

STRV

ANATOMY OF AN SDK ECOSYSTEM

Michal Jenicek

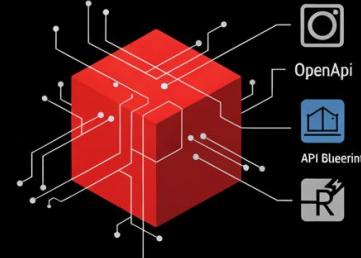
THE MISSION

THE CHALLENGE

- **Complexity:** An SDK isn't just a library. It's a distributed system living inside a hostile host app.

THE APPROACH

- **Deconstruction:** Today, we will dissect a production-grade SDK ecosystem layer by layer to reveal its internal mechanics.

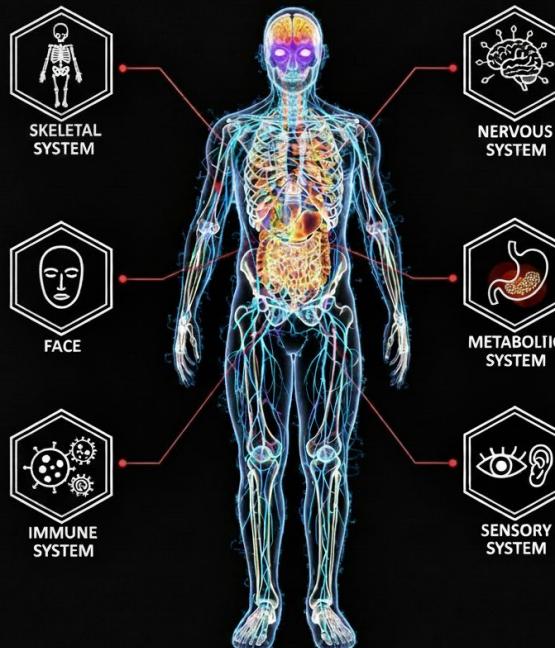


THE SDK ECOSYSTEM

The SDK is more than code.

It has anatomy.

*Deconstructing the Ecosystem,
layer by layer.*



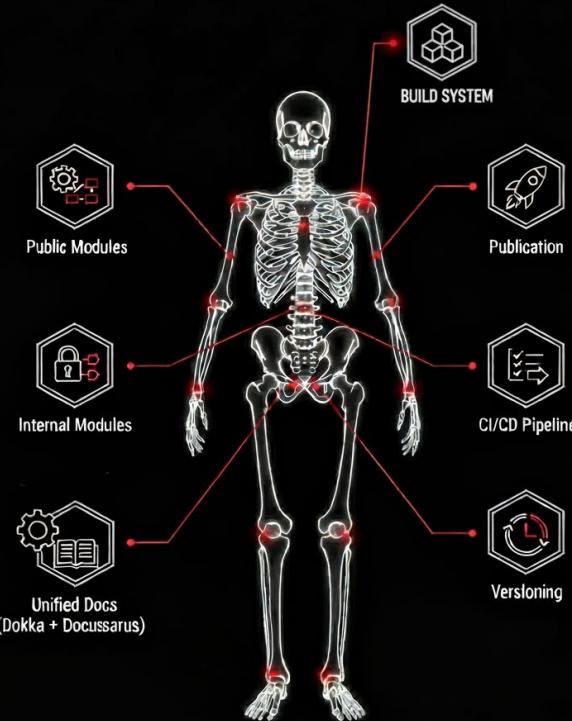
THE SKELETAL SYSTEM

01

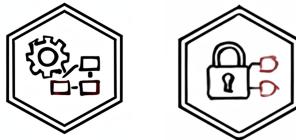
01 • THE SKELETAL SYSTEM

*Provides the **essential framework** and gives the ecosystem its structure.*

- Modules
- Versioning
- Publication
- Automation



MODULES



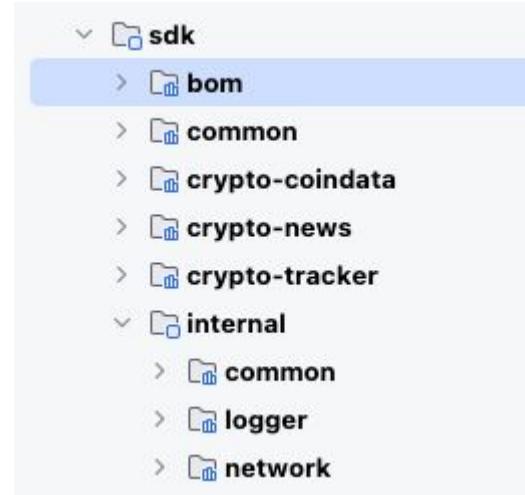
01 SKELETAL

THE STRATEGY

- **Segregating** public vs. internal logic
- **Centralizing** shared domain in *common*
- **Exposing** only feature modules

LESSONS LEARNED

- Strict visibility
- DRY Principle



VERSIONING



01 SKELETAL

THE PAIN POINTS

- Establishing clear traceability

semver.org

OUR SOLUTION

Semantic Versioning 2.0.0

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes
2. MINOR version when you add functionality in a backward compatible manner
3. PATCH version when you make backward compatible bug fixes

VERSIONING



THE PAIN POINTS

- Establishing clear traceability
- Need for guaranteed integrity

OUR SOLUTION

- BOM Integration (*Maven repo*)

```
create< MavenPublication> ( name = "bom" ) {  
    groupId = this.groupId  
    artifactId = this.name  
    version = getLocalPropertyValue( key = "sdk.bom.version")  
    ?: getLatestBomVersion(versionTagPrefix = "SDK_BOM_")
```

01 SKELETAL

```
<groupId>cz.kotox.crypto.sdk</groupId>  
<artifactId>bom</artifactId>  
<version>2025.11.02</version>  
<packaging>pom</packaging>  
<dependencyManagement>  
    <dependencies>  
        <dependency>  
            <groupId>cz.kotox.crypto.sdk</groupId>  
            <artifactId>crypto-tracker</artifactId>  
            <version>1.1.1</version>  
        </dependency>  
        <dependency>  
            <groupId>cz.kotox.crypto.sdk</groupId>  
            <artifactId>crypto-coindata</artifactId>  
            <version>2.1.1</version>  
        </dependency>  
        <dependency>  
            <groupId>cz.kotox.crypto.sdk</groupId>  
            <artifactId>crypto-news</artifactId>  
            <version>3.1.0</version>  
        </dependency>  
    </dependencies>  
</dependencyManagement>
```

VERSIONING



01 SKELETAL

THE PAIN POINTS

- Establishing clear traceability
- Need for guaranteed integrity
- Eliminating manual errors

OUR SOLUTION

- Zero-Touch Builds

The screenshot shows a code editor interface with a dark theme. At the top, there's a header bar with a green dot icon, the text "CR CRYPTO-SDK", and a dropdown menu "main". Below the header is a "Project" view showing a tree structure of three projects: "crypto-common", "crypto-news", and "crypto-tracker". Each project has a "build" folder highlighted in yellow. To the right of the project tree is a code editor window displaying a "version.properties" file. The file contains the following content:

```
#Version auto-bumped by
#Mon Nov 03 13:36:54 CET
sdk.content.hash=473a514
sdk.version=0.0.4
```

PUBLICATION



01 SKELETAL

THE FLOW

- **BOM** as the unifying container
- **Git Tags** as the release trigger
- **Maven Central** as the destination

PRO TIPS

- Tag-based triggers
- Trust the Pipeline



SDK_BOM_2025.11.02

8 packages

cz.kotox.crypto.sdk.bom

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.bom](#)

cz.kotox.crypto.sdk.crypto-common

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.crypto-common](#)

cz.kotox.crypto.sdk.internal.common

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.internal.common](#)

cz.kotox.crypto.sdk.internal.logger

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.internal.logger](#)

cz.kotox.crypto.sdk.internal.network

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.internal.network](#)

cz.kotox.crypto.sdk.crypto-news

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.crypto-news](#)

cz.kotox.crypto.sdk.crypto-coindata

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.crypto-coindata](#)

cz.kotox.crypto.sdk.crypto-tracker

Published 20 hours ago by Michal Jenicek in [cz.kotox.crypto.sdk.crypto-tracker](#)

CI/CD PIPELINES



01 SKELETAL

THE AUTOMATED FLOW

1. **Verify:** Lint, Unit & Integration Test
2. **Calculate:** Derive version from content
3. **Document:** Auto-generate API docs
4. **Publish:** Upload artifacts

PRODUCTION SAFEGUARDS

- **Local Parity:** The pipeline runs locally exactly as it does on CI. Logic sits in build scripts, not YAML. **Verify & Calculate should be part of local routine.**

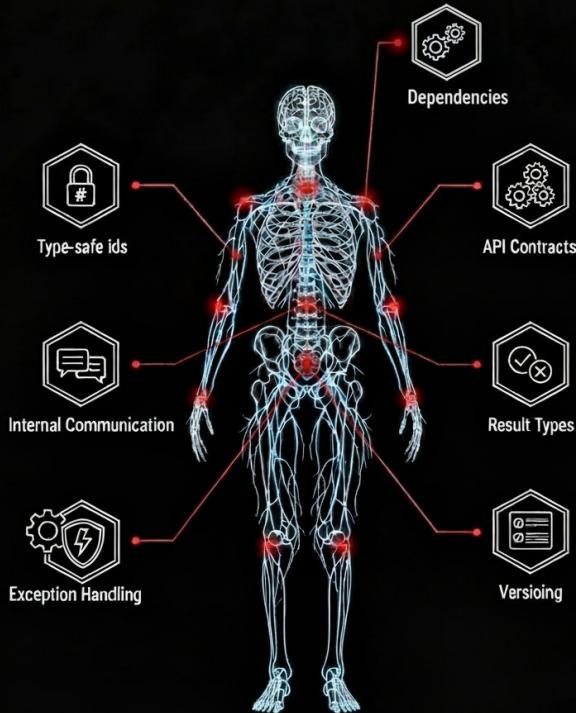
THE NERVOUS SYSTEM

02

02 • THE NERVOUS SYSTEM

*Inconsistency breaks the **Developer Experience**.*

- API Contracts
 - Type-safe IDs
 - Error Modeling
- Dependency Hygiene



TYPE SAFE IDS



02 NERVOUS

THE PITFALL: PRIMITIVE OBSESSION

- The Swap Risk
- Runtime Crashes

THE FIX: INLINE CLASSES

- (*Almost*)Zero Overhead
- Self-Documenting

```
/**
 * Represents unique id of the Story across Crypto sdks.
 */
public sealed interface StoryId { 2 implementations
    /**
     * Human readable unique id of the story usable also as the slug on the web.
     * @property value - human readable unique id of the story.
     */
    @JvmInline
    public value class StorySlug(public val value: String) : StoryId
    /**
     * Unique id of the story in the CMS system.
     * @property value - id of the story in the CMS system.
     */
    @JvmInline
    public value class StoryCmsId(public val value: String) : StoryId
}
```

`@JvmInline`

`public value class NewsId(public val value: String)`

ERROR MODELING



02 NERVOUS

PHILOSOPHY

- Errors are Data

IMPLEMENTATION

- Typed Error Hierarchy

```
public sealed class ApiError( 20 Usages 4 Inheritors
    message: String,
    cause: Throwable? = null,
) : SdkError(message, cause) {

    public data class ConnectivityError(...) :
        ApiError(message, cause)

    public data class CancellationError(...) :
        ApiError(message, cause)

    public data class EmptyResponseError(...) :
        ApiError(message, cause)

    public data class ResponseError(...) :
        ApiError(message, cause)
}

public fun Exception.transformApiError(errorResponseMessage: String? = null):
    ApiError {...}
```

ERROR MODELING



02 NERVOUS

PHILOSOPHY

- Errors are Data
- The "No-Crash" Guarantee

```
public sealed class Either<out E, out V> { 2 Inheritors
    public data class Value<out V>(val value: V) : Either<Nothing, V>()
    public data class Error<out E>(val error: E) : Either<E, Nothing>()
```

```
public suspend fun getCoinDetail(coinMarketId: CoinMarketId): Either<SdkError, CoinDetail>
```

IMPLEMENTATION

- Typed Error Hierarchy
- Explicit Contracts

```
fun testGetCoinDetail() = runTest {
    val testTag = "[testGetCoinMarkets]"
    coinData.getCoinDetail(coinMarketId = "usd").fold(
        error = {
            logE(t = null) { "$testTag ERROR: $it" }
            fail("getCoinMarkets failed to return value")
        },
        value = {
            logD { "$testTag VALUE: $it" }
        }
    )
}
```

DEPENDENCIES



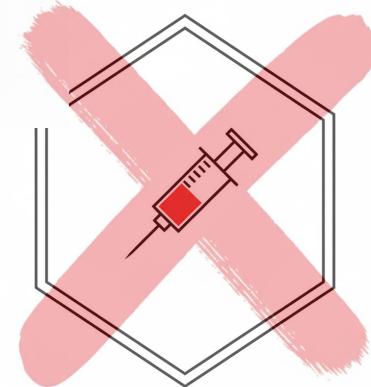
THE ZERO-BLOAT POLICY

- Transitive Hygiene
- KMP Readiness

ARCHITECTURAL CHOICES

- Constructor Injection over DI Framework
- Network Stack: Ktor(fit) over Retrofit

02 NERVOUS



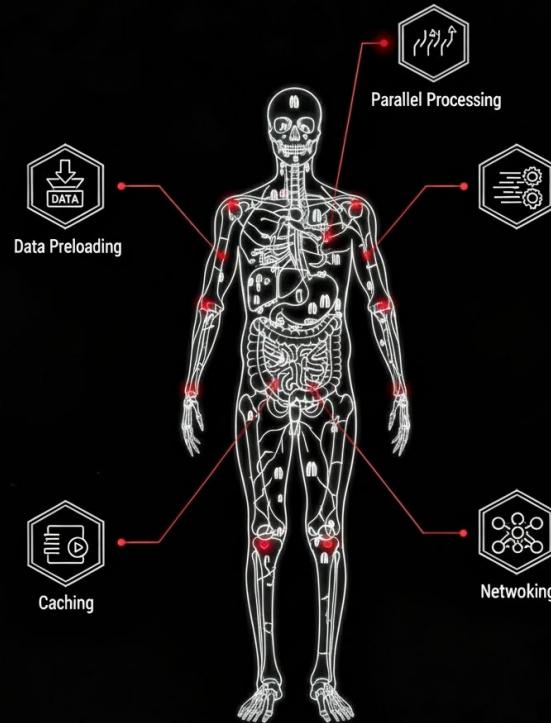
THE METABOLIC SYSTEM

03

03 • THE METABOLIC SYSTEM

An inefficient metabolic system will be slow, drain the battery, and waste resources.

- Network Hygiene
- Data Consistency
- Performance Optimization



NETWORK HYGIENE



03 METABOLIC

THE PHILOSOPHY: POLITE GUEST

- **Zero-Trust Security:** The binary never contains secrets.
- **Traffic Efficiency:** We minimize radio wake-ups. One user is fine; 10 million users is a DDoS attack.

THE TACTICS: IMPLEMENTATION

- **Credential Injection:** API Keys are injected by the host app at runtime.
- **Exponential Backoff:** Retries must be spaced out to avoid self-DDoSing the backend during outages

DATA CONSISTENCY



03 METABOLIC

THE PATTERN: REPOSITORIES

- **Business Logic (*UseCase*):**
Orchestrates the flow, remaining blind to the data source.
- **Source Resolution (*Repository*):**
Arbitrates transparently between Disk, Memory, and Network.

THE TOOLING: DON'T REINVENT

- The Complexity Trap:
Implementing **deduplication**, **memory caching**, and **disk persistence** from scratch is error-prone.
- Recommendation: Leverage the **STORE** library to handle these race conditions out of the box.

<https://store.mobilenativefoundation.org/docs/meet-store>

STORE



MobileNativeFoundation / Store

Watch 65 Fork 208 Starred 3.3k

Code Issues 42 Pull requests 14 Discussions Actions

5.1.0-alpha06 · 5.1.0-alpha05 · 5.1.0-alpha04 · 5.1.0-alpha02 · 5.1.0-alpha01 · 5.0.0

on Feb 27 · d7af217 · 42 · 14 · 1 · 3.3k

on Oct 18, 2024 · f9072 · 42 · 14 · 1 · 3.3k

on Jul 7, 2024 · 83e5f8i · 42 · 14 · 1 · 3.3k

on Jan 28, 2024 · 40e66 · 42 · 14 · 1 · 3.3k

on Jan 16, 2024 · 5ef94c2 · zip · tar.gz · Notes · 3.3k

on Sep 14, 2023 · 219a251 · zip · tar.gz · 3.3k

Store Public

main · 107 Branches · 38 Tags

matt-ramotar · Release 5.1.0-alpha06 (#701)

723 Commits · 8 months ago · 2 years ago · 5 months ago · last year

About

A Kotlin Multiplatform solution for working with data. Whether you're building alone or with a team of thousands, Store can help

store.mobilenativefoundation.org

repository · offline-first · cache · kotlin-multiplatform

<https://github.com/MobileNativeFoundation/Store>



STRV

PERFORMANCE OPTIMIZATION



03 METABOLIC

CONCURRENCY

- **Non-Blocking by Default:** Heavy lifting is offloaded to background dispatchers. The Main Thread is sacred.
- **Structured Concurrency:** operations inherit the Host's Scope. If the ViewModel clears, the SDK call cancels immediately.

PERCEIVED SPEED

- **Warm Start:** Critical config is pre-fetched during init. The first user interaction is zero-latency.

THE SENSORY SYSTEM

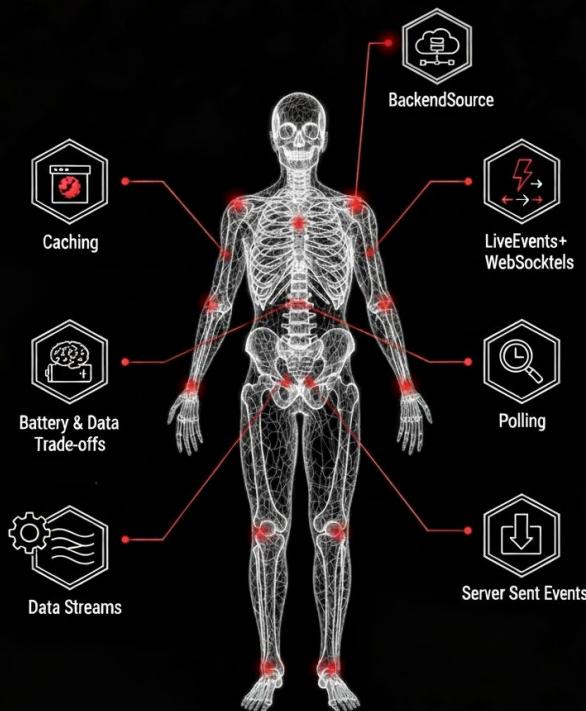
04

04 • THE SENSORY SYSTEM

*Perceiving the world in real-time. Push vs.
Pull.*

- Real-Time Perception (SSE/WSS)
- Efficient Updates

The Sensory System



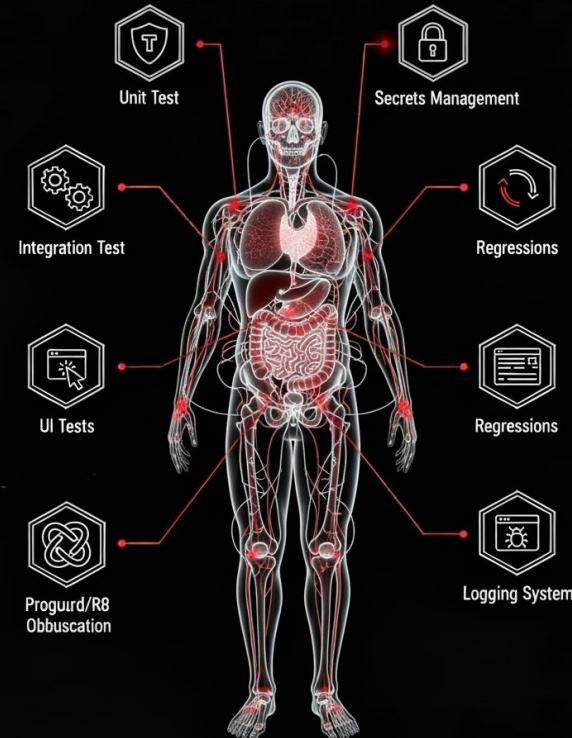
THE IMMUNE SYSTEM

05

05 • THE IMMUNE SYSTEM

Defending against bugs and regressions. A strong immune system is the foundation of Trust.

- Quality Assurance
- Binary Hardening
- Signal Observability



QUALITY ASSURANCE



THE STRATEGY: EARLY DETECTION

- Regression Prevention
- Pressure Testing
- Edge Case Coverage

THE TACTIC: INTEGRATION FROM DAY 1

- Unit Test
- Integration Test

05 IMMUNE

```
@Before
fun setup() {
    context = ApplicationProvider.getApplicationContext()
    coinData = CoinDataBuilder().setLoggerCallback(
        sdkLoggerCallback = object : SDKLoggerCallback {
            override fun onLogMessage(
                tag: String,
                priority: LogPriority,
                t: Throwable?,
                message: String,
            ) {
                println(message = message)
            }
        },
    ).build()
}

@Test
fun testGetCoinDetail() = runTest {
    val testTag = "[testGetCoinMarkets]"
    coinData.getCoinDetail(coinMarketId = "usd").fold(
        error = {
            logE(t = null) { "$testTag ERROR: $it" }
            fail("getCoinMarkets failed to return value")
        },
        value = {
            logD { "$testTag VALUE: $it" }
        }
    )
}
```

BINARY HARDENING



05 IMMUNE

THE STRATEGY: INTERNAL HIDING

≡ consumer-proguard-rules.pro ×

```
1  ## Keeps all public classes (and their public/protected members) within the
2  ## 'cz.kotox.crypto.sdk.tracker' package and all its sub-packages,
3  ## EXCEPT for any classes found inside the 'cz.kotox.crypto.sdk.tracker.internal'
4  ## sub-package.
5  ##
6  ## This effectively protects the SDK's public API surface while allowing
7  ## the 'internal' implementation packages to be fully obfuscated and shrunk
8  ## by the consuming app.
9  -keep public class cz.kotox.crypto.sdk.tracker.**, !cz.kotox.crypto.sdk.tracker.internal.**
10    public protected *;
11 }
```

BINARY HARDENING



05 IMMUNE

THE CHALLENGE: DEBUGGABILITY

- **Stack Trace Management:** Obfuscated crashes yield unreadable stack traces. This can be solved by uploading mapping files. Anyway using shared modules makes it hard to use. It's a challenge.
- **Reproducibility:** New builds generate different obfuscation signatures. This means a single production bug might appear as multiple unique crashes in monitoring tools.

SIGNAL OBSERVABILITY



05 IMMUNE

THE STRATEGY: DELEGATION

- **Inversion of Control:** The SDK defines a Logger interface, but the host app provides the implementation

```
@Before
fun setup() {
    context = ApplicationProvider.getApplicationContext()
    coinData = CoinDataBuilder().setLoggerCallback(
        sdkLoggerCallback = object : SDKLoggerCallback {
            override fun onLogMessage(
                tag: String,
                priority: LogPriority,
                t: Throwable?,
                message: String,
            ) {
                println(message = message)
            }
        },
    ).build()
}
sdkLoggerCallback = object : SDKLoggerCallback {
    override fun onLogMessage(
        tag: String,
        priority: LogPriority,
        t: Throwable?,
        message: String,
    ) {
        Timber.tag(tag)
        Timber.log(priority = priority.priorityInt, t = t, message = message)
    }
},
```

SIGNAL OBSERVABILITY



05 IMMUNE

THE CHALLENGE: FALSE POSITIVES

- **Alert Fatigue:** Monitoring tools often treat any Exception as an incident.
- **Context Filtering:** We parse exception messages for known “noise” and downgrade their level of importance.

```
public fun SdkError.toLogPriority(): LogPriority = 10 Usages
when (this) {
    is ApiError.ConnectivityError -> this.logPriority ?: LogPriority.INFO
    is ApiError.CancellationError -> this.logPriority ?: LogPriority.VERBOSE
    is ApiError.EmptyResponseError,
    -> LogPriority.INFO

    is ApiError.UnsupportedSearchParameterError -> LogPriority.WARN
    else -> LogPriority.ERROR
}

public fun Throwable?.toLogPriority(): LogPriority = 10 Usages
when (this) {
    is kotlinx.coroutines.CancellationException -> LogPriority.VERBOSE
    else -> {
        if (this?.message?.containsAnyOf(
            strings = connectivityInfoDictionary,
            ignoreCase = true,
        ) == true
    ) {
        LogPriority.INFO
    } else {
        LogPriority.ERROR
    }
}
}

public val connectivityInfoDictionary: List<String> =
listOf(
    "Unable to resolve host",
    "Software caused connection abort",
    "timeout",
    "Read timed out",
    "Broken pipe",
    "failed to connect",
    "Required SETTINGS preface not received",
    "Expected a SETTINGS frame but was",
    "unexpected end of stream",
    "No network connection is available",
    "stream was reset",
    "Connection reset",
    "connection closed",
    "Connection timed out",
    "SSL handshake timed out",
    "SSL handshake aborted",
    "exhausted all routes",
    "Connection closed by peer",
    "Read error",
    "ConnectionShutdownException",
)
```

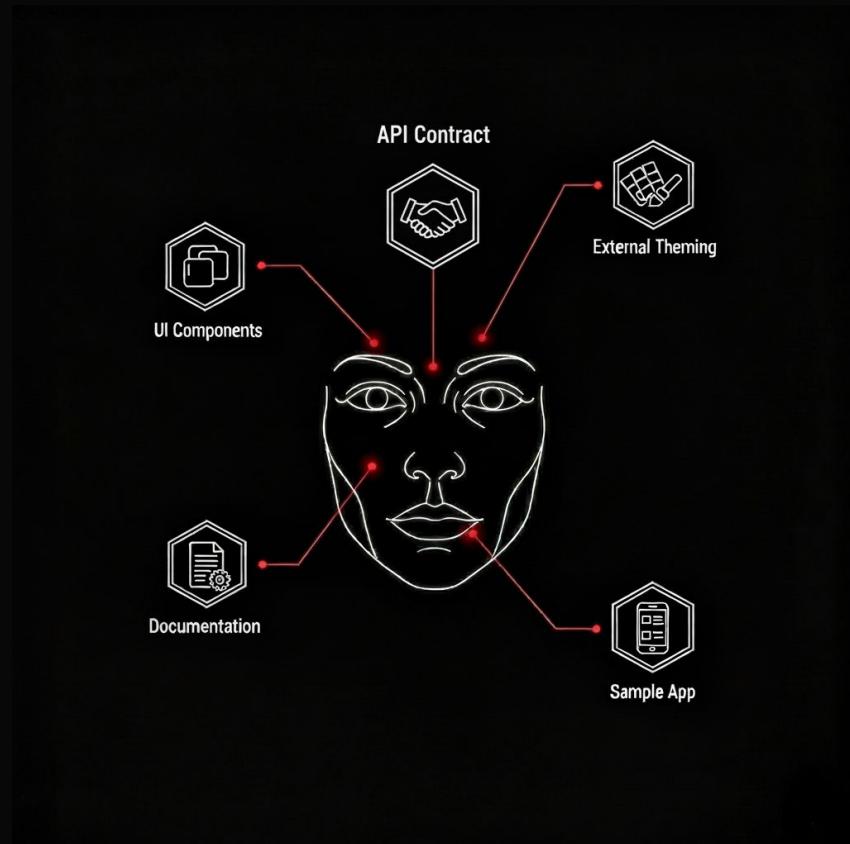
THE PUBLIC FACE

06

06 • THE PUBLIC FACE

The part of the ecosystem that developers directly see and interact with.

- UI Component Library
- Theming Strategy
- The Developer Portal
- Reference Implementation



UI COMPONENTS



06 FACE

THE STRATEGY: THE STARTER KIT

- Rapid Prototyping
- Ejectable by Design

THE IMPLEMENTATION: COMPOSABLE SLOT API

- Atomic Design

```
@Composable
public fun StoryContentColumn(
    storyContent: StoryContent,
    modifier: Modifier = Modifier,
    contentPadding: PaddingValues = PaddingValues( all = 0.dp ),
    onCloseHeaderClick: (() -> Unit)? = null,
    photoPlaceholderShimmeringColor: Color? = null,
    photoPlaceholderShimmeringBackgroundColor: Color? = null,
    padding: Dp = 16.dp,
    headerBottomContent: @Composable () -> Unit = {},
    sdkLoggerCallback: SDKLoggerCallback = SDKLoggerCallbackNoOp(),
) {
    SDKLogger.install( tag = MODULE_IDENTIFIER, sdkLoggerCallback )
}
```

```
> storycard
> thumbnail
> StoryContentColumn.kt
> StoryContentColumnContent.kt
> StoryContentColumnNavHost.kt
< theme.sample
> Colors.kt
> Shape.kt
```

EXTERNAL THEMING



06 FACE

THE STRATEGY:

MATERIAL 3 COMPLIANCE

- Native Citizens
- Zero-Config Start

THE IMPLEMENTATION:

- Transparent Styling.
Material Theme by
default.

```
setContent {  
    SDKSampleAppTheme {  
        MainActivityContent(  
            shakeDetector = this,  
            restartActivityWithDataCleanUp = {  
                finish()  
                startActivity(intent)  
                viewModel.dataCleanUp()  
            },  
        )  
    }  
}
```

THE API REFERENCE



06 FACE

THE STRATEGY: ZERO DRIFT

- Single Source of Truth
- Automated Generation

THE STACK: DOKKA

- Kotlin-First: We use Dokka to correctly render Kotlin features
- Deep Linking: Every function has permalink.

The screenshot shows a dark-themed API documentation interface for the 'CRYPTO SDK 2025.11.02'. On the left is a navigation sidebar with a tree view of package structures:

- ↳ Crypto-coindata
 - ↳ cz.kotox.crypto.sdk.coindata
 - ↳ cz.kotox.crypto.sdk.coindata.domain
 - ↳ cz.kotox.crypto.sdk.coindata.domain.model
 - ↳ cz.kotox.crypto.sdk.coindata.internal
 - ↳ cz.kotox.crypto.sdk.coindata.internal.data.api
 - ↳ cz.kotox.crypto.sdk.coindata.internal.data.dto
 - ↳ cz.kotox.crypto.sdk.coindata.internal.data.mapper
 - ↳ cz.kotox.crypto.sdk.coindata.internal.test
 - ↳ cz.kotox.crypto.sdk.coindata.internal.usecase
 - ↳ cz.kotox.crypto.sdk.coindata.internal.utils
- ↳ Crypto-common
- ↳ Crypto-news
- ↳ Crypto-tracker
 - ↳ cz.kotox.crypto.sdk.tracker
 - ↳ cz.kotox.crypto.sdk.tracker.domain
 - ↳ cz.kotox.crypto.sdk.tracker.domain.model
 - ↳ cz.kotox.crypto.sdk.tracker.internal
 - ↳ cz.kotox.crypto.sdk.tracker.internal.data.api
 - ↳ cz.kotox.crypto.sdk.tracker.internal.dto

On the right side, there are several sections with placeholder text ('TBD.1', 'TBD.2', 'TBD.3') and a 'All modules:' section with links to 'Crypto-coindata' and 'Crypto-common'.

CRYPTO SDK Components

Coindata
TBD.1

News
TBD.2

Tracker
TBD.3

All modules:

Crypto-coindata

Crypto-common

THE DEVELOPER PORTAL



06 FACE

THE STRATEGY

- Repository Separation:** We host the Portal in a dedicated repository so we don't pollute git history with doc files.
- The “5-Minute” Rule:** The portal is designed to get a developer from Zero to Hello World in under 5 minutes.

The screenshot shows the homepage of the CRYPTO developer portal. At the top, there's a logo with a stylized 'C' and the word 'CRYPTO'. Below it is a sub-headline: "Power your mobile apps with the CRYPTO SDKs". A prominent "Get started" button is located in the center. To the right, there are several mobile phone screens displaying different features of the Tracker SDK, such as a news feed, alerts, and settings. On the left, there's a section titled "Tracker SDK" with some placeholder text and a bulleted list of developer benefits. At the bottom, there's a call-to-action: "For more information, visit our Tracker SDK documentation."

THE DEVELOPER PORTAL



06 FACE

THE STACK: DOCUSAURUS

- **Markdown-Based:** Writers/Devs commit docs as Markdown. API Reference is attached as an external pages
- **Unified Navigation:** The API Reference is anchored in the main sidebar, serving as a direct gateway to the generated external documentation.

The screenshot shows a dark-themed developer portal for the "CRYPTO SDK". The left sidebar contains a navigation menu with sections like "Get started", "Introduction", "Android", "Tracker SDK", "News SDK", and "Coindata SDK". Under "Coindata SDK", "Introduction" is selected. The main content area has a breadcrumb navigation: Home > Get started > Coindata SDK > Introduction. The main title is "Introduction". Below it is a block of Latin placeholder text: "Lirum Coindata SDK offert nativus mobilis componenta ad visibiliter reprezentare data ex . Haec componenta sunt stilo apta ita ut eorum aspectus et sensus aptari possunt ad quolibet manipulus/fustis visuale thema. Omnia componenta sunt composita simul ita pro exemplo fabula visum componentum dynamice componet suum proprium contentum ex aliis visum componentibus. Compositio agitur per data venientia ex ." To the right of the main content are two lists: "API Feature list" and "SDK Feature list", each containing a bulleted list of items in Latin.

- FabulaColumna visum componentum
- Markdown visum componentum
- Photo visum componentum
- Video visum componentum
- Promo visum componentum
- Insere socialis visum componentum

- Ordo caput visum componentum
- Tabula visum componentum

REFERENCE IMPLEMENTATION



06 FACE

THE ROLE: CANONICAL GUIDE

- **Integration Blueprint:**
demonstrates integrating every feature of the SDK.
- **Architecture Agnostic**

THE IMPLEMENTATION

- **Low Noise: Minimal**
architecture designed to focus strictly on SDK integration.

The screenshot displays the Android Studio interface. On the left, the project structure shows a nested folder 'app-mobile-sdk-crypto' within 'android-sdk-crypto-ecosystem'. The 'gradle' and 'src' folders are also visible. In the center, the code editor shows the 'AppModule.kt' file with the following content:

```
1 package cz.kotox.sdk.crypto.app.di
2
3 import ...
4
5 @Module(4 Usages)
6 @ComponentScan(...value = "cz.kotox.sdk.crypto.app")
7 class AppModule {
8
9     @Single(1 Usage)
10    fun provideCoinData(): CoinData = CoinData.Builder()
11        .setLoggerCallback(
12            sdkLoggerCallback = object : SDKLoggerCallback {
13                override fun onLogMessage(
14                    tag: String,
15                    priority: LogPriority,
16                    t: Throwable?,
17                    message: String,
18                ) {
19                    Timber.tag(tag)
20                    Timber.log(priority = priority.priority)
21                }
22            },
23        )
24        .build()
25
26
27
28
29
30
31
32 }
```

On the right, a mobile application screen titled 'Pixel 6' shows a list of cryptocurrencies with their logos, names, symbols, and current prices. The list includes Dogecoin (DOGE), Cardano (ADA), Figure Heloc (FIGR_HELOC), Wrapped stETH (WSTETH), Wrapped Bitcoin (WBTC), WhiteBIT Coin (WBT), Wrapped Beacon ETH (WBETH), Zcash (ZEC), and Hyperliquid (HLQ). Below the application is a coin market price feed with similar data.

Explore the **Ecosystem**.
The complete implementation is
Open Source and evolving on GitHub



THE CODE



[android-sdk-crypto-ecosystem](#)

THE KNOWLEDGE



[android-sdk-crypto-ecosystem-doc](#)