



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.

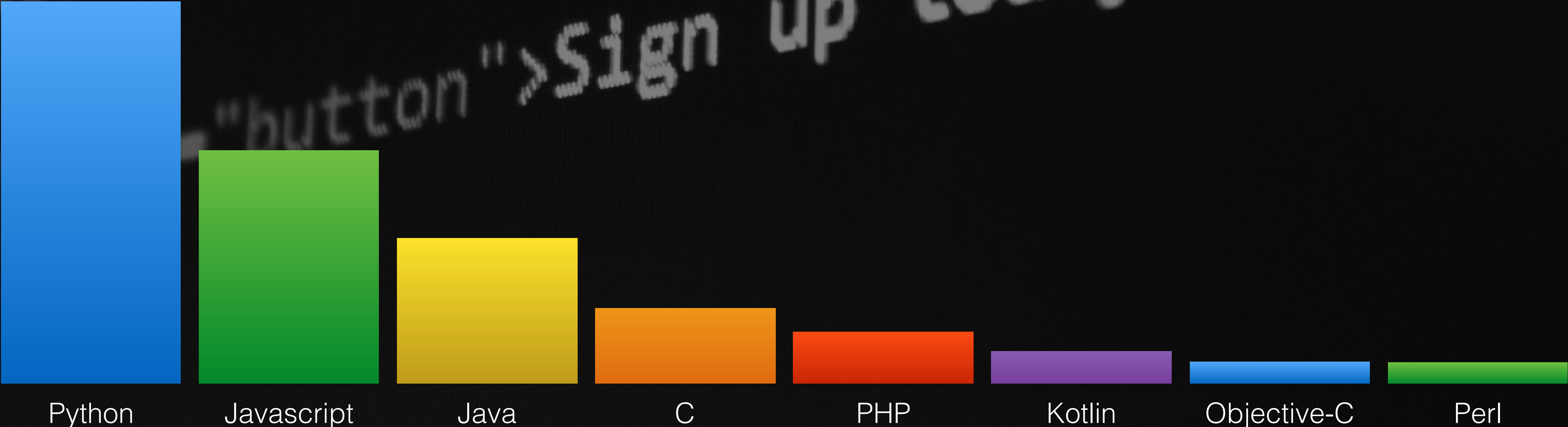
The OLX logo is displayed in a large, white, bold, sans-serif font. The 'O' is a simple circle, the 'L' is a vertical bar with a horizontal top bar, and the 'X' is formed by two intersecting diagonal bars. The logo is centered horizontally in the upper half of the image.

OLX

Nosso Propósito

Aproximamos Brasileiros para transformar itens em felicidade.

Kotlin @ OLY







**Karol
&**

MVP de um chat multiplataforma



**Por quê reutilizar código entre
plataformas?**

DRY


A large, empty stadium with rows of red seats receding into the distance under a dark sky. The seats are arranged in a curved pattern, and the lighting is dim, creating a sense of vastness and emptiness.

Mitigação de risco



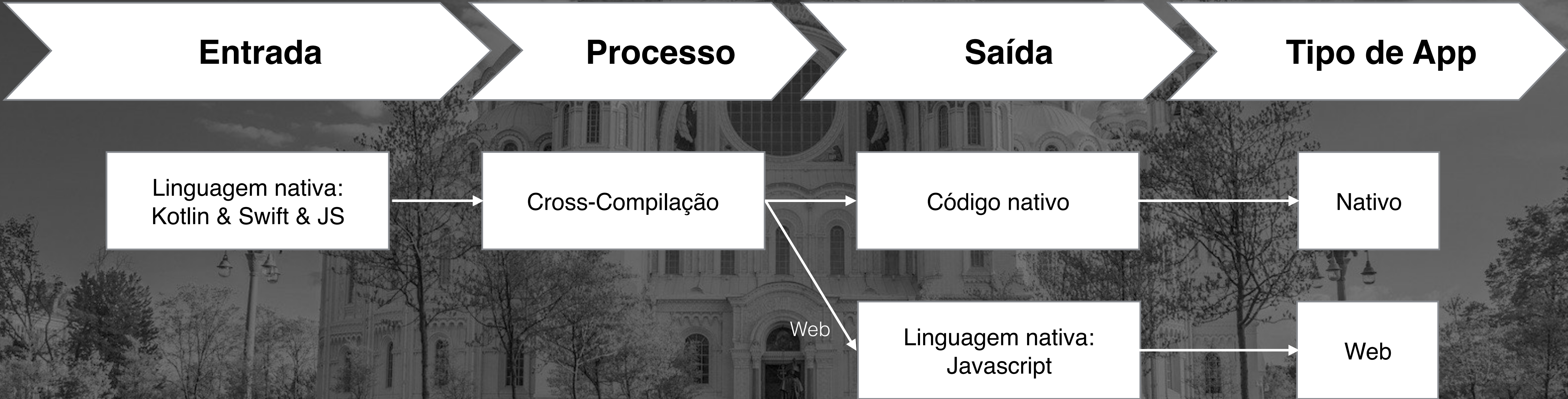
Paridade de
funcionalidades



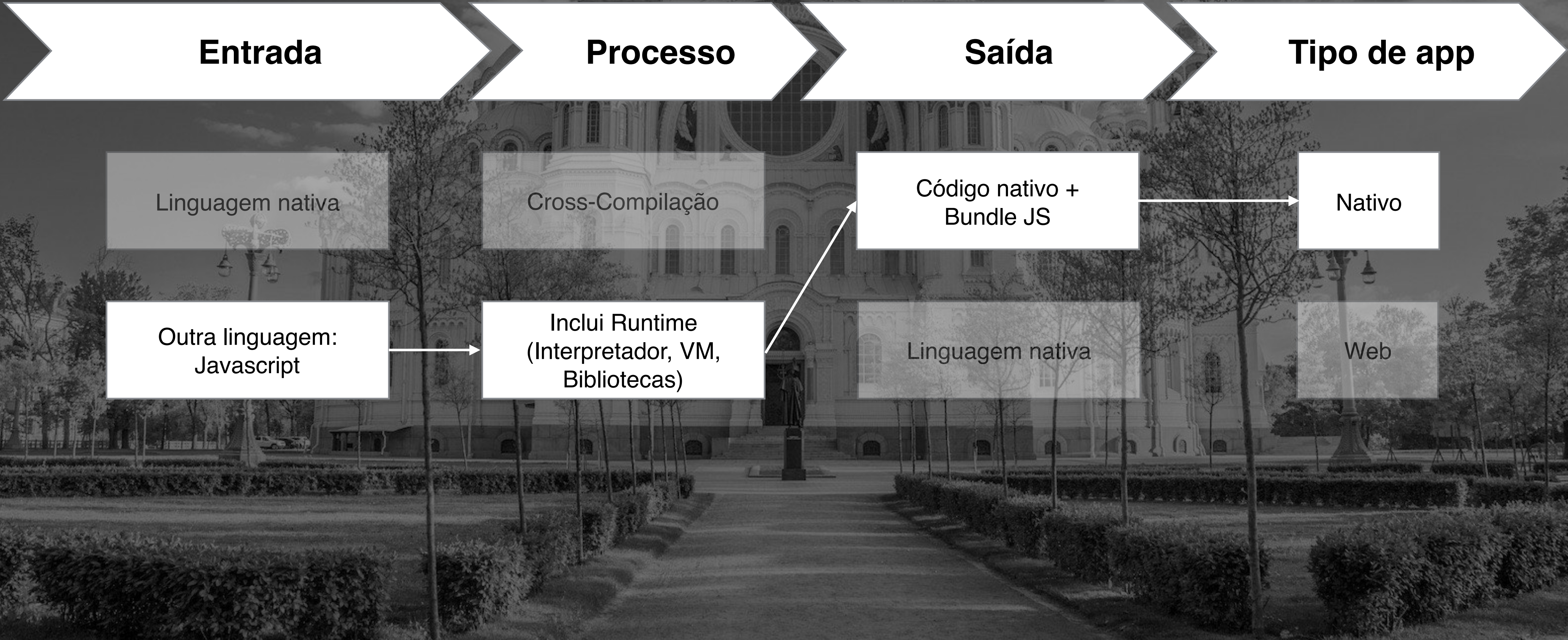
A wooden signpost with four directional arrows is positioned in a field. The signpost is made of weathered wood and has four arrows pointing in different directions. The background is a blurred landscape with a path leading into the distance. Overlaid on the image is a network diagram consisting of white dots connected by thin lines, representing a complex network or data structure. The word "Soluções" is written in a large, white, sans-serif font across the middle of the image, partially overlapping the signpost and the network diagram.

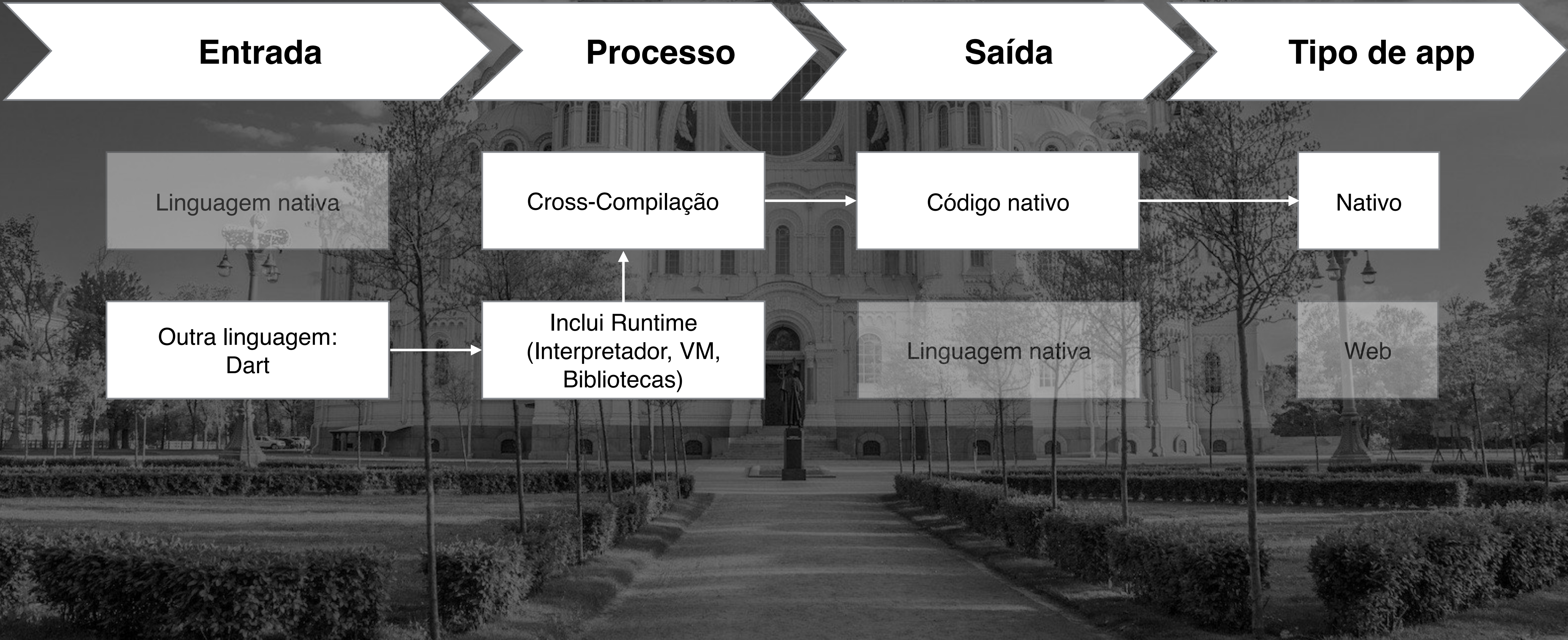
Soluções

Nativo

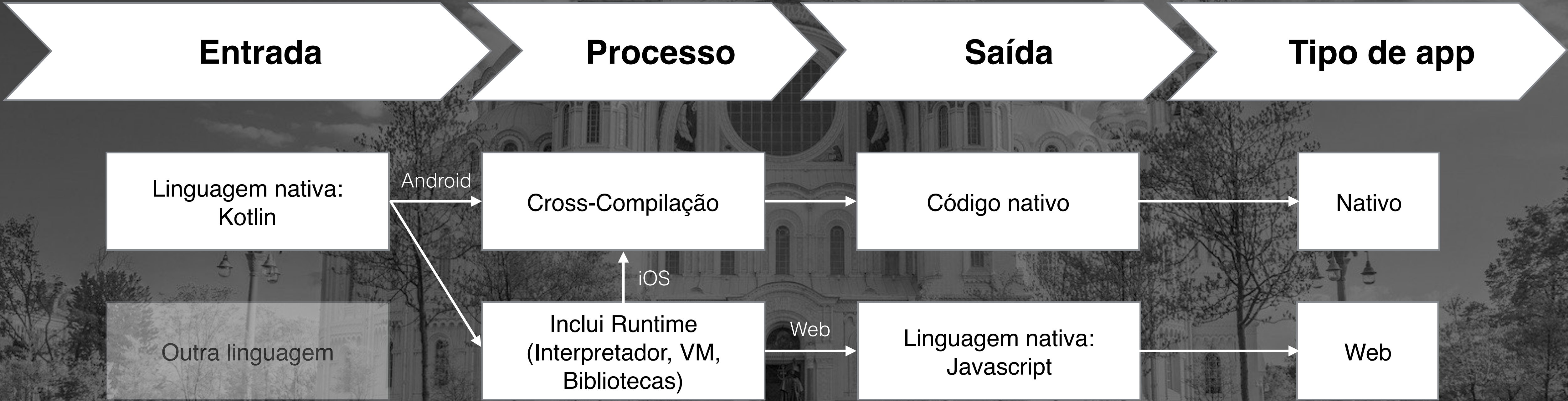


React Native





Kotlin Multiplataforma



A misty forest scene with a wooden bridge over a stream, overlaid with a white geometric network diagram. The text "Por quê Kotlin Multiplataforma?" is centered in white.

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



Muitas plataformas

```
5 abort("The Rails environment is running in production mode!")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line.
22
23 # Requires supporting ruby files with support/ and its subdirectories.
24 # spec/support/ and its subdirectories. These files will be
25 # run as spec files by default. You can use the
26 # in _spec.rb will both be required and run as spec files.
27 # run twice. It is recommended that you use the
28 # end with _spec.rb. You can configure the
29 # option on the command line as follows:
30 # rails spec --require=spec_helper.rb --require=spec_helper.rb --require=spec_helper.rb
31 # for 'mongoid'
```




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')  
            }  
        }  
    }  
}
```

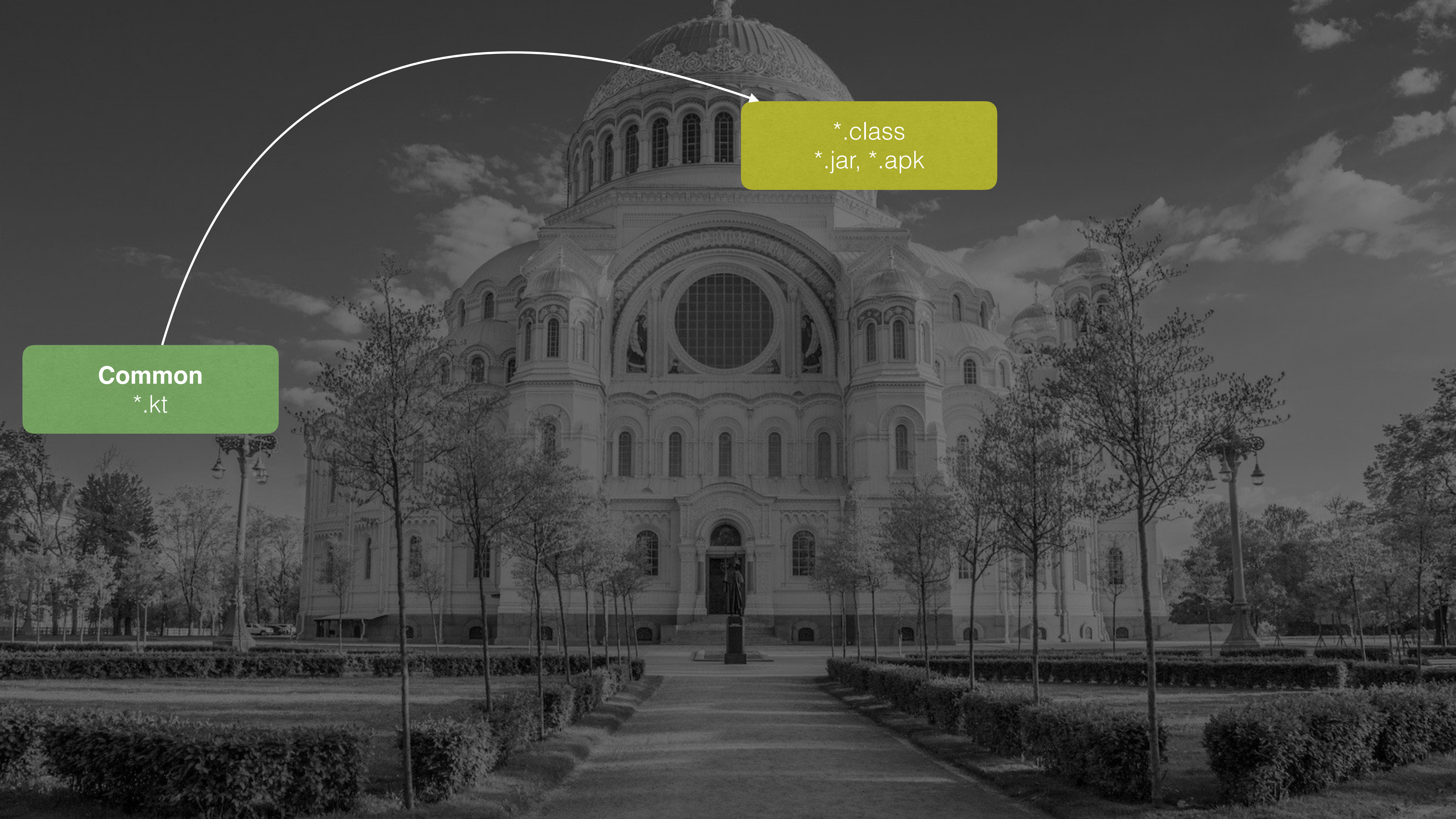


Common
*.kt

Common

*.kt

*.class
*.jar, *.apk



Build.gradle

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


Build.gradle

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

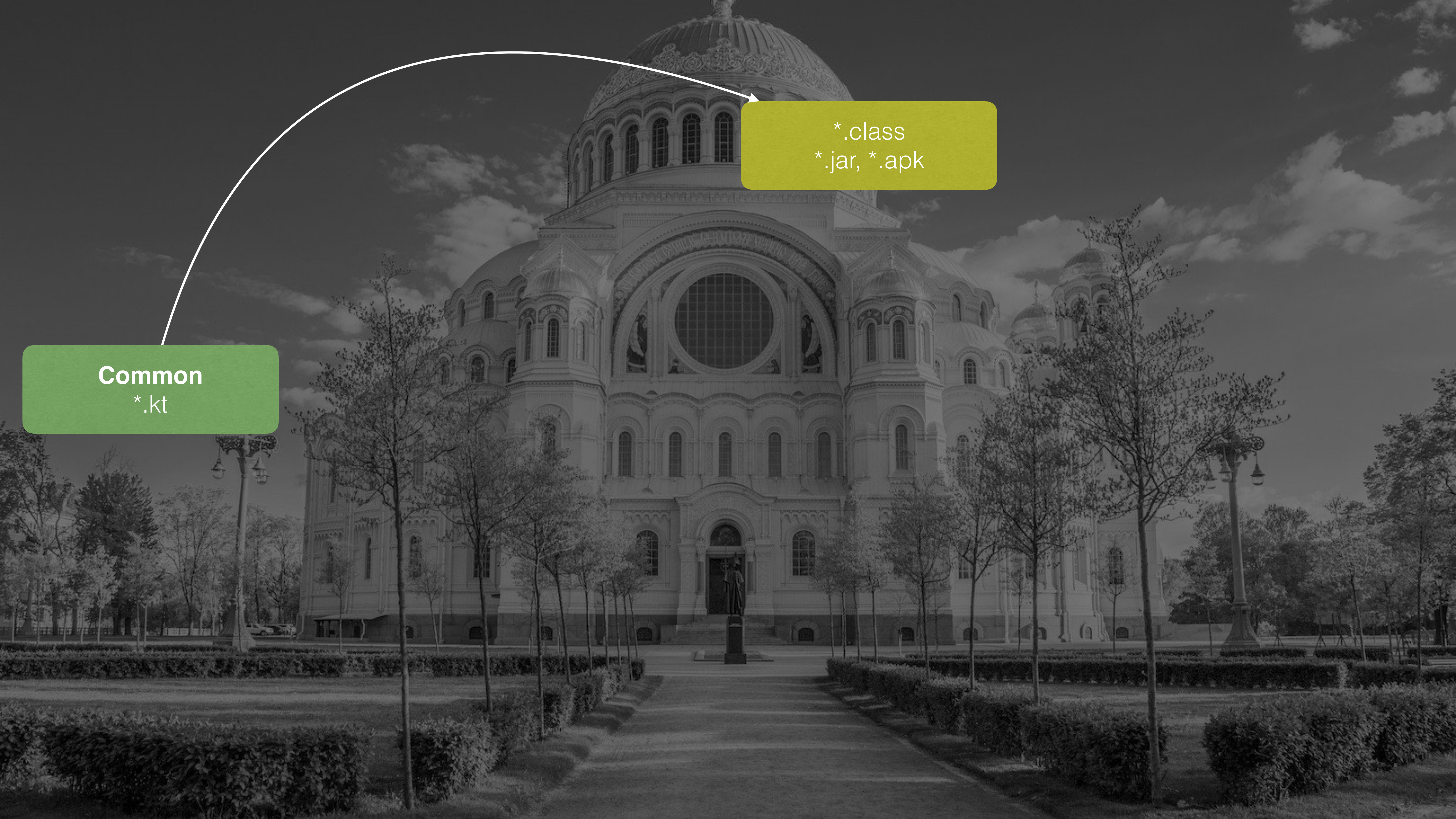
Build.gradle

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

Common

*.kt

*.class
*.jar, *.apk



Common
*.kt

*.class
*.jar, *.apk

*.js

Build.gradle

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

Build.gradle

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

Build.gradle

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

Common
*.kt

*.class
*.jar, *.apk

*.js

Common
*.kt

*.class
*.jar, *.apk

*.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";
    }
}
```

```
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}"
```

Common
*.kt

*.class
*.jar, *.apk

*.js

binário nativo

iOS

Mac

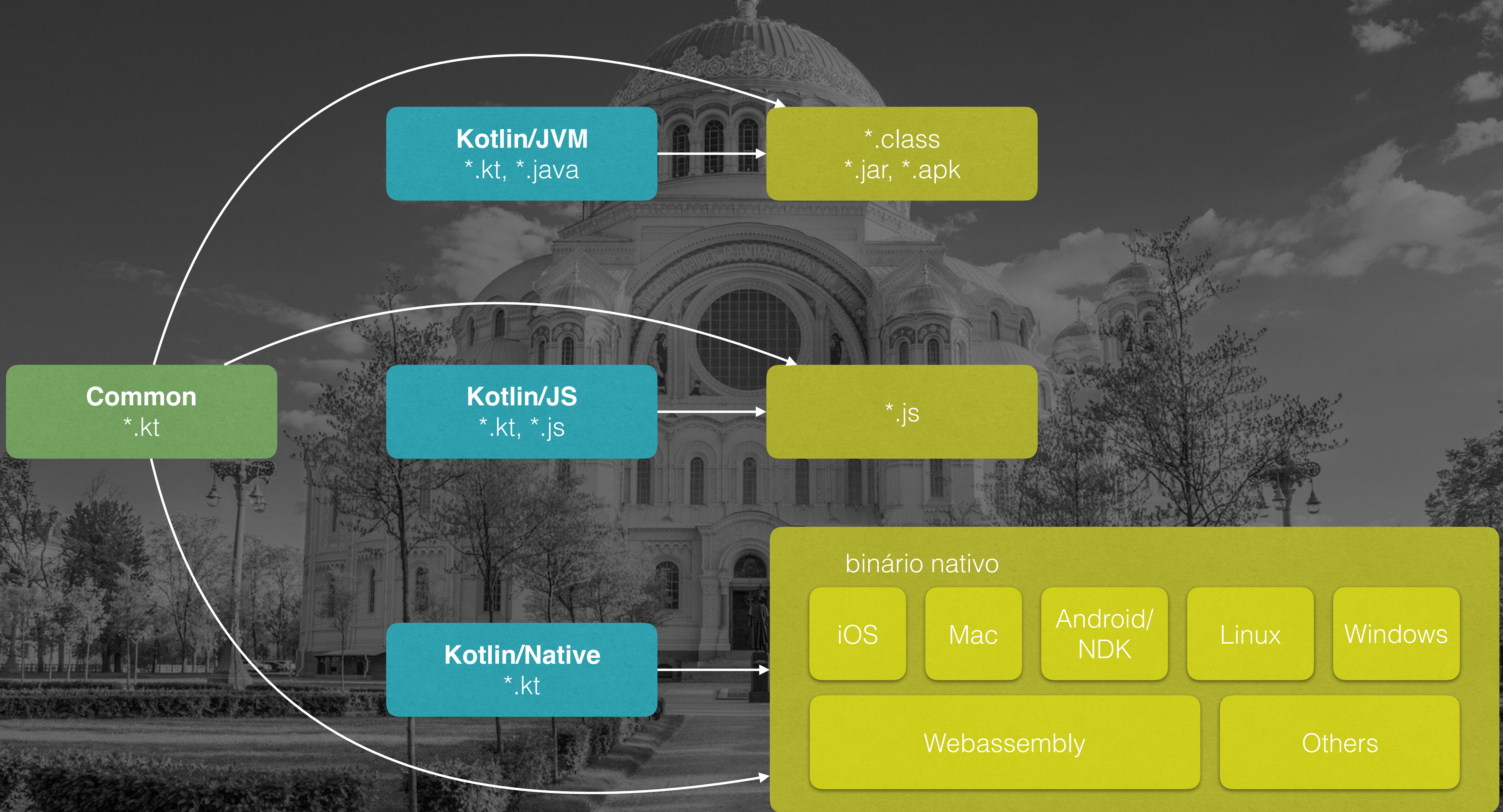
Android/
NDK

Linux

Windows

Webassembly

Outros



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()  
    macOS64("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)
    }
}
```

A wide-angle, low-perspective shot of a grand, historic library. The room is filled with tall, dark wood bookshelves reaching up to a high, intricately carved wooden ceiling. The shelves are packed with books. In the center, a mezzanine level with a red carpeted staircase is visible. Below, a study area features several wooden desks with computer monitors and laptops. The lighting is warm and focused, highlighting the architectural details and the vast collection of books.

Bibliotecas

Ktor

kotlinx.serialization

sqlDelight

Acesso a arquivos

Melhor suporte a testes



kotlinx.io

MPSettings

Data

Estado de UI

KorLibs

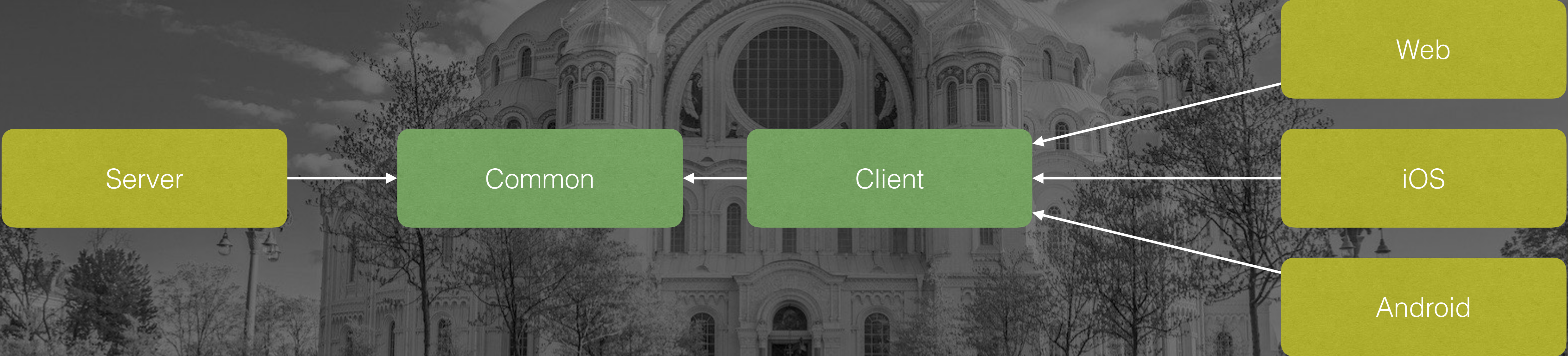
Kotlinx.coroutines

Stately

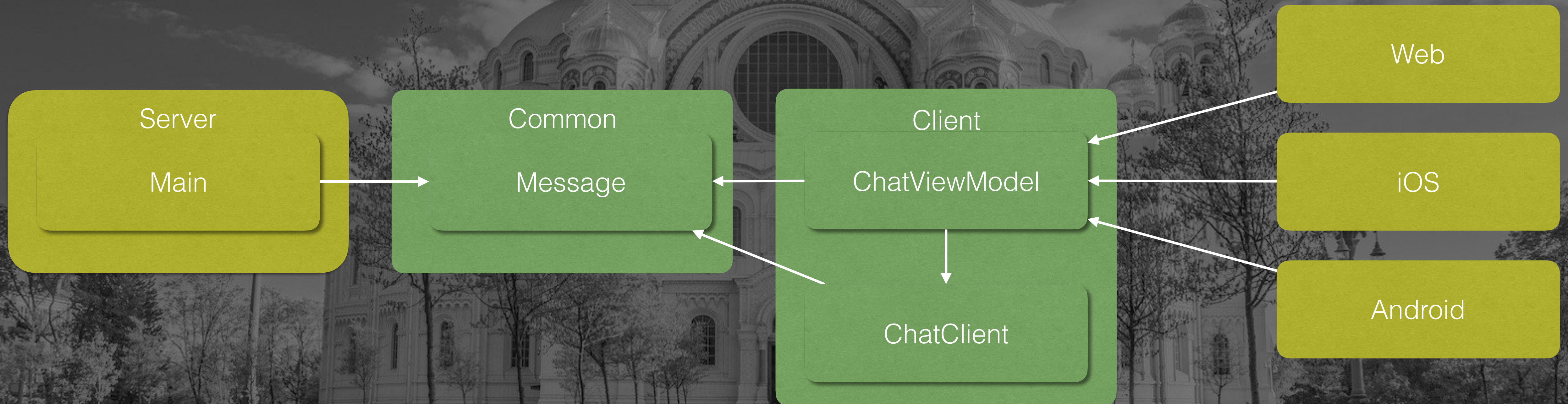


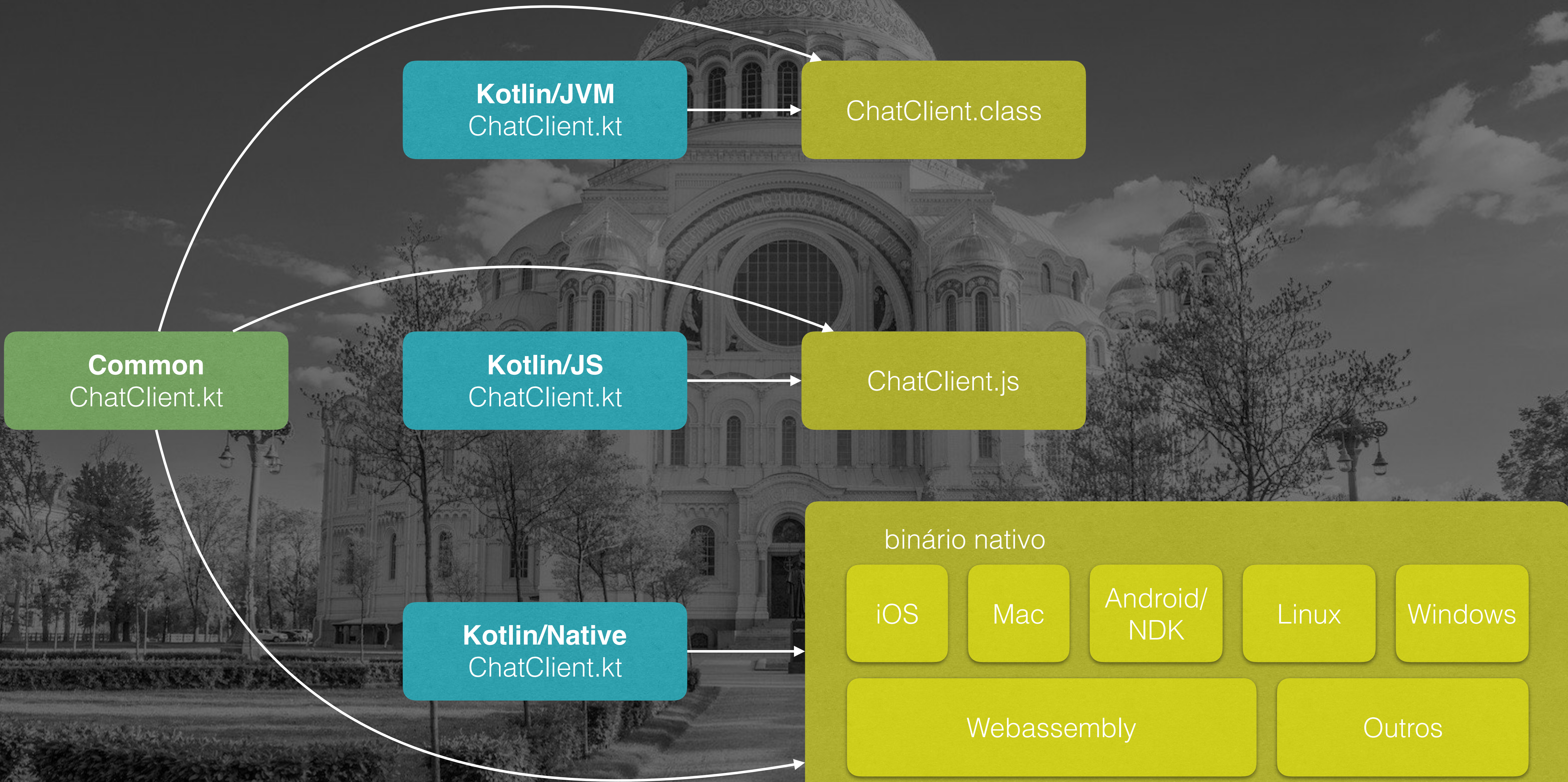
MVP de um Chat Multiplataforma

Arquitetura



Arquitectura





Common
ChatClient.kt

Kotlin/JVM
ChatClient.kt

ChatClient.class

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?) { /* */ }
}
```

```
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) { /* */ }  
}
```

```
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) { /* */ }  
}
```

```
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) { /* */ }  
}
```

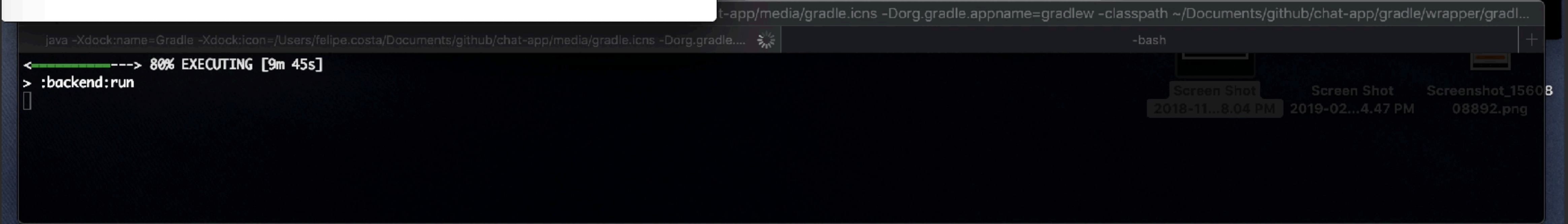
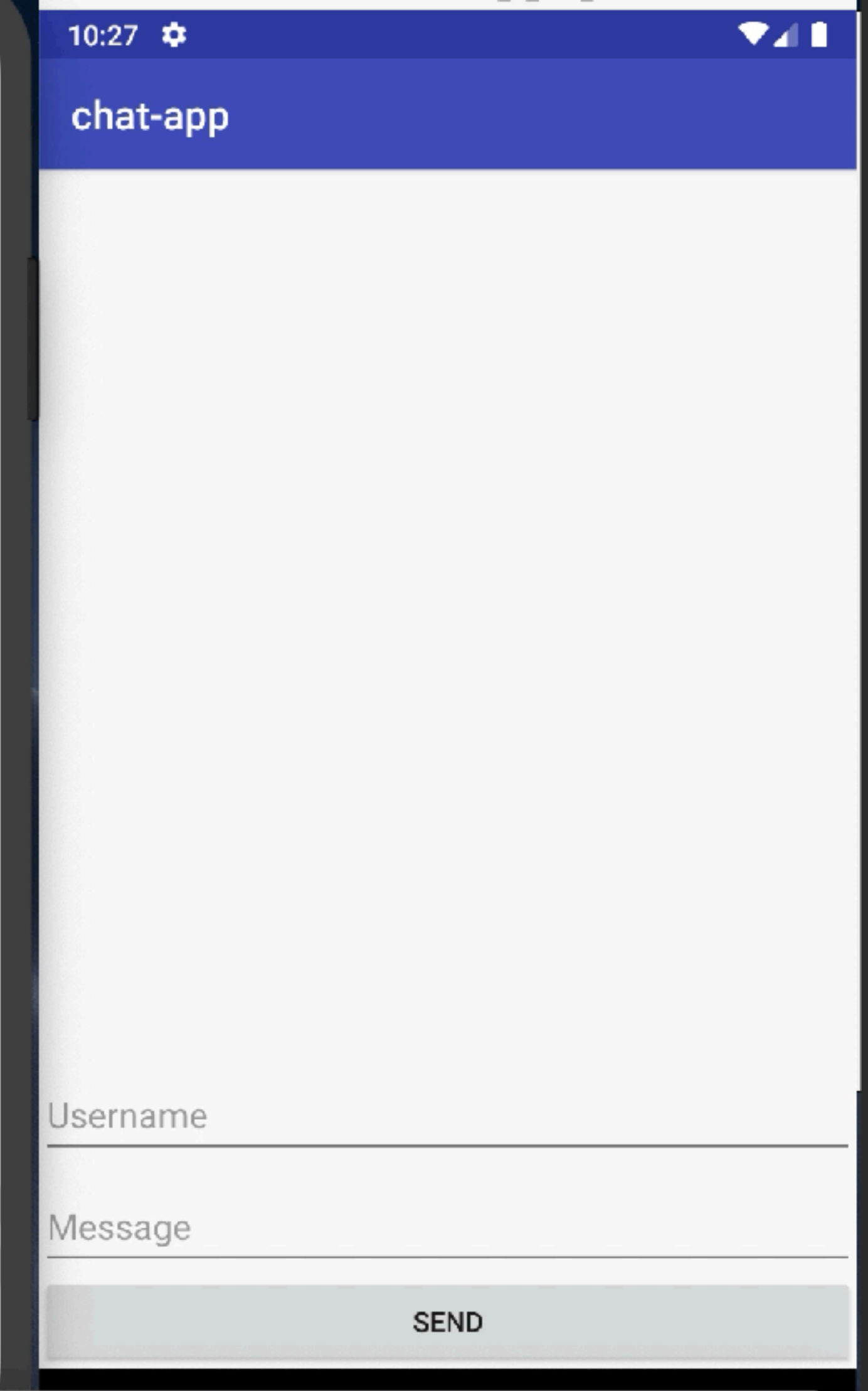
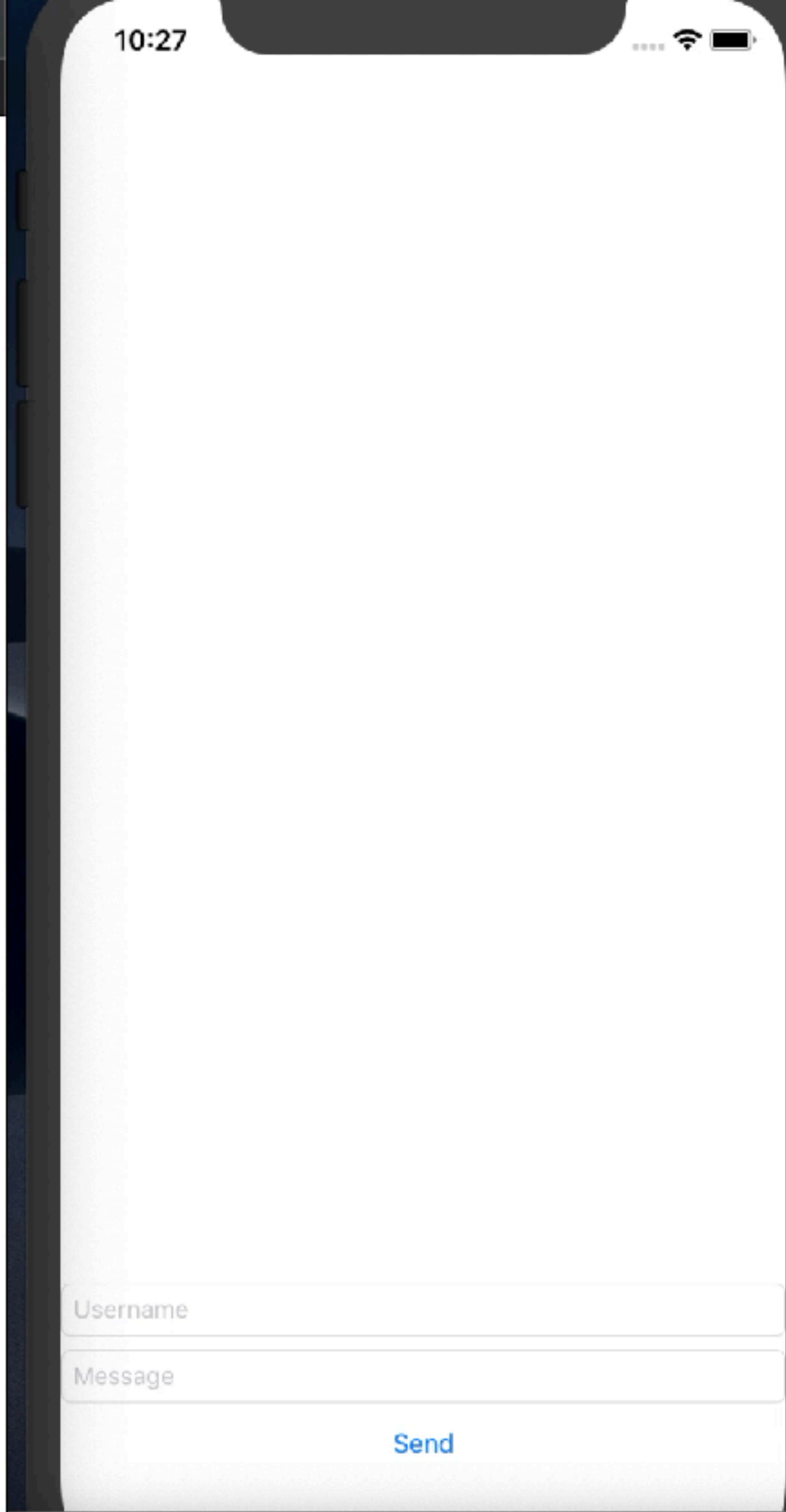
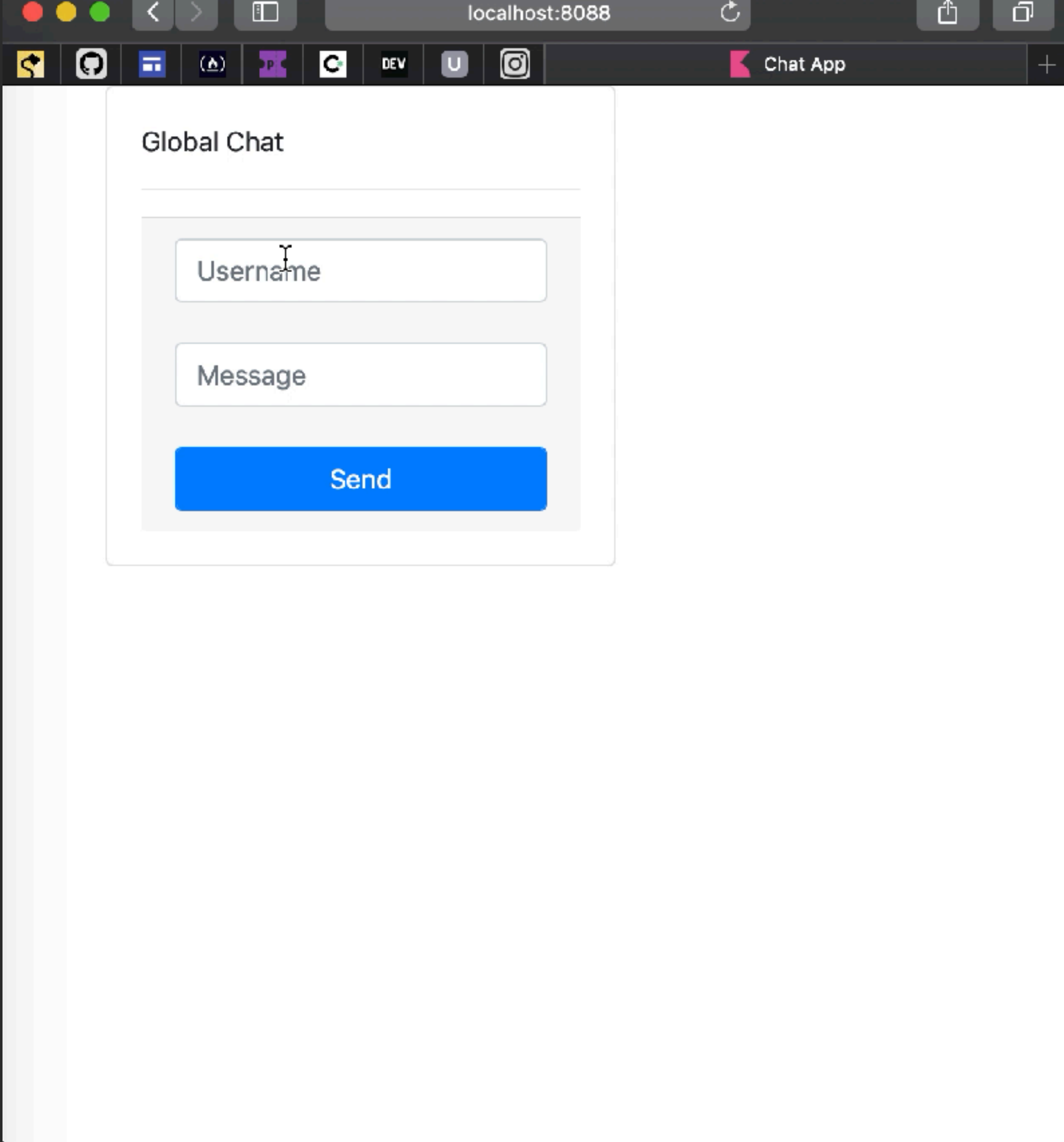
```
internal actual open class ChatClient actual constructor(val url: String) {  
    actual open fun start() { /* */ }  
    actual open fun send(message: Message) { /* */ }  
    actual open fun receive(receiveBlock: (Message) -> Unit) { /* */ }  
    actual open fun onFailure(throwableBlock: (Throwable) -> Unit) { /* */ }  
    inner class WebSocketDelegate : NSObject(), SRWebSocketDelegateProtocol {  
        override fun websocket(webSocket: SRWebSocket?, didReceiveMessage:  
Any?) { /* */ }  
    }  
}
```

```
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?) { /* */ }  
    }  
}
```

```
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?) { /* */ }  
    }  
}
```

A person is sitting at a desk, working on a laptop. The laptop screen displays a code editor with Java code. To the left of the laptop, there is a smartphone connected to the laptop via a cable. A coffee cup is also on the desk. The word "Demo" is overlaid in the center of the image. The background is a blurred office setting with a window and some papers. There are also some glasses on the desk to the right of the laptop.

Demo



Desafios



Experimental



Threads



Debug



Tempo de compilação

MVP de Chat entregue





Thank

You

Questions Answers

felipe.costa@olxbr.com



[felipehjcosta](#)