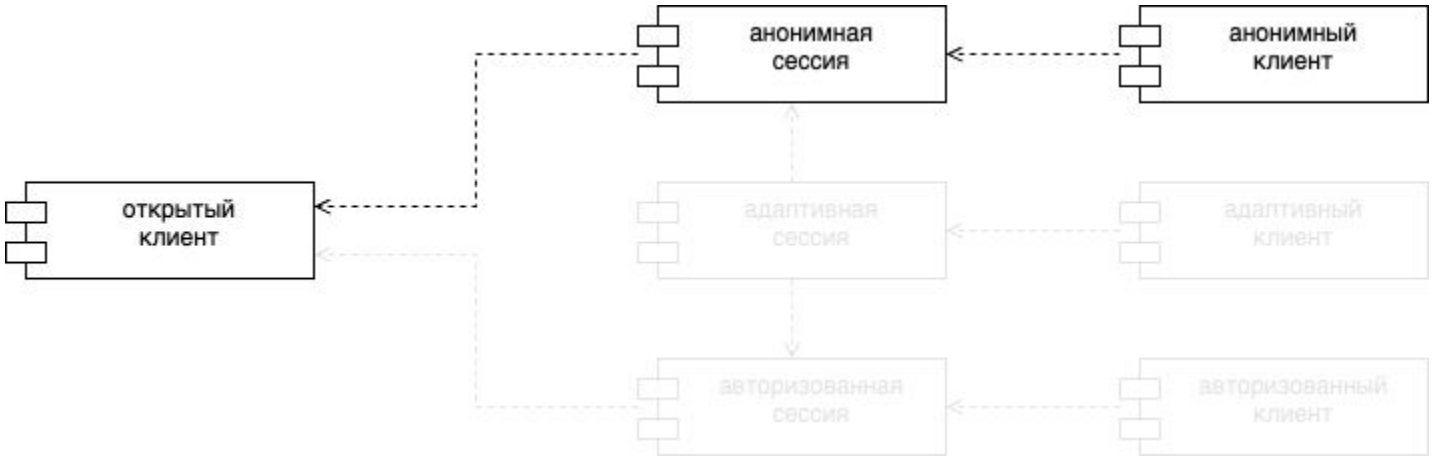
Совокупность компонентов, предназначенных для организации работы с сетью

Архитектура компонентов сетевого слоя

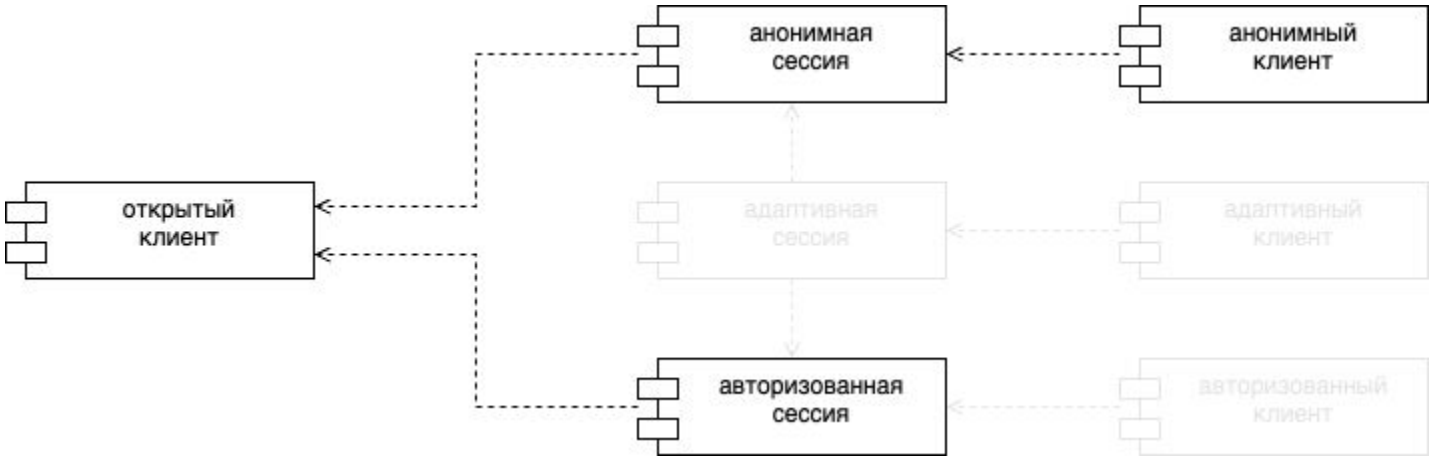


Архитектура компонентов сетевого слоя

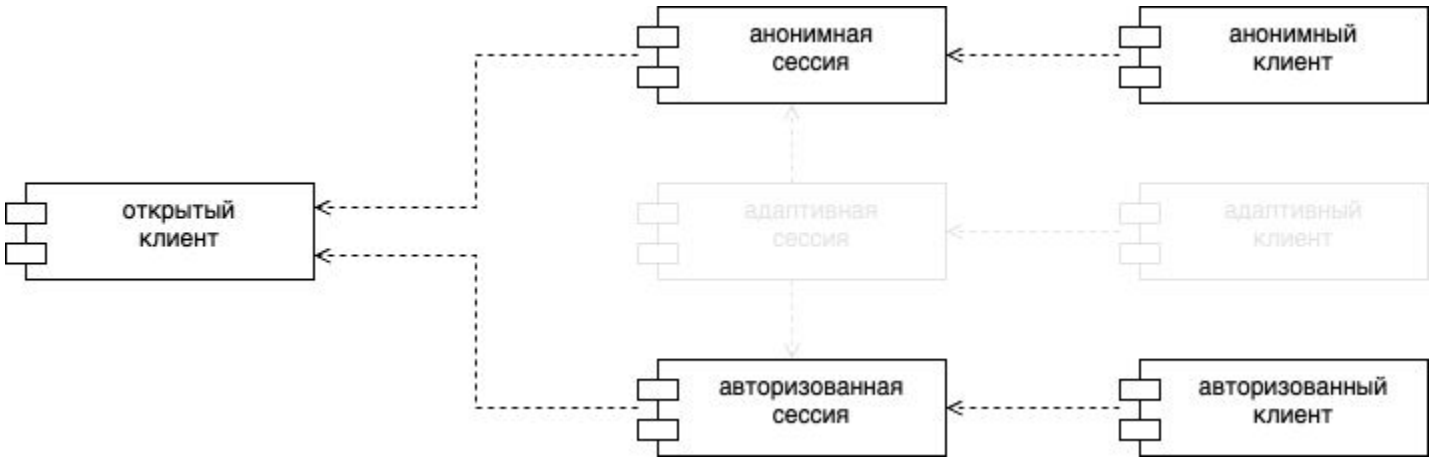
Архитектура компонентов сетевого слоя



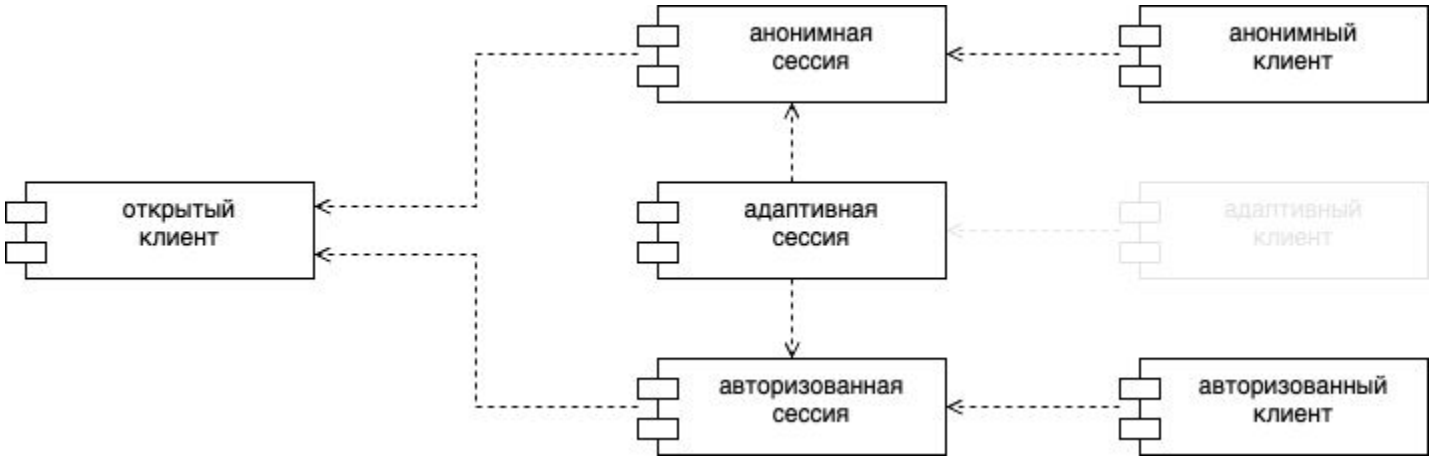
Архитектура компонентов сетевого слоя



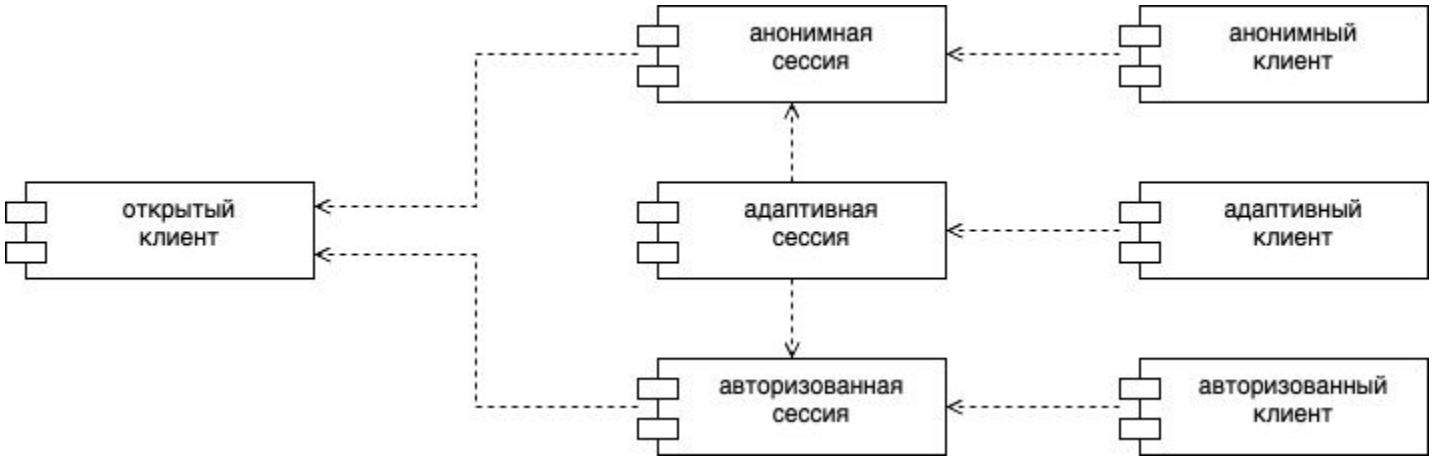
Архитектура компонентов сетевого слоя



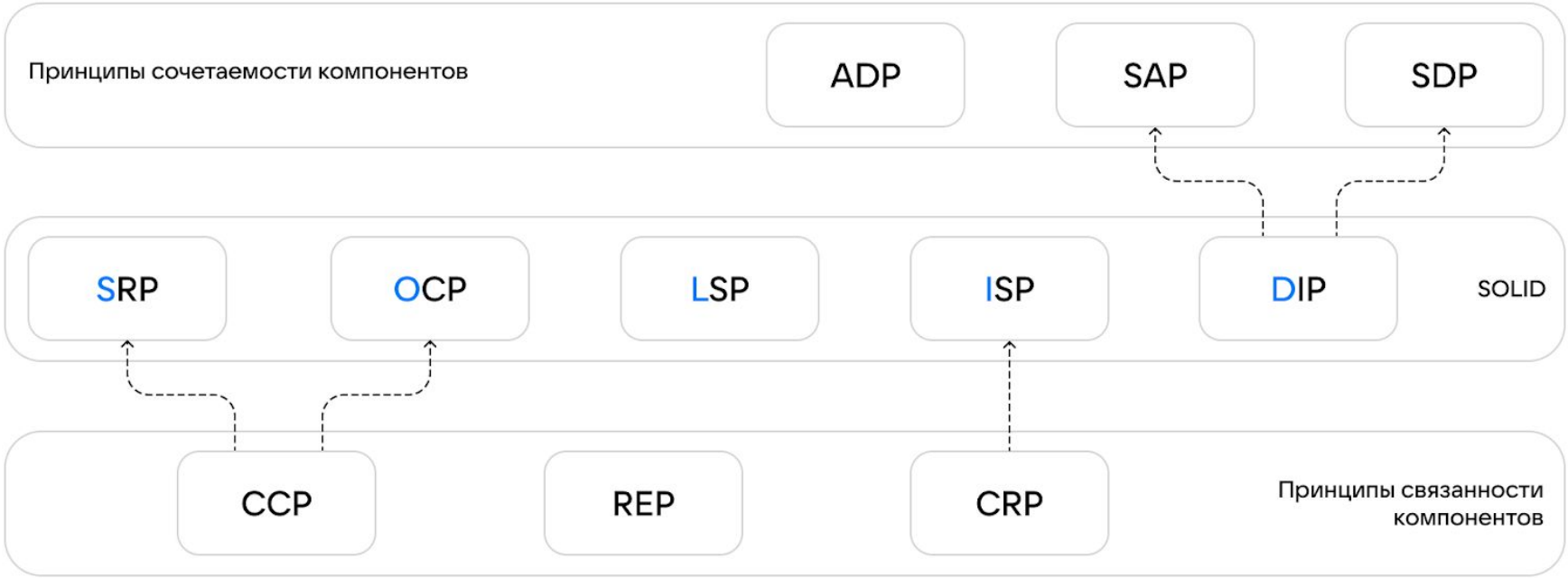
Архитектура компонентов сетевого слоя



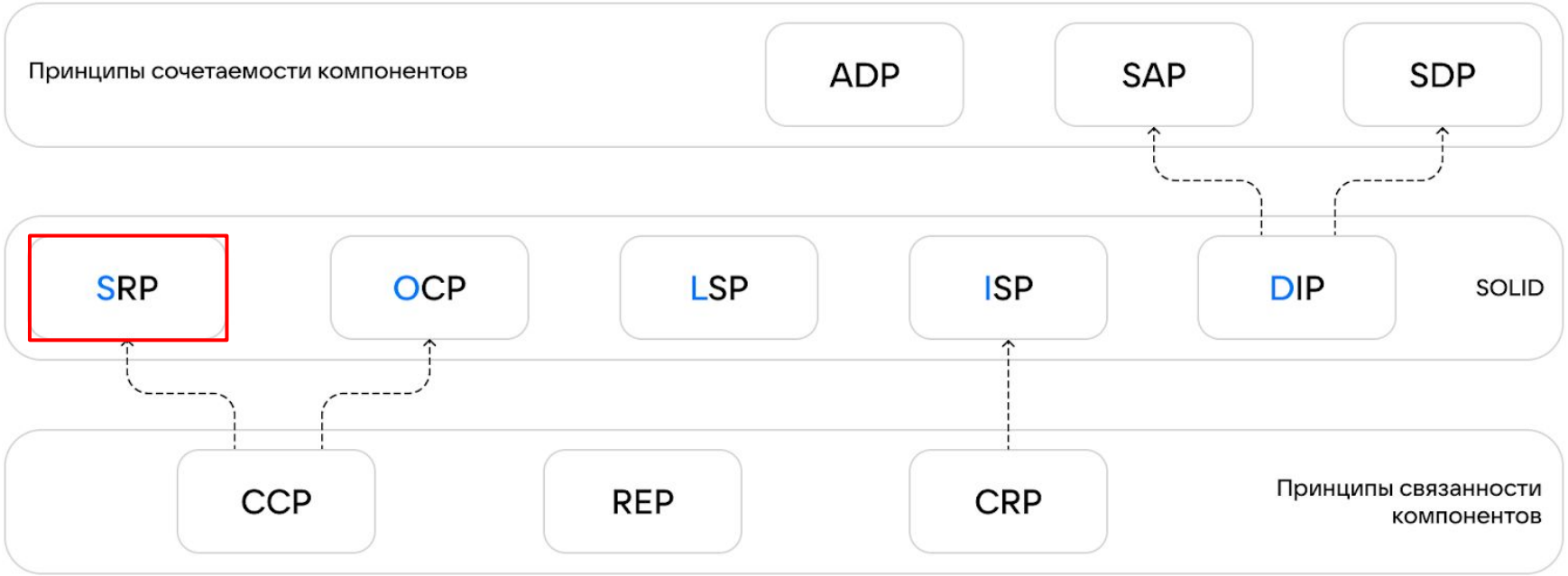
Архитектура компонентов сетевого слоя



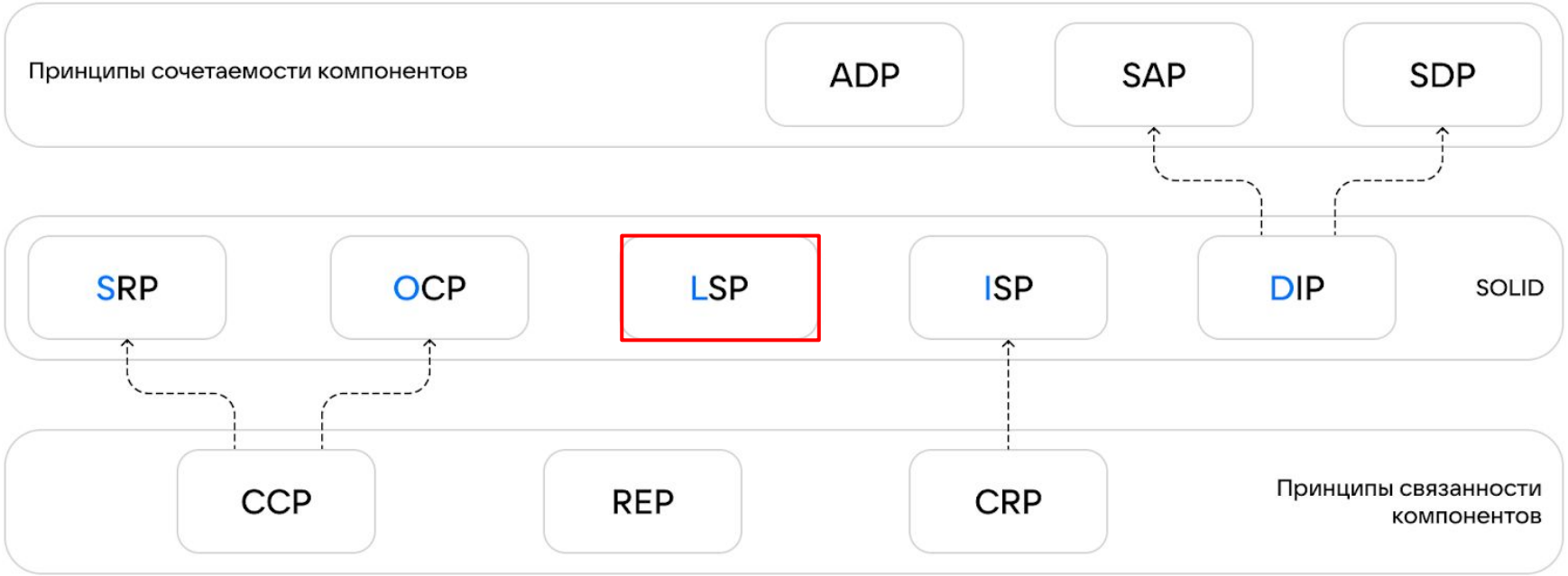
Принципы Чистой Архитектуры



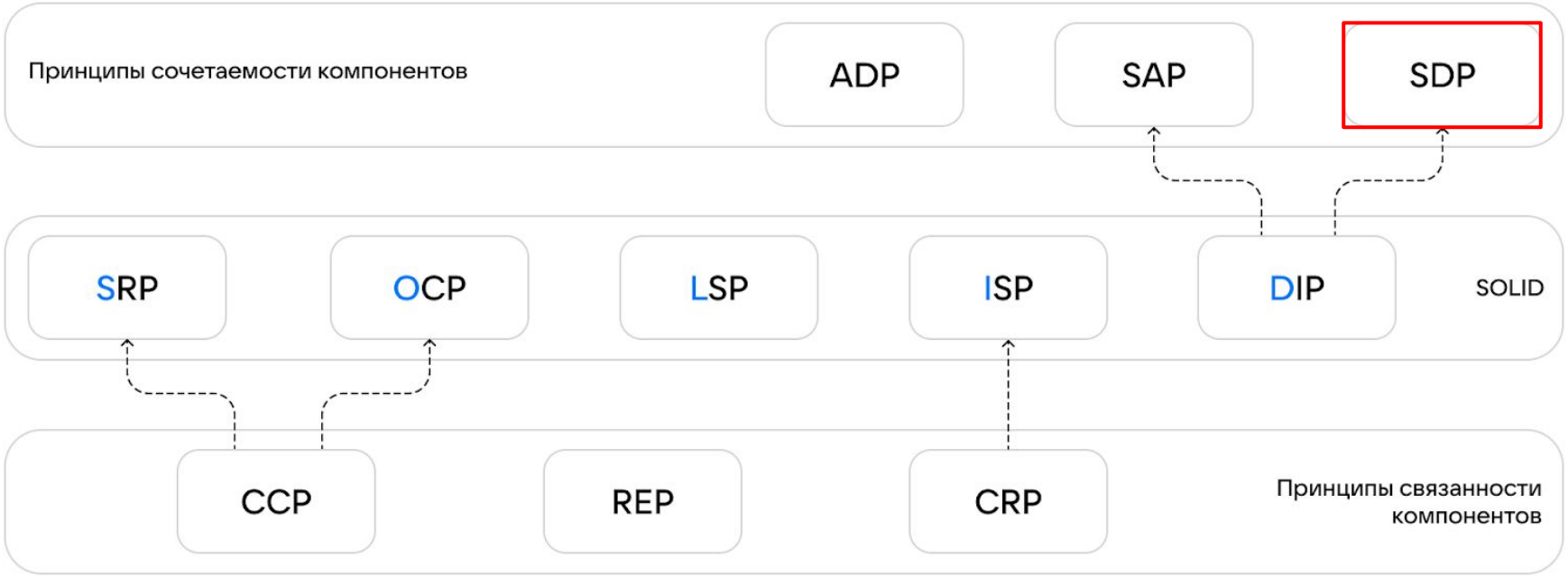
Принципы Чистой Архитектуры



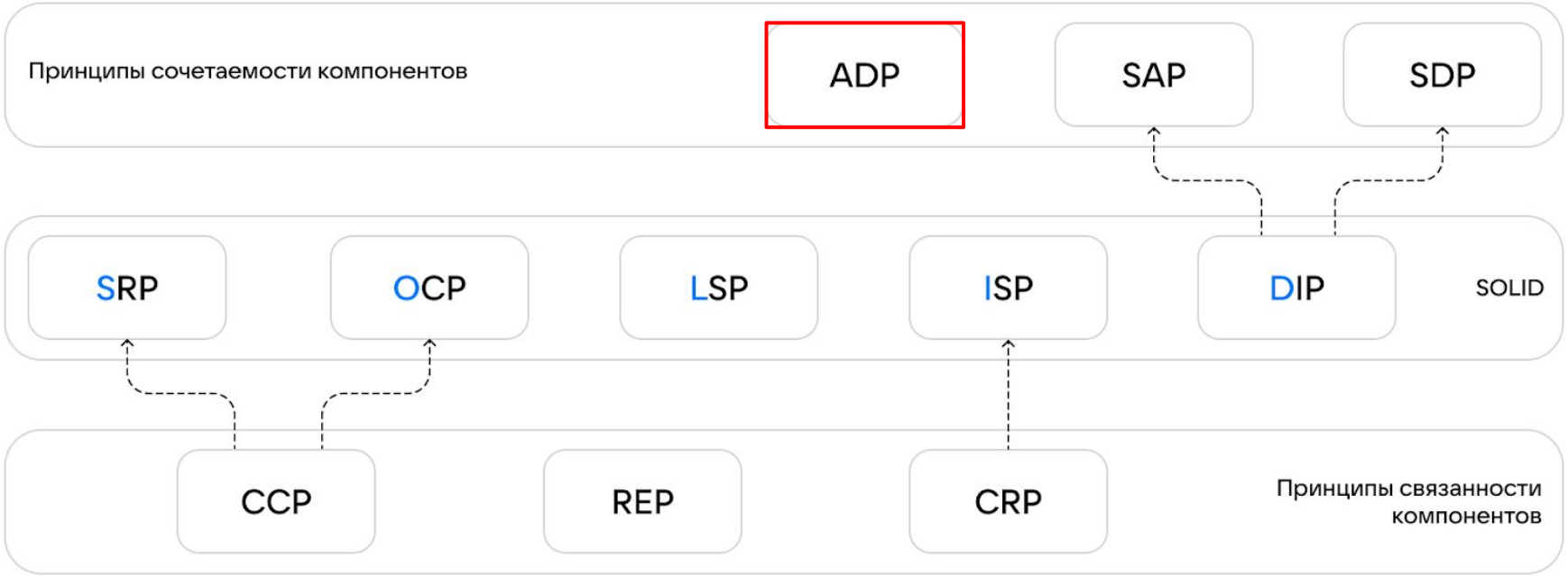
Принципы Чистой Архитектуры

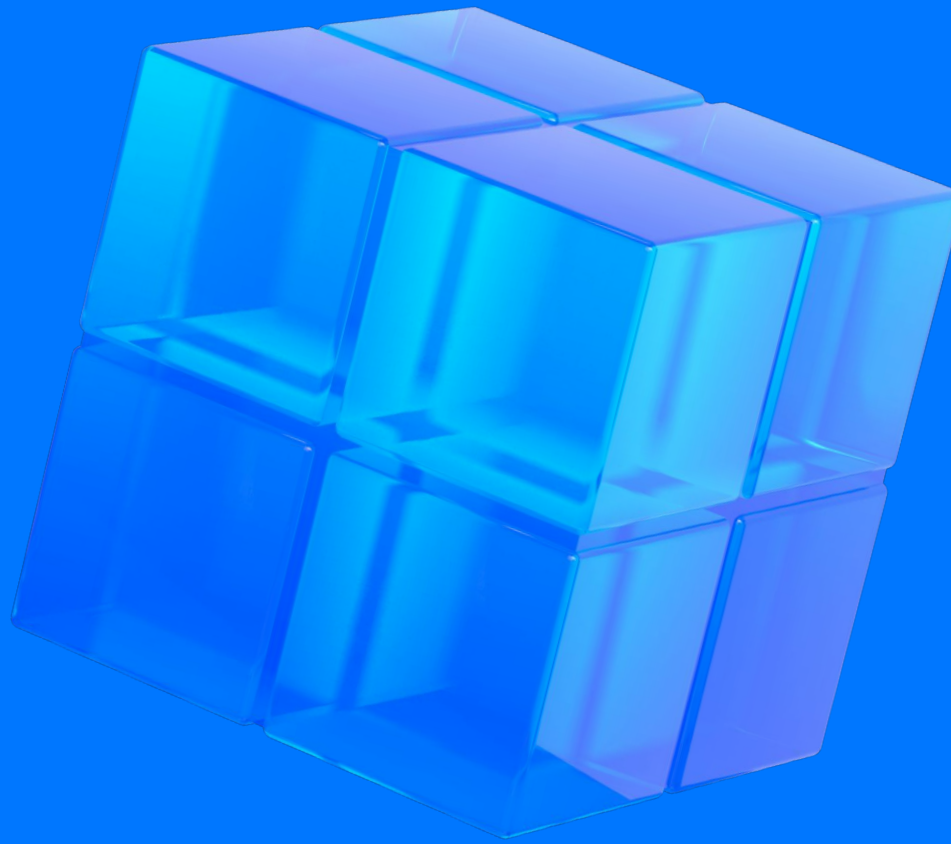


Принципы Чистой Архитектуры



Принципы Чистой Архитектуры





Выводы

Выводы

- Архитектура сетевому слою нужна
- Сетевой слой зависит от Framework-ов
- Сетевой слой зависит от собственных компонентов сети
- Сетевой слой формируется под каждый запрос

Спасибо!

Ссылки

Выступление

Статья (часть 1)

Статья (часть 2)