Reuso de código com Kotlin Multiplataforma

Felipe Costa

Engenheiro de software Senior @ OLX

Profissional de software apaixonado. Atuo como desenvolvedor, principalmente, para dispositivos móveis desde 2011 e fazendo-o principalmente em **Kotlin** desde 2016.

Nosso Propósito

Aproximamos Brasileiros para transformar itens em felicidade.

Kotlin @ OLX

Kotlin @ OLX


Android
Autenticação
Experimentação
inserção
Busca
Recomendação

Karol
&
MVP de um chat multiplataforma


Por quê reutilizar código entre plataformas?

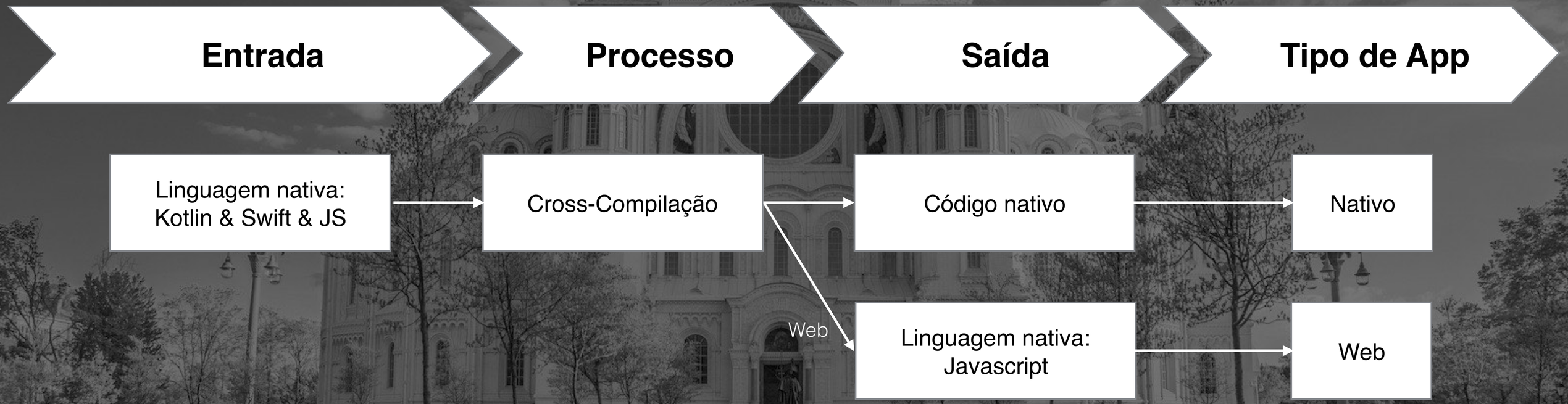

DRY

Mitigação de risco

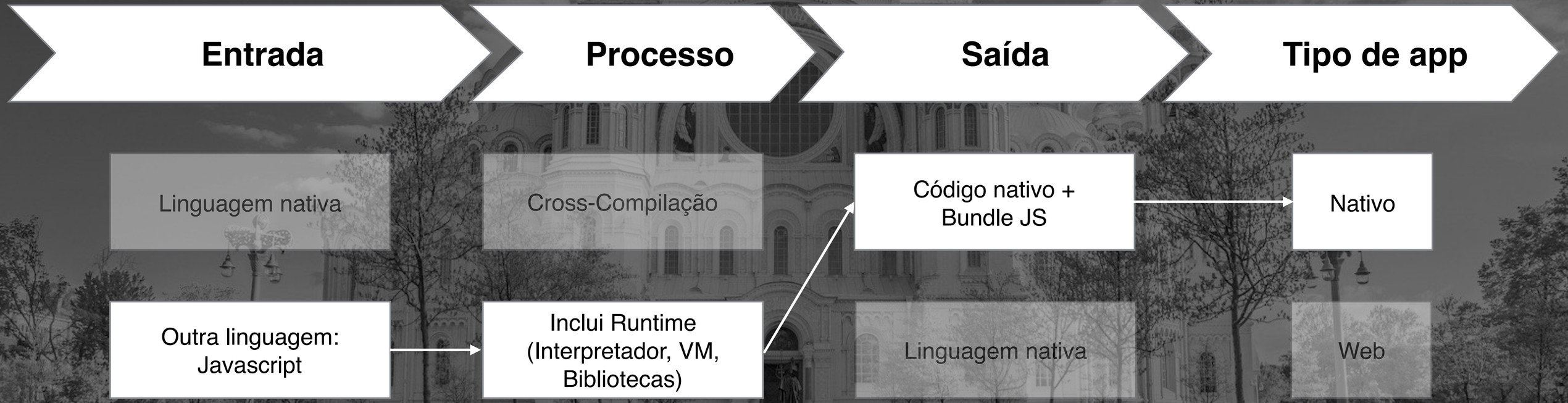
Paridade de funcionalidades


Soluções

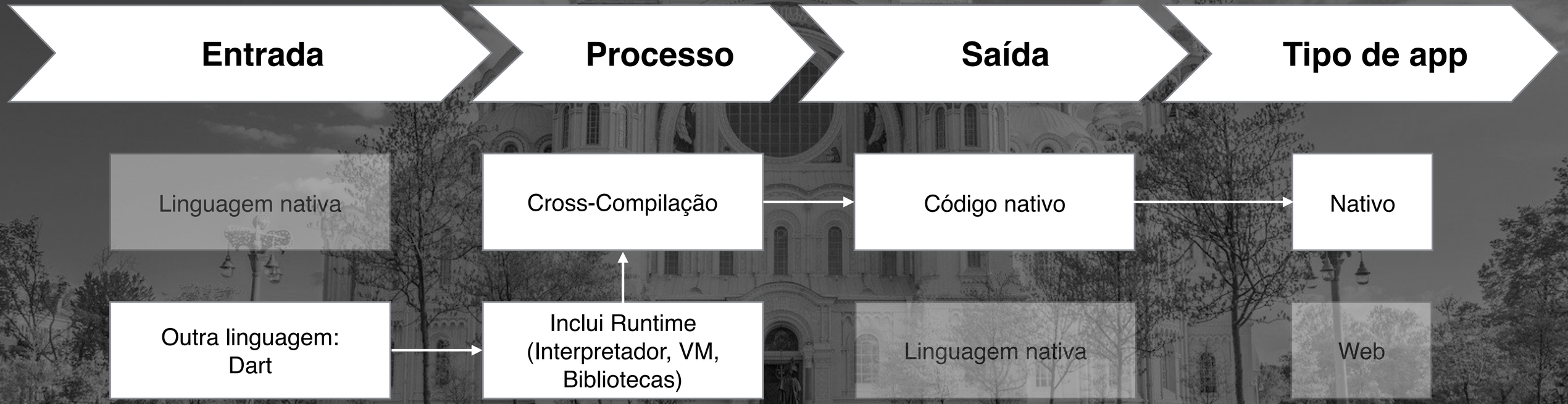
Nativo


Entrada
Processo
Saída
Tipo de App
Linguagem nativa:
Kotlin & Swift & JS
Cross-Compilação
Código nativo
Nativo
Web
Linguagem nativa:
Javascript
Web

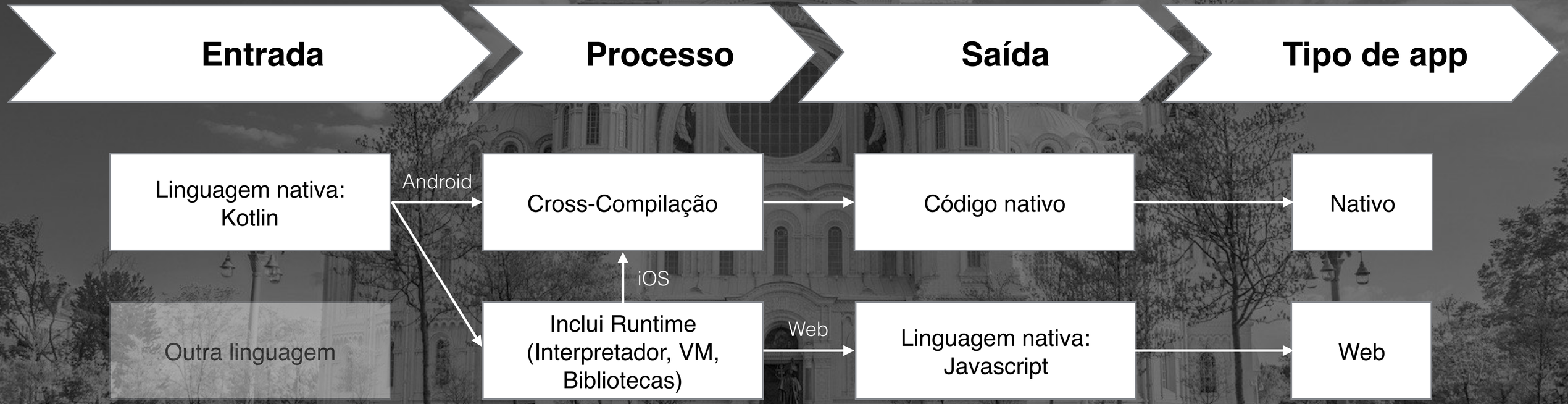
React Native



Flutter


Entrada
Processo
Saída
Tipo de app
Linguagem nativa
Cross-Compilação
Código nativo
Nativo
Outra linguagem: Dart
Inclui Runtime (Interpretador, VM, Bibliotecas)
Linguagem nativa
Web

Kotlin Multiplataforma



Por quê Kotlin Multiplataforma?

Compartilhamento opcional

Baixo risco
Sem **Grandes decisões**

100% Nativo

Interop suave

Compartilhamento de código

Não é “cross plataforma”

Comunidade ativa

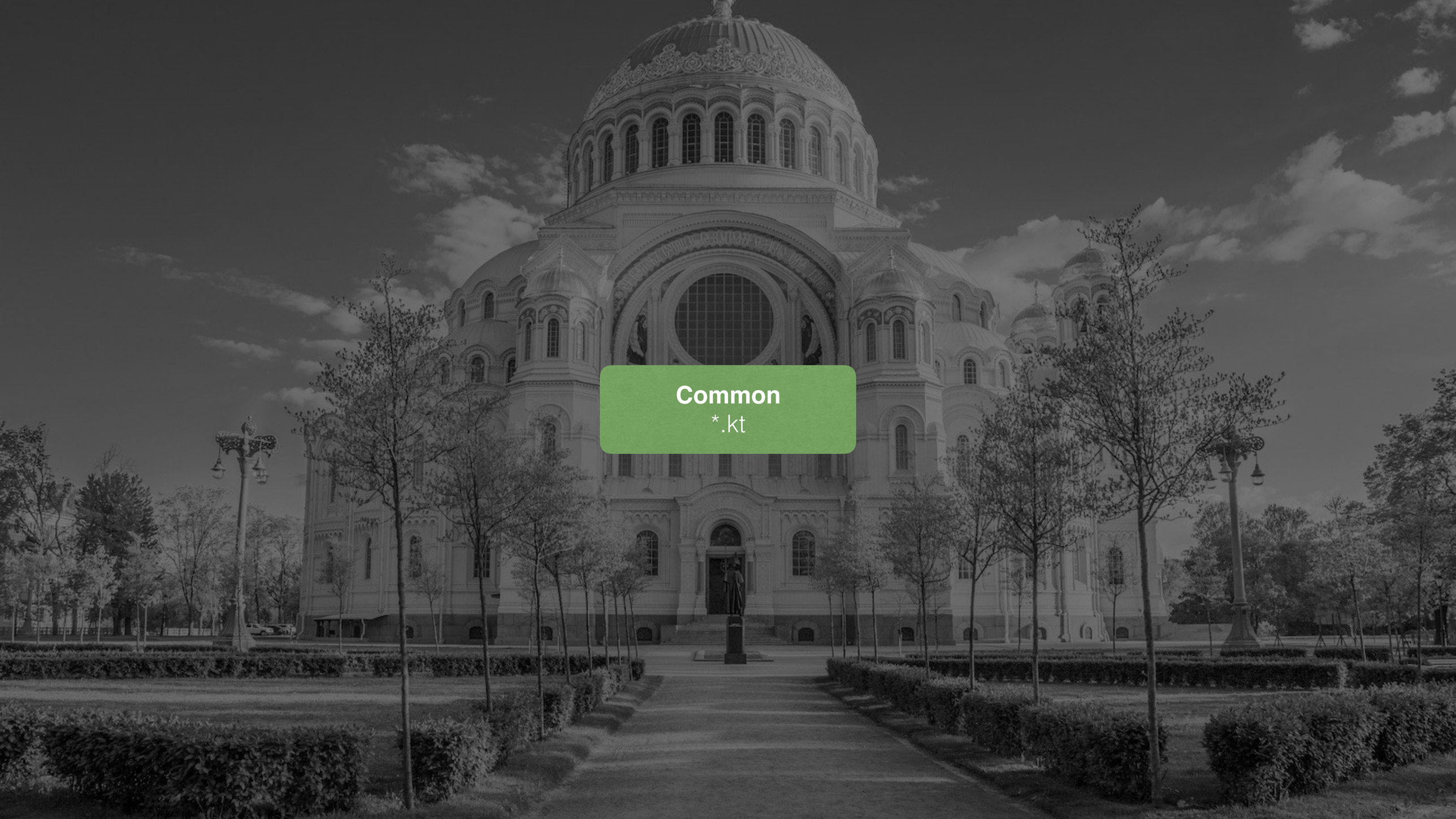

Boas ferramentas

Linguagem moderna

Sem UI

não necessariamente


require 'spec_helper'
require 'rspec/rails'
require 'capybara/rspec'
require 'capybara/rails'
Capybara.javascript_driver
Category.delete_all;
Shoulda::Matchers.configure
config.integrate
with.test_framework
end
Add additional requires
Requires supporting
spec/support/ and its
run as spec files by default.
in _spec.rb will
run twice. It is
end with _spec.rb.
Muitas plataformas


Common
*.kt

Build.gradle

```
plugins {
    id 'kotlin-multiplatform' version '1.3.31'
}

repositories {
    mavenCentral()
}

kotlin {
    sourceSets {
        commonMain {
            dependencies {
                implementation kotlin('stdlib-common')
            }
        }
    }
}
```

Build.gradle

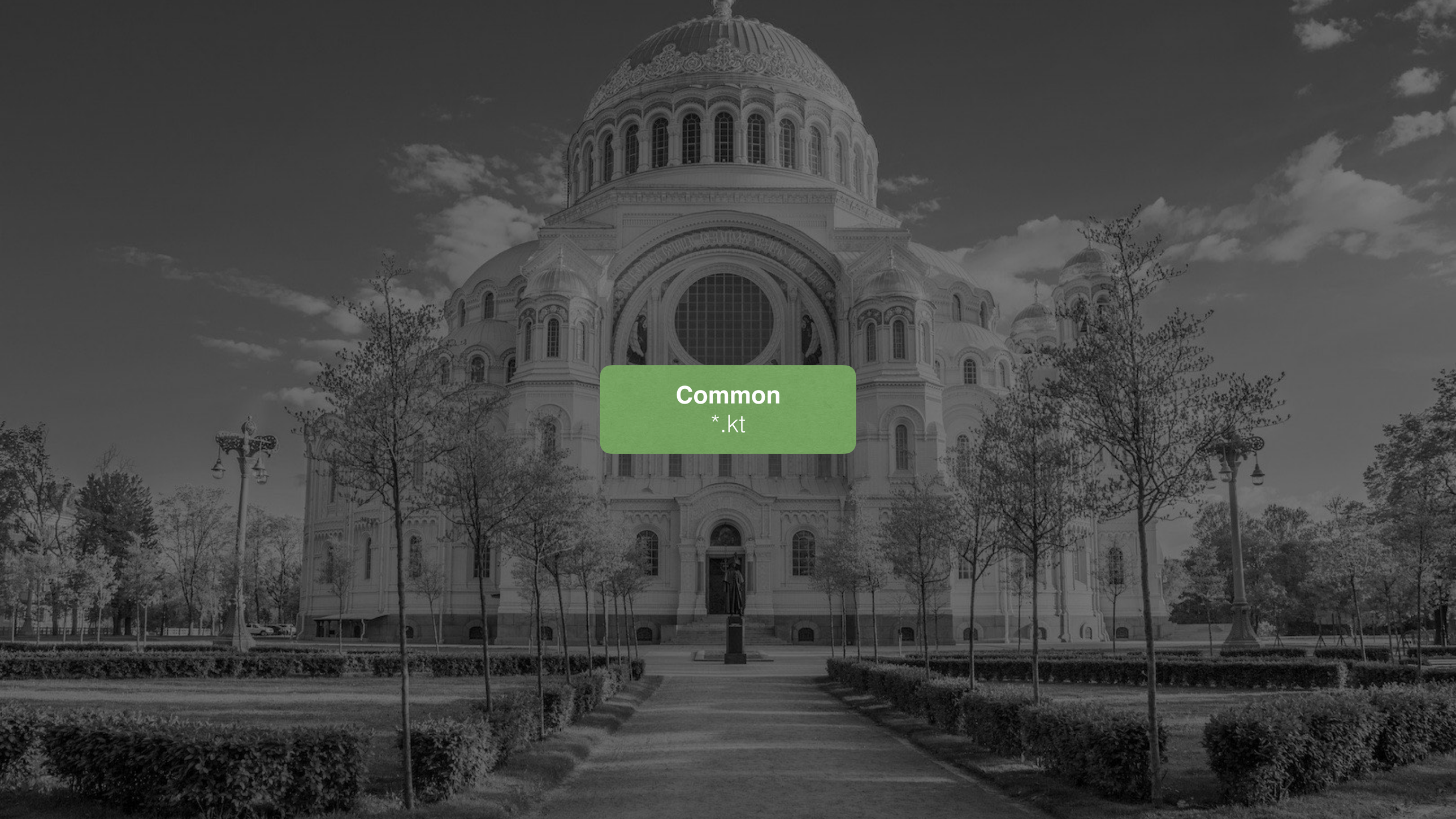
```
plugins {
    id 'kotlin-multiplatform' version '1.3.31'
}

repositories {
    mavenCentral()
}

kotlin {
    sourceSets {
        commonMain {
            dependencies {
                implementation kotlin('stdlib-common')
            }
        }
    }
}
```

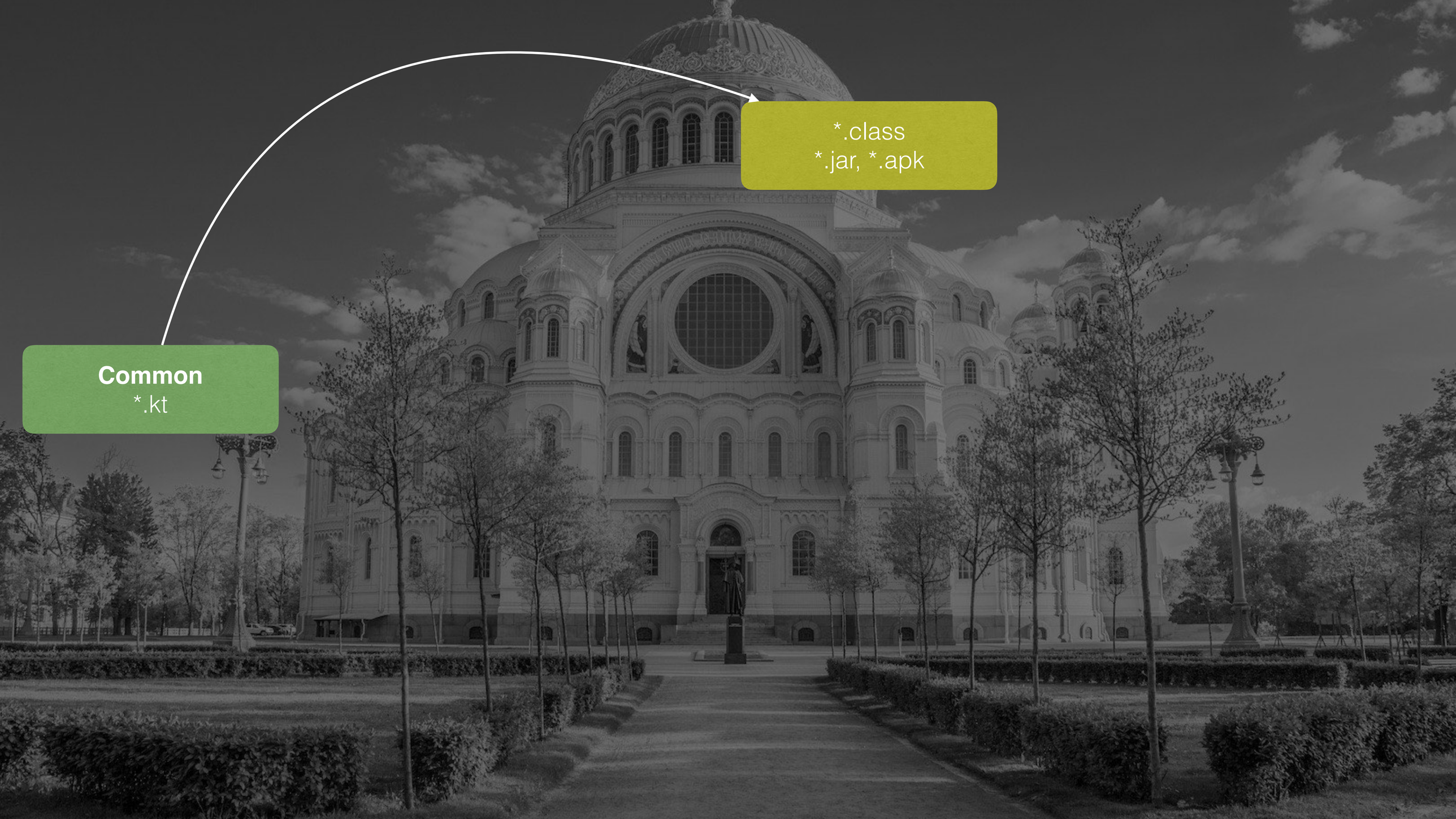
Build.gradle

```
plugins {
    id 'kotlin-multiplatform' version '1.3.31'
}

repositories {
    mavenCentral()
}

kotlin {
    sourceSets {
        commonMain {
            dependencies {
                implementation kotlin('stdlib-common')
            }
        }
    }
}
```

Build.gradle

```
plugins {
    id 'kotlin-multiplatform' version '1.3.31'
}

repositories {
    mavenCentral()
}

kotlin {
    sourceSets {
        commonMain {
            dependencies {
                implementation kotlin('stdlib-common')
            }
        }
    }
}
```


Common
*.kt


*.class
*.jar, *.apk
Common
*.kt

Build.gradle

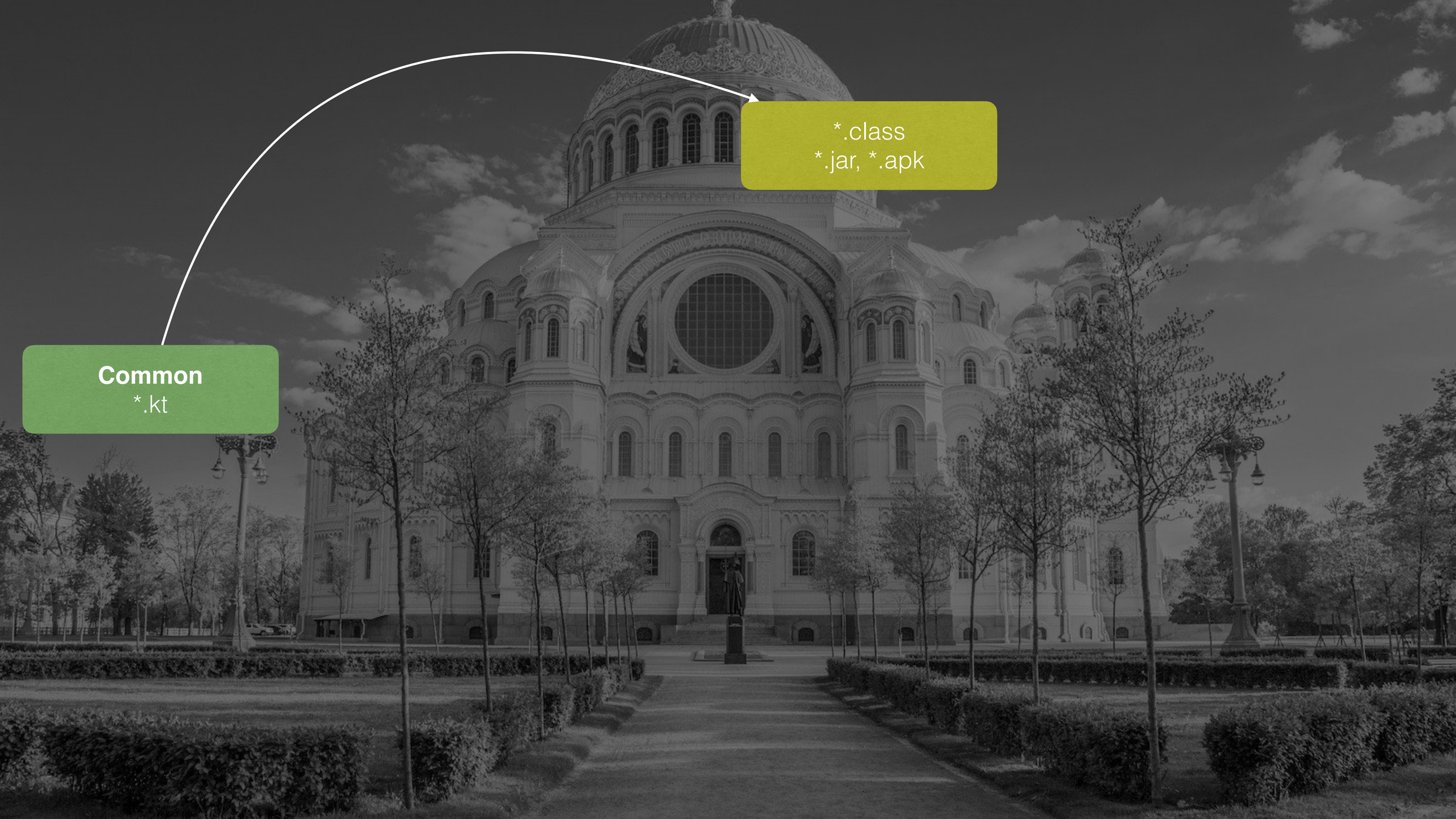
```
kotlin {
    jvm()
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    sourceSets {
        commonMain { /* */ }
    }
}
```


*.class
*.jar, *.apk
Common
*.kt


*.class
*.jar, *.apk
Common
*.kt
*.js

Build.gradle

```
kotlin {
    jvm()
    js()
    sourceSets {
        commonMain { /* */ }
    }
}
```

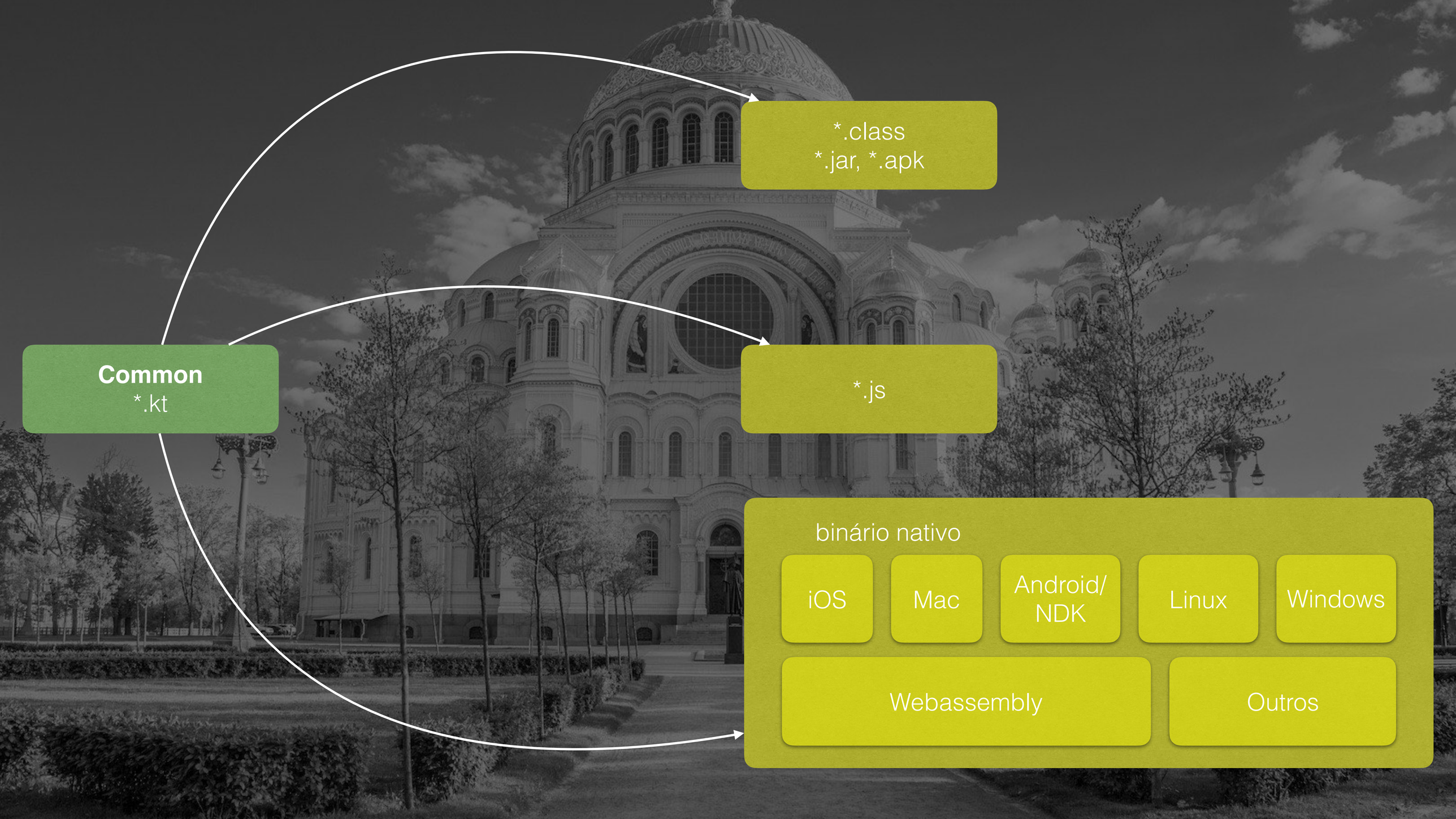
Build.gradle

```
kotlin {
    jvm()
    js()
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    sourceSets {
        commonMain { /* */ }
    }
}
```


*.class
*.jar, *.apk
Common
*.kt
*.js


*.class
*.jar, *.apk
Common
*.kt
*.js
binário nativo
iOS
Mac
Android/
NDK
Linux
Windows
Webassembly
Outros

Build.gradle

```
kotlin {
    jvm()
    js()
    // For ARM, should be changed to iosArm32 or iosArm64
    // For Linux, should be changed to e.g. linuxX64
    // For MacOS, should be changed to e.g. macosX64
    // For Windows, should be changed to e.g. mingwX64
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    // For ARM, should be changed to iosArm32 or iosArm64
    // For Linux, should be changed to e.g. linuxX64
    // For MacOS, should be changed to e.g. macosX64
    // For Windows, should be changed to e.g. mingwX64
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    // For ARM, should be changed to iosArm32 or iosArm64
    // For Linux, should be changed to e.g. linuxX64
    // For MacOS, should be changed to e.g. macosX64
    // For Windows, should be changed to e.g. mingwX64
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
    }
}
```

Common

```
fun hello(): String = "Hello from Kotlin
Multiplatform"
```

JVM

```
import kotlin.Metadata;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 15},
   bv = {1, 0, 3},
   k = 2,
   d1 =
{"\u0000\b\n\u0000\n\u0002\u0010\u000e\n\u0000\u001a\u0006\u0010\u0000\u001a\u00020\u0001¨\u0006\u0002"},
   d2 = {"hello", "", "Platform module com.example.hello-world-multiplatform.jvmMain including
[com.example.hello-world-multiplatform.commonMain]"}
)
public final class SampleKt {
   @NotNull
   public static final String hello() {
      return "Hello from Kotlin Multiplatform";
   }
}
```

JS

```
if (typeof kotlin === 'undefined') {
  throw new Error("Error loading module 'hello-world-multiplatform'. Its dependency 'kotlin' was not found.
Please, check whether 'kotlin' is loaded prior to 'hello-world-multiplatform'.");
}
this['hello-world-multiplatform'] = function (_, Kotlin) {
  'use strict';
  function hello() {
    return 'Hello from Kotlin Multiplatform';
  }
  var package$sample = _.sample || (_.sample = {});
  package$sample.hello = hello;
  Kotlin.defineModule('hello-world-multiplatform', _);
  return _;
}(typeof this['hello-world-multiplatform'] === 'undefined' ? {} : this['hello-world-multiplatform'], kotlin);
```

MacOS Binary

.kexe

Build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos") {
        binaries {
            sharedLib("sharedLib")
        }
    }
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos") {
        binaries {
            sharedLib("sharedLib")
        }
    }
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos") {
        binaries {
            sharedLib("sharedLib")
        }
    }
    sourceSets {
        commonMain { /* */ }
    }
}
```

MacOS Dynamic Library

```
#ifndef KONAN_LIBSHAREDLIB_H
#define KONAN_LIBSHAREDLIB_H
#ifdef __cplusplus
extern "C" {
#endif
/* typedefes */

typedef struct {
  /* Service functions. */
  /* ... */

  /* User functions. */
  struct {
    struct {
      struct {
        const char* (*hello)();
      } sample;
    } root;
  } kotlin;
} libsharedLib_ExportedSymbols;
extern libsharedLib_ExportedSymbols* libsharedLib_symbols(void);
#ifdef __cplusplus
}  /* extern "C" */
#endif
#endif  /* KONAN_LIBSHAREDLIB_H */
```

Build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos") {
        binaries {
            framework("framework")
        }
    }
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos") {
        binaries {
            framework("framework")
        }
    }
    sourceSets {
        commonMain { /* */ }
    }
}
```

Build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos") {
        binaries {
            framework("framework")
        }
    }
    sourceSets {
        commonMain { /* */ }
    }
}
```

MacOS Framework

```
#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

@interface KotlinBase : NSObject
- (instancetype)init __attribute__((unavailable));
+ (instancetype)new __attribute__((unavailable));
+ (void)initialize __attribute__((objc_requires_super));
@end;

@interface KotlinBase (KotlinBaseCopying) <NSCopying>
@end;

/* More types */

__attribute__((objc_subclassing_restricted))
__attribute__((swift_name("SampleKt")))
@interface FrameworkSampleKt : KotlinBase
+ (NSString *)hello __attribute__((swift_name("hello()")));
@end;

NS_ASSUME_NONNULL_END
```

Common

```
fun hello(): String = "Hello from Kotlin
Multiplatform"
```

Common

```
object Platform {
    val name: String
}

fun hello(): String = "Hello from ${Platform.name}"
```

JVM

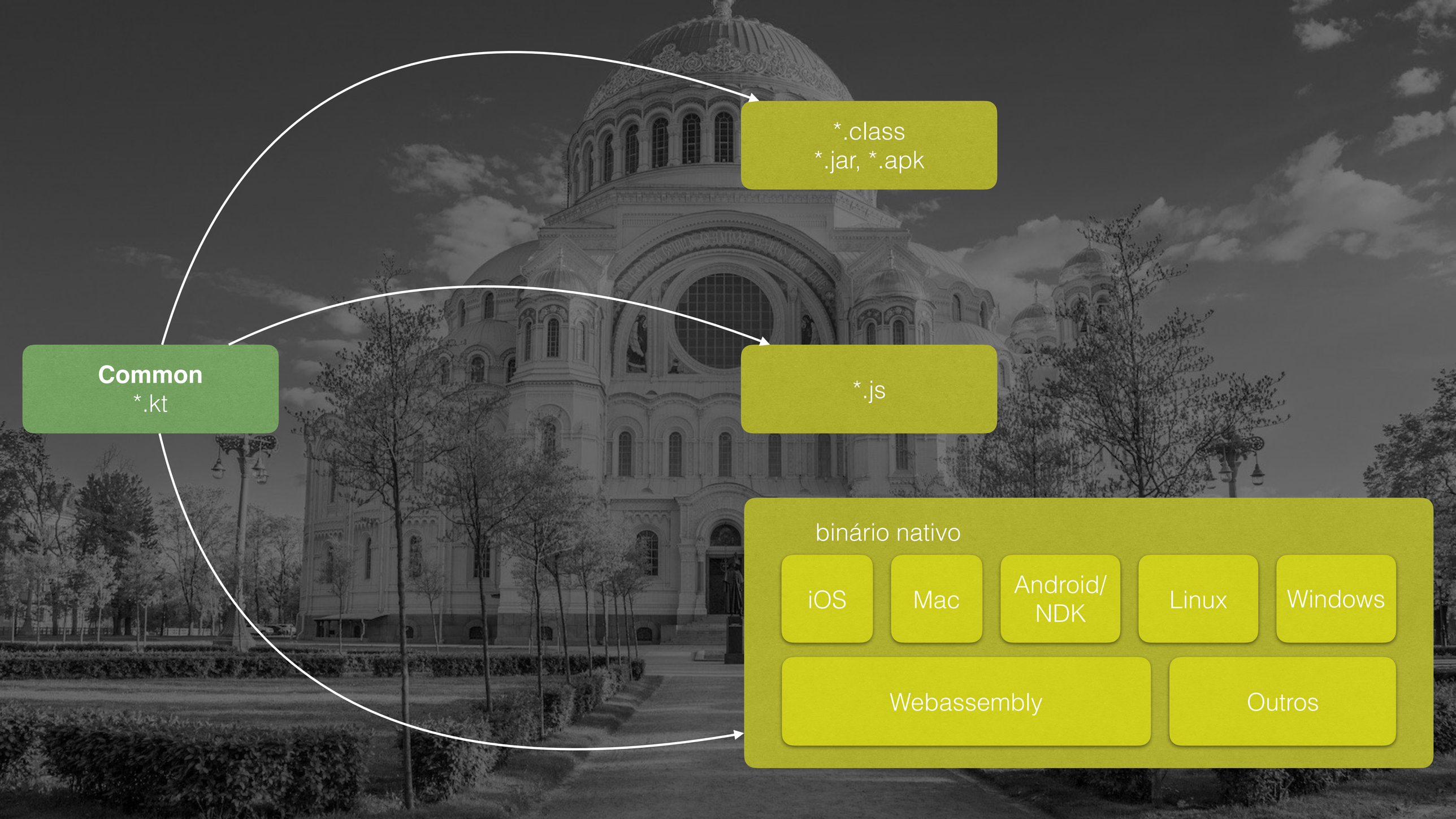
```
object Platform {
    val name: String = "JVM"
}

fun hello(): String = "Hello from ${Platform.name}"
```

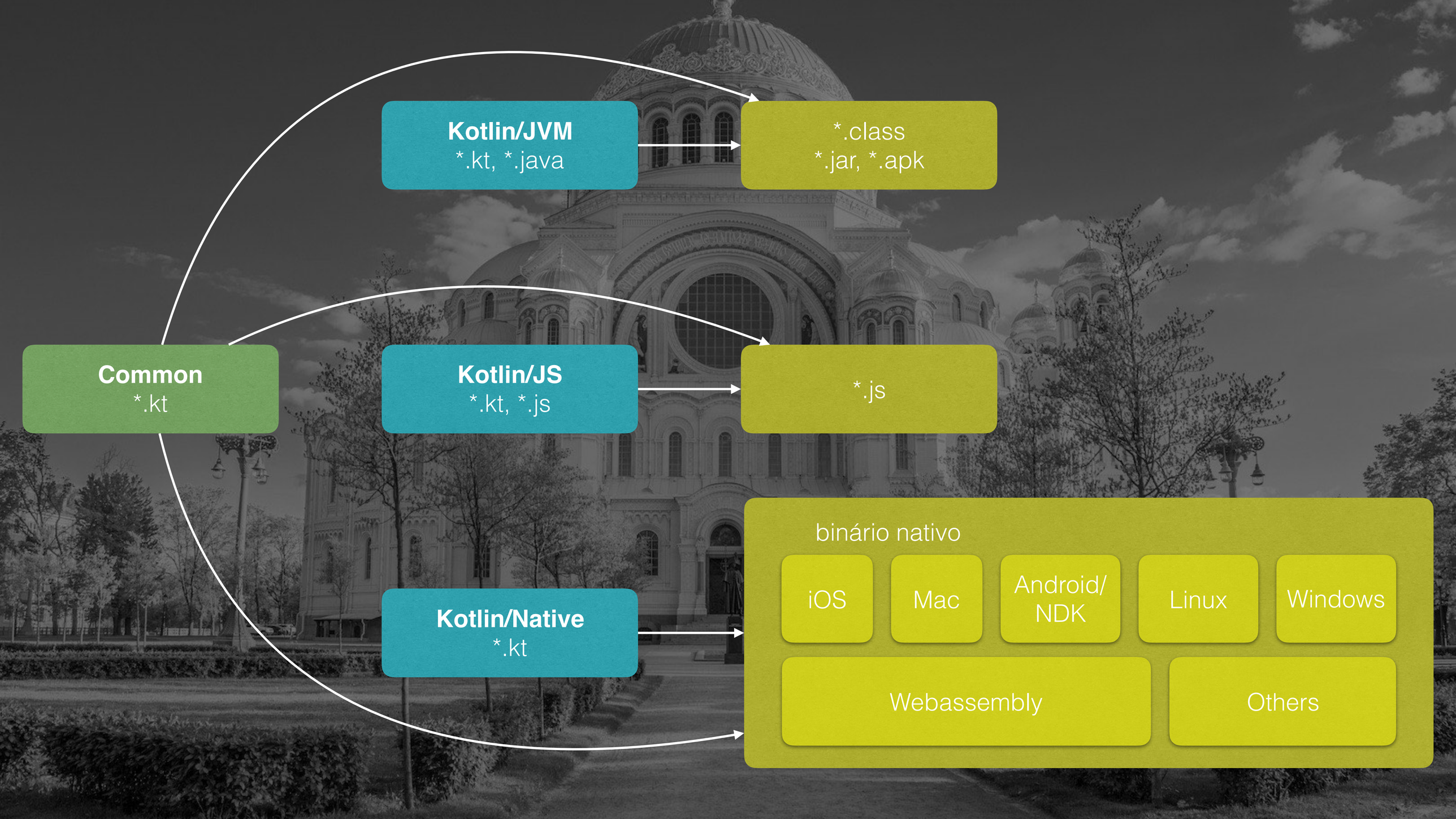
JS

```
object Platform {
    val name: String = "JS"
}

fun hello(): String = "Hello from ${Platform.name}"
```

Native

```
object Platform {
    val name: String = "Native"
}

fun hello(): String = "Hello from ${Platform.name}"
```


*.class
*.jar, *.apk
Common
*.kt
*.js
binário nativo
iOS
Mac
Android/
NDK
Linux
Windows
Webassembly
Outros


Kotlin/JVM
*.kt, *.java
*.class
*.jar, *.apk
Common
*.kt
Kotlin/JS
*.kt, *.js
*.js
Kotlin/Native
*.kt
binário nativo
iOS
Mac
Android/
NDK
Linux
Windows
Webassembly
Others

Common

```
expect object Platform {
    val name: String
}

fun hello(): String = "Hello from ${Platform.name}"
```

Common

```
expect object Platform {
    val name: String
}

fun hello(): String = "Hello from ${Platform.name}"
```

Common

```
expect object Platform {
    val name: String
}

fun hello(): String = "Hello from ${Platform.name}"
```

build.gradle

```
kotlin {
    jvm()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain {
            dependencies {
                implementation kotlin('stdlib-jdk8')
            }
        }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain {
            dependencies {
                implementation kotlin('stdlib-jdk8')
            }
        }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain {
            dependencies {
                implementation kotlin('stdlib-jdk8')
            }
        }
    }
}
```

JVM

```
actual object Platform {
    actual val name: String = "JVM"
}

fun hello(): String = "Hello from JVM"
```

JVM

```
actual object Platform {
    actual val name: String = "JVM"
}

fun hello(): String = "Hello from JVM"
```

JVM

```
actual object Platform {
    actual val name: String = "JVM"
}

fun hello(): String = "Hello from JVM"
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain { /* */ }
        jsMain {
            dependencies {
                implementation kotlin('stdlib-js')
            }
        }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain { /* */ }
        jsMain {
            dependencies {
                implementation kotlin('stdlib-js')
            }
        }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain { /* */ }
        jsMain {
            dependencies {
                implementation kotlin('stdlib-js')
            }
        }
    }
}
```

JS

```
actual object Platform {
    actual val name: String = "JS"
}

fun hello(): String = "Hello from JS"
```

JS

```
actual object Platform {
    actual val name: String = "JS"
}

fun hello(): String = "Hello from JS"
```

JS

```
actual object Platform {
    actual val name: String = "JS"
}

fun hello(): String = "Hello from JS"
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain { /* */ }
        jsMain { /* */ }
        macosMain {
        }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain { /* */ }
        jsMain { /* */ }
        macosMain {
        }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        jvmMain { /* */ }
        jsMain { /* */ }
        macosMain {
        }
    }
}
```

Native

```
actual object Platform {
    actual val name: String = "Native"
}

fun hello(): String = "Hello from Native”
```

Native

```
actual object Platform {
    actual val name: String = "Native"
}

fun hello(): String = "Hello from Native”
```

Native

```
actual object Platform {
    actual val name: String = "Native"
}

fun hello(): String = "Hello from Native”
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        commonTest { /* */ }
        jvmMain { /* */ }
        jvmTest { /* */ }
        jsMain { /* */ }
        jsTest { /* */  }
        macosMain { /* */ }
        macosTest { /* */ }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        commonTest { /* */ }
        jvmMain { /* */ }
        jvmTest { /* */ }
        jsMain { /* */ }
        jsTest { /* */  }
        macosMain { /* */ }
        macosTest { /* */ }
    }
}
```

build.gradle

```
kotlin {
    jvm()
    js()
    macosX64("macos")
    sourceSets {
        commonMain { /* */ }
        commonTest { /* */ }
        jvmMain { /* */ }
        jvmTest { /* */ }
        jsMain { /* */ }
        jsTest { /* */  }
        macosMain { /* */ }
        macosTest { /* */ }
    }
}
```

build.gradle

```
kotlin {
    /* */
    sourceSets {
        /* */
        commonTest {
            dependencies {
                implementation kotlin('test-common')
                implementation kotlin('test-annotations-common')
            }
        }
        jvmTest {
            dependencies {
                implementation kotlin('test')
                implementation kotlin('test-junit')
            }
        }
        jsTest {
            dependencies {
                implementation kotlin('test-js')
            }
        }
        macosTest {
        }
    }
}
```

JVM

```
import kotlin.test.Test
import kotlin.test.assertTrue

class SampleTestsJVM {
    @Test
    fun testHello() {
        assertTrue("JVM" in hello())
    }
}
```

JS

```
import kotlin.test.Test
import kotlin.test.assertTrue

class SampleTestsJS {
    @Test
    fun testHello() {
        assertTrue("JS" in hello())
    }
}
```

Native

```
import kotlin.test.Test
import kotlin.test.assertTrue

class SampleTestsNative {
    @Test
    fun testHello() {
        assertTrue("Native" in hello())
    }
}
```

Common

```
expect class Sample() {
    fun checkMe(): Int
}

object Platform {
    val name: String
}

fun hello(): String = "Hello from ${Platform.name}"
```

Common

```
import kotlin.test.Test
import kotlin.test.assertTrue

class SampleTests {
    @Test
    fun testMe() {
        assertTrue(Sample().checkMe() > 0)
    }
}
```


Bibliotecas

Ktor

kotlinx.serialization

sqldelight

Acesso a arquivos

Melhor suporte a testes

kotlinx.io

MPSettings

Data

Estado de UI

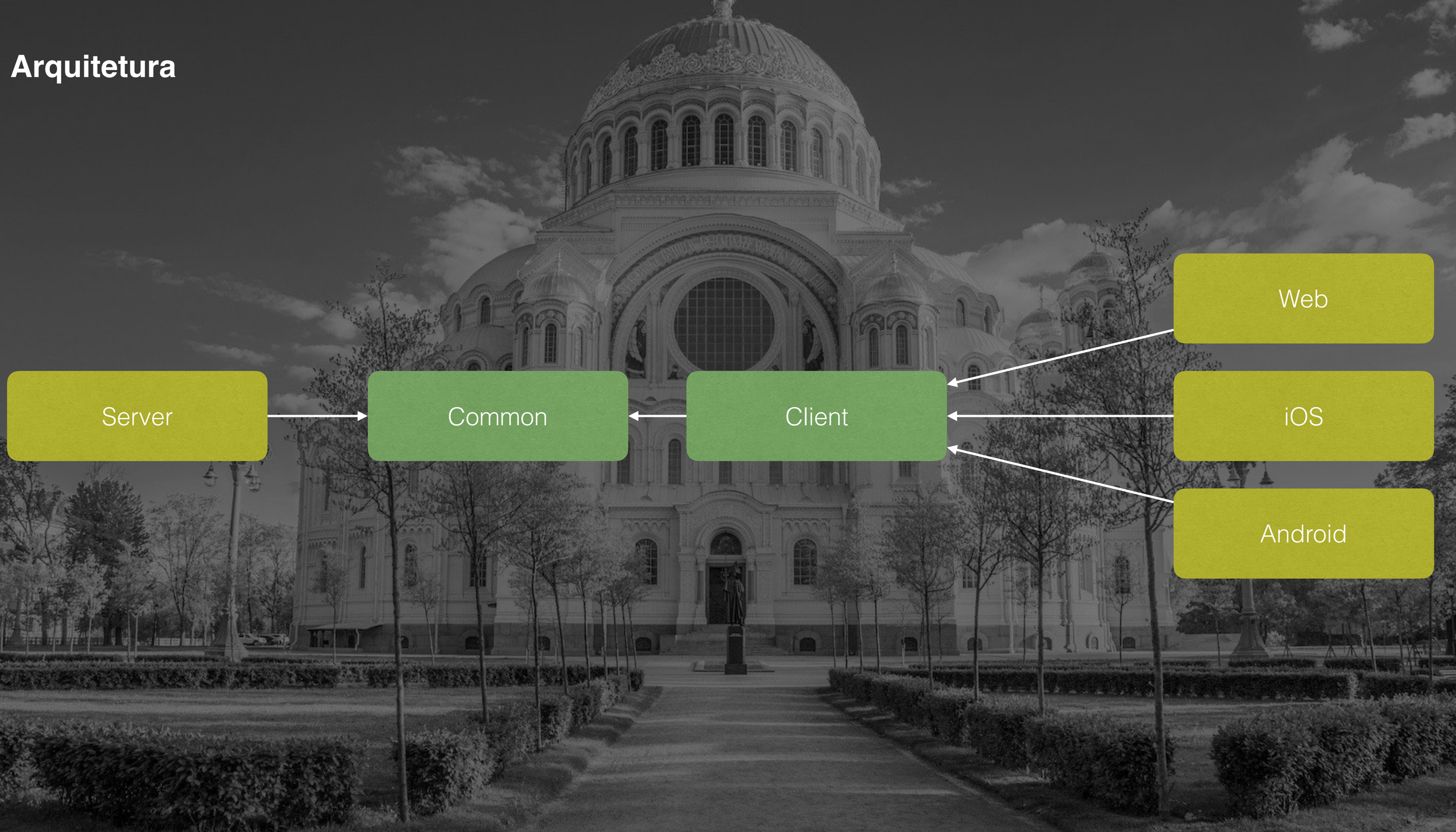
KorLibs

Kotlinx.Coroutines

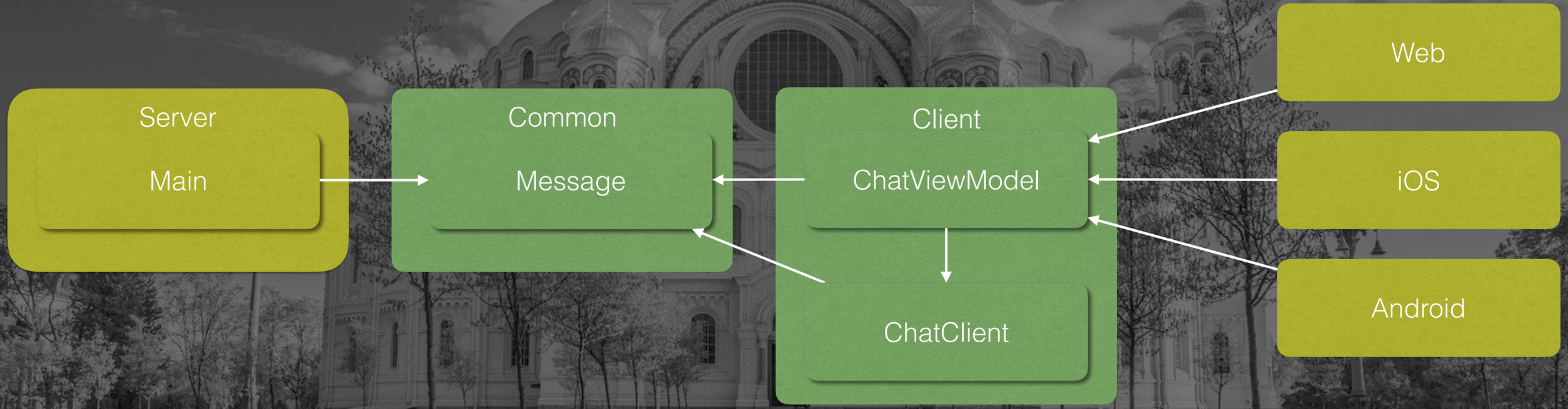
Stately

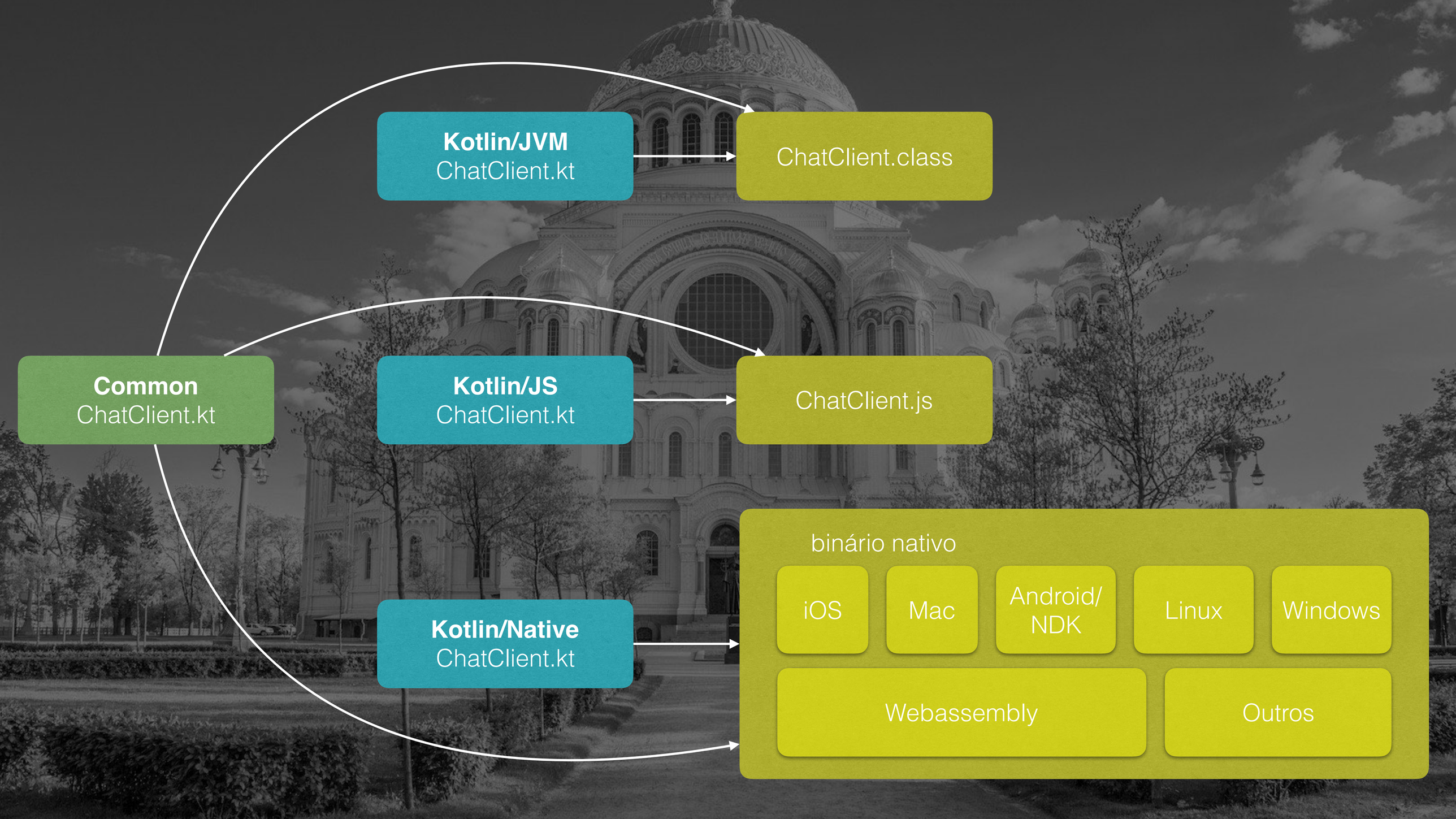
MVP de um Chat Multiplataforma

Arquitetura



Arquitetura




Kotlin/JVM
ChatClient.kt
ChatClient.class
Common
ChatClient.kt
Kotlin/JS
ChatClient.kt
ChatClient.js
Kotlin/Native
ChatClient.kt
binário nativo
iOS
Mac
Android/
NDK
Linux
Windows
Webassembly
Outros

Common

```
internal expect open class ChatClient(url: String) {
    open fun start()
    open fun send(message: Message)
    open fun receive(receiveBlock: (Message) -> Unit)
    open fun onFailure(throwableBlock: (Throwable) -> Unit)
}
```

Common

```
internal expect open class ChatClient(url: String) {
    open fun start()
    open fun send(message: Message)
    open fun receive(receiveBlock: (Message) -> Unit)
    open fun onFailure(throwableBlock: (Throwable) -> Unit)
}
```

Common

```
internal expect open class ChatClient(url: String) {
    open fun start()
    open fun send(message: Message)
    open fun receive(receiveBlock: (Message) -> Unit)
    open fun onFailure(throwableBlock: (Throwable) -> Unit)
}
```

Android

```
internal actual open class ChatClient actual constructor(val url: String) :
WebSocketListener() {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */  }

    override fun onMessage(webSocket: WebSocket, text: String) { /* */ }

    override fun onFailure(webSocket: WebSocket, t: Throwable, response:
Response?) { /* */ }
}
```

Android

```
internal actual open class ChatClient actual constructor(val url: String) :
WebSocketListener() {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */  }

    override fun onMessage(webSocket: WebSocket, text: String) { /* */ }

    override fun onFailure(webSocket: WebSocket, t: Throwable, response:
Response?) { /* */ }
}
```

Android

```
internal actual open class ChatClient actual constructor(val url: String) :
WebSocketListener() {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */  }

    override fun onMessage(webSocket: WebSocket, text: String) { /* */ }

    override fun onFailure(webSocket: WebSocket, t: Throwable, response:
Response?) { /* */ }
}
```

JS

```
internal actual open class ChatClient actual constructor(val url: String) {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }
}
```

JS

```
internal actual open class ChatClient actual constructor(val url: String) {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }
}
```

JS

```
internal actual open class ChatClient actual constructor(val url: String) {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }
}
```

iOS

```
internal actual open class ChatClient actual constructor(val url: String) {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }

    inner class WebSocketDelegate : NSObject(), SRWebSocketDelegateProtocol {
        override fun webSocket(webSocket: SRWebSocket?, didReceiveMessage:
Any?) { /* */ }
    }
}
```

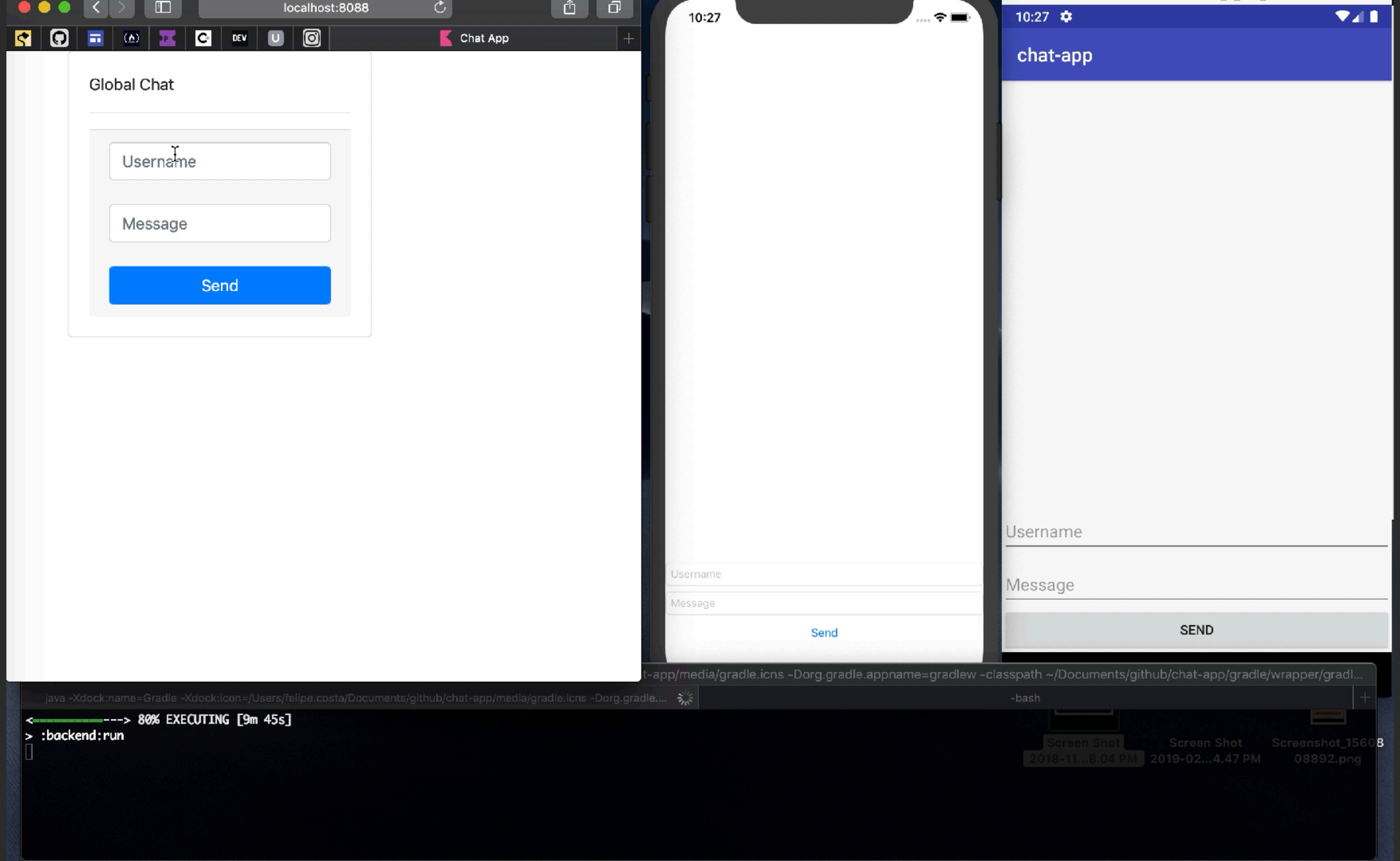
iOS

```
internal actual open class ChatClient actual constructor(val url: String) {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }

    inner class WebSocketDelegate : NSObject(), SRWebSocketDelegateProtocol {
        override fun webSocket(webSocket: SRWebSocket?, didReceiveMessage:
Any?) { /* */ }
    }
}
```

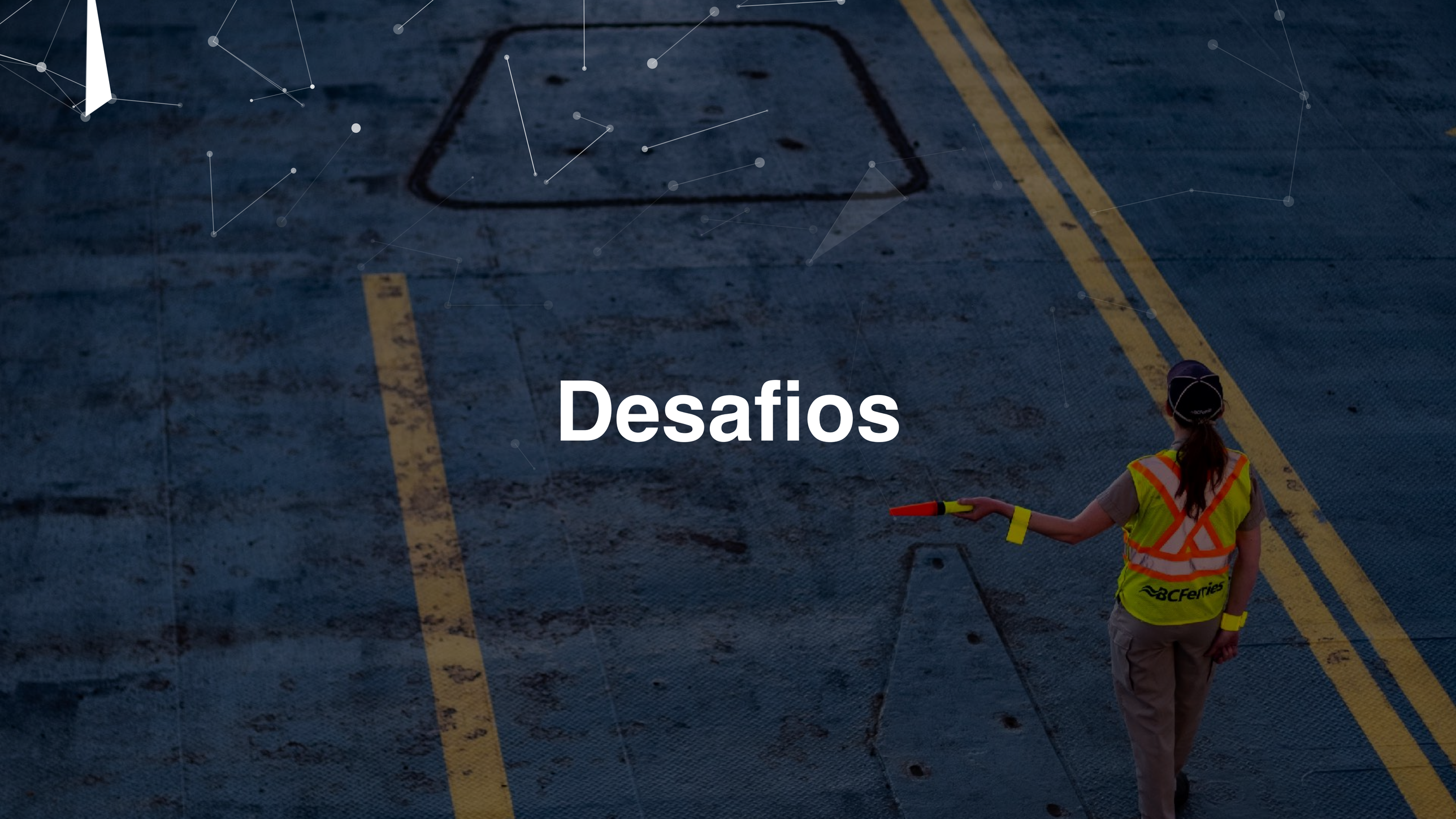
iOS

```
internal actual open class ChatClient actual constructor(val url: String) {

    actual open fun start() { /* */ }

    actual open fun send(message: Message) { /* */ }

    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }

    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }

    inner class WebSocketDelegate : NSObject(), SRWebSocketDelegateProtocol {
        override fun webSocket(webSocket: SRWebSocket?, didReceiveMessage:
Any?) { /* */ }
    }
}
```


Demo


localhost:8088
Chat App
Global Chat
Username
Message
Send
10:27
Username
Message
Send
10:27
chat-app
Username
Message
SEND
java -Xdock:name=Gradle -Xdock:icon=/Users/felipe.costa/Documents/github/chat-app/media/gradle.icns -Dorg.gradle....
-bash
<---------> 80% EXECUTING [9m 45s]
> :backend:run

Desafios



Experimental


Threads

Debug


Unknown property 'justify-content', Line 86, Column 11
Sublime Text 2


Tempo de
compilação

MVP de Chat entregue


Thank
You

Questions
Answers

felipe.costa@olxbr.com




felipehjcosta