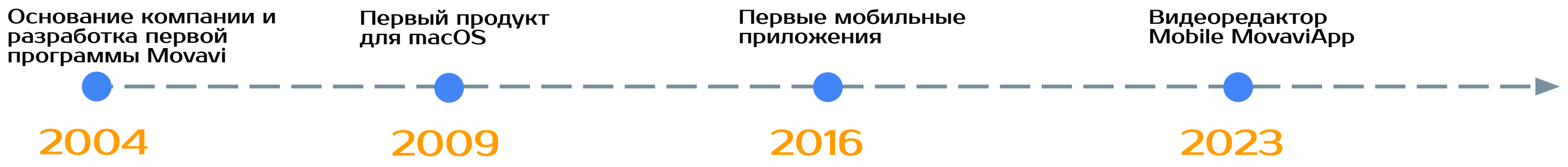
movavi

Рецепт приготовления кроссплатформенного мобильного видеоредактора

Дмитрий Кузнецов
Руководитель отдела мобильной разработки


Основание компании и разработка первой программы Movavi
2004
Первый продукт для macOS
2009
Первые мобильные приложения
2016
Видеоредактор Mobile MovaviApp
2023

Какой продукт хотим сделать?

Меню мобильных продуктов

Видеоредактор
Clips

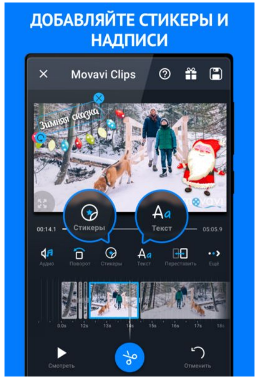
2016

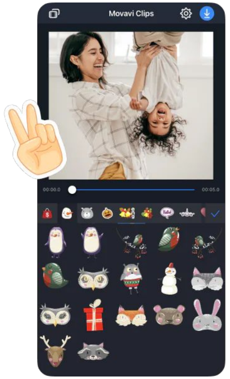
Фоторедактор
Picverse

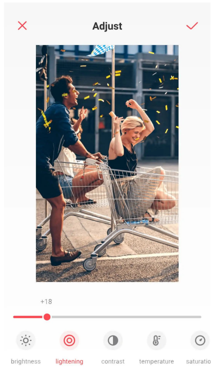
2019

Суперапп
Mobile MovaviApp

2023





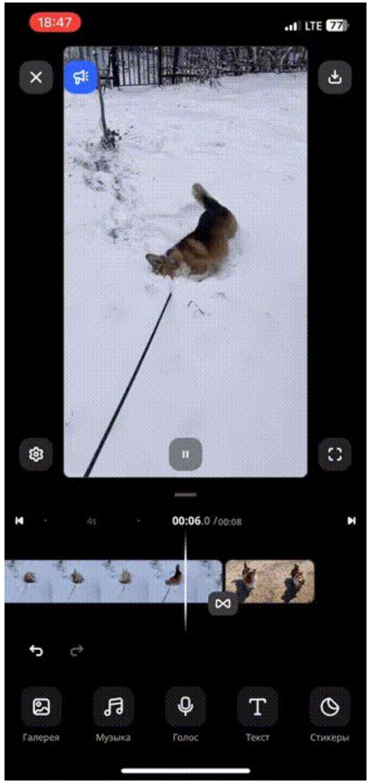


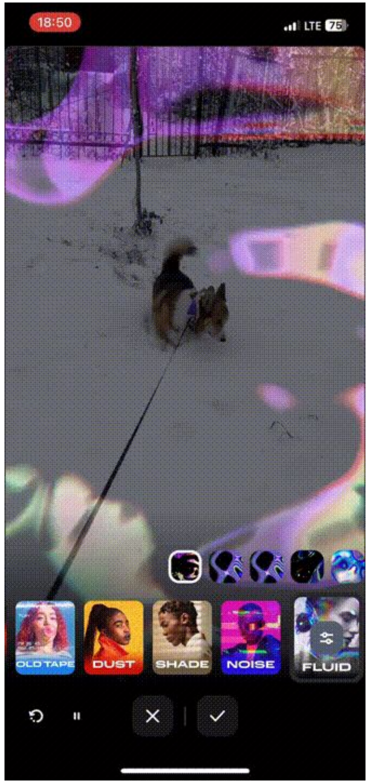
Технологии мобильных продуктов

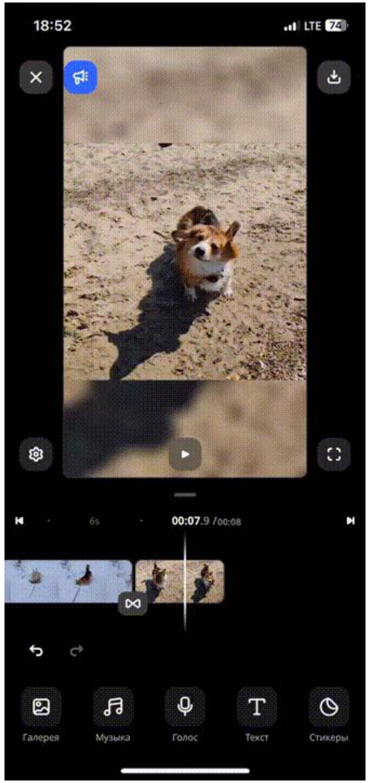
Видеоредактор Clips	Фоторедактор Picverse	Эксперименты с Kotlin Multiplatform	Суперапп Mobile MovaviApp
2016	2019	2021	2023
C++	Kotlin	Kotlin (KMP)	C++
Java	Swift	Swift	Kotlin (KMP)
Kotlin			Swift
Objective-C			
Swift			

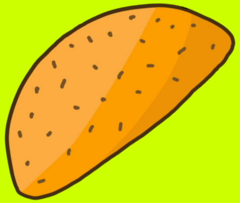
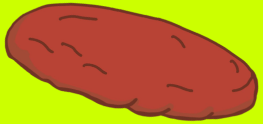
Mobile MovaviApp 2023

Суперапп - видео и фоторедактор в одном приложении.

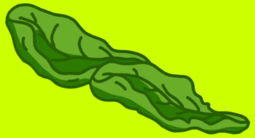




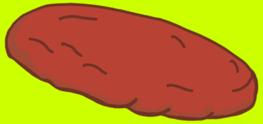


Ингредиенты



Обработка мультимедиа

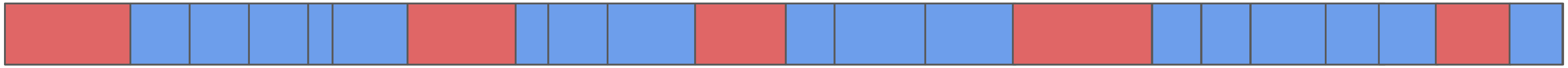


Обработка мультимедиа Медиаконтейнер

Формат файла - определяет спецификацию хранения данных в файле.

Примеры: 3gp, MP4, MOV, AVI, WAV, Ogg, MKV.

Обработка мультимедиа
Parser, demuxer

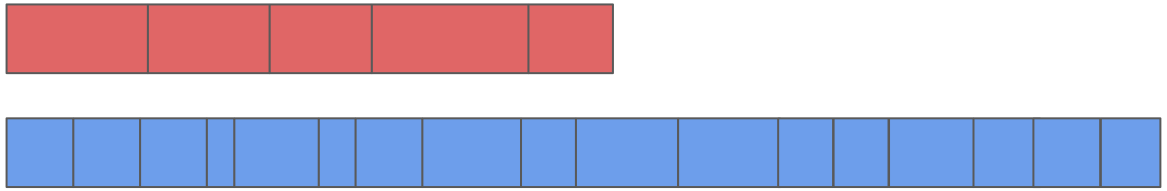


- Количество мультимедийных потоков
- Длительность
- Тип
- Позиционирование
- Чтение пакетов закодированных данных

🟥 Видео

🟦 Аудио

Обработка мультимедиа Decoder

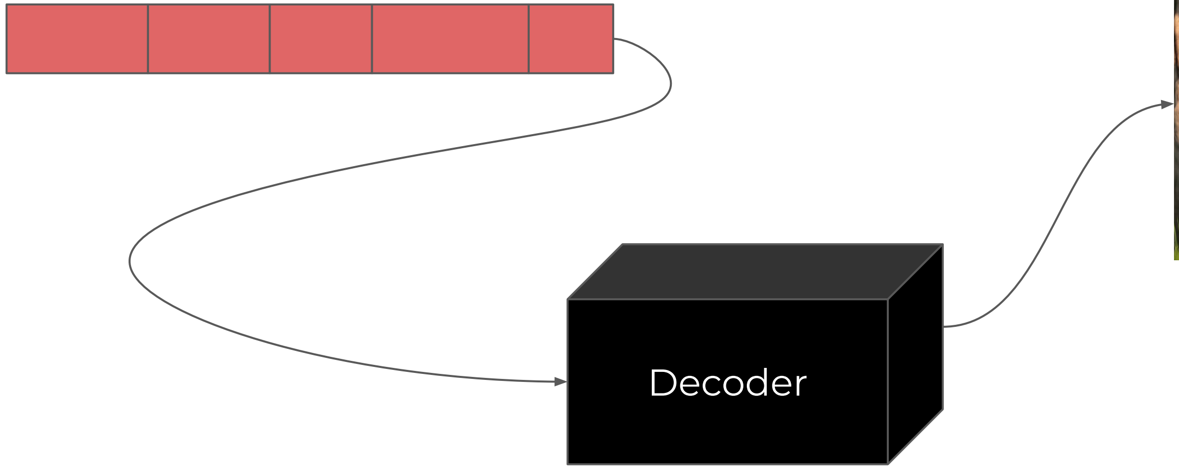


- Вход: пакет закодированных данных
- Выход: кадр видео или звуковой сэмпл, в зависимости от типа потока

🟥 Видео

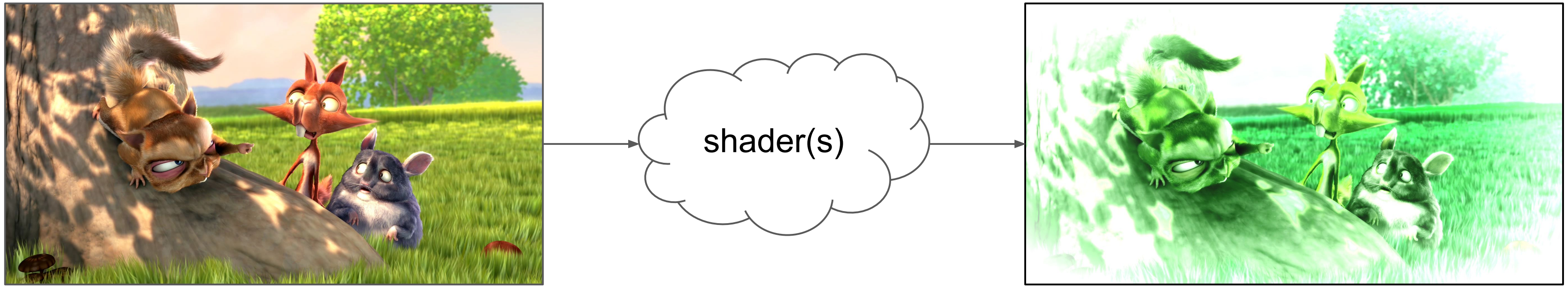
🟦 Аудио

Обработка мультимедиа Decoder

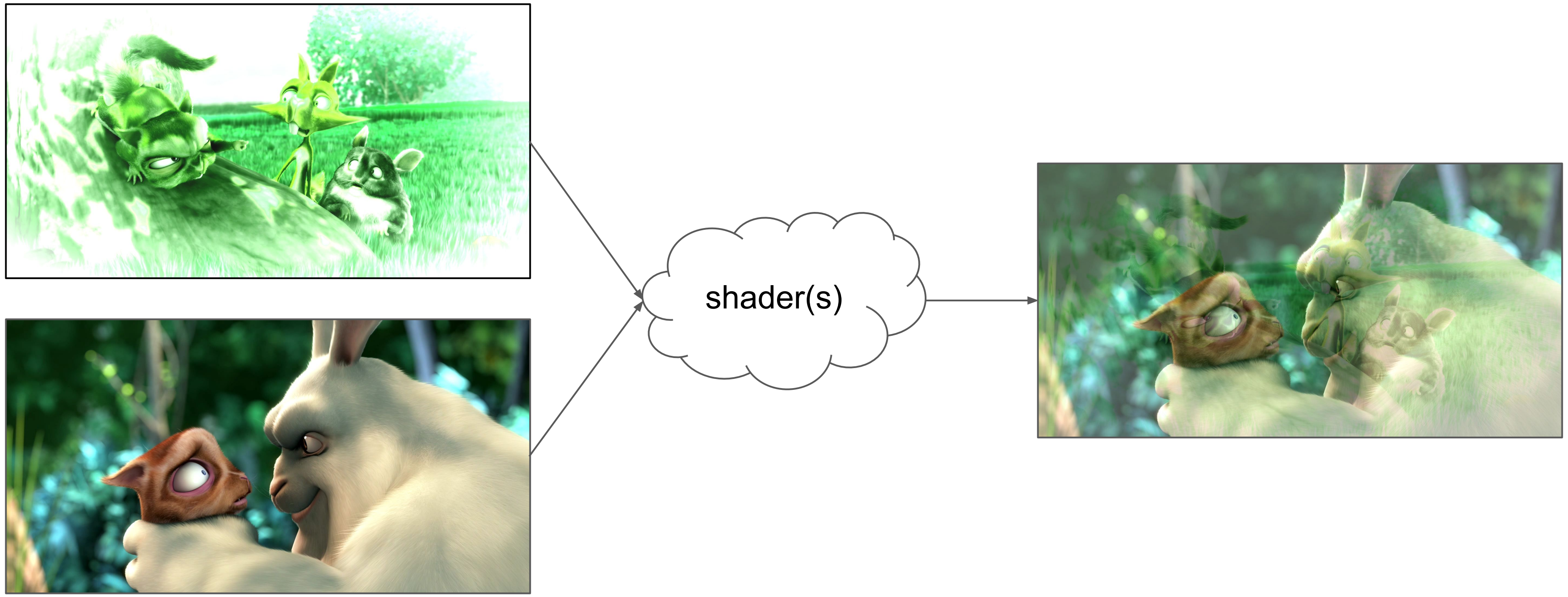


- timestamp
- duration

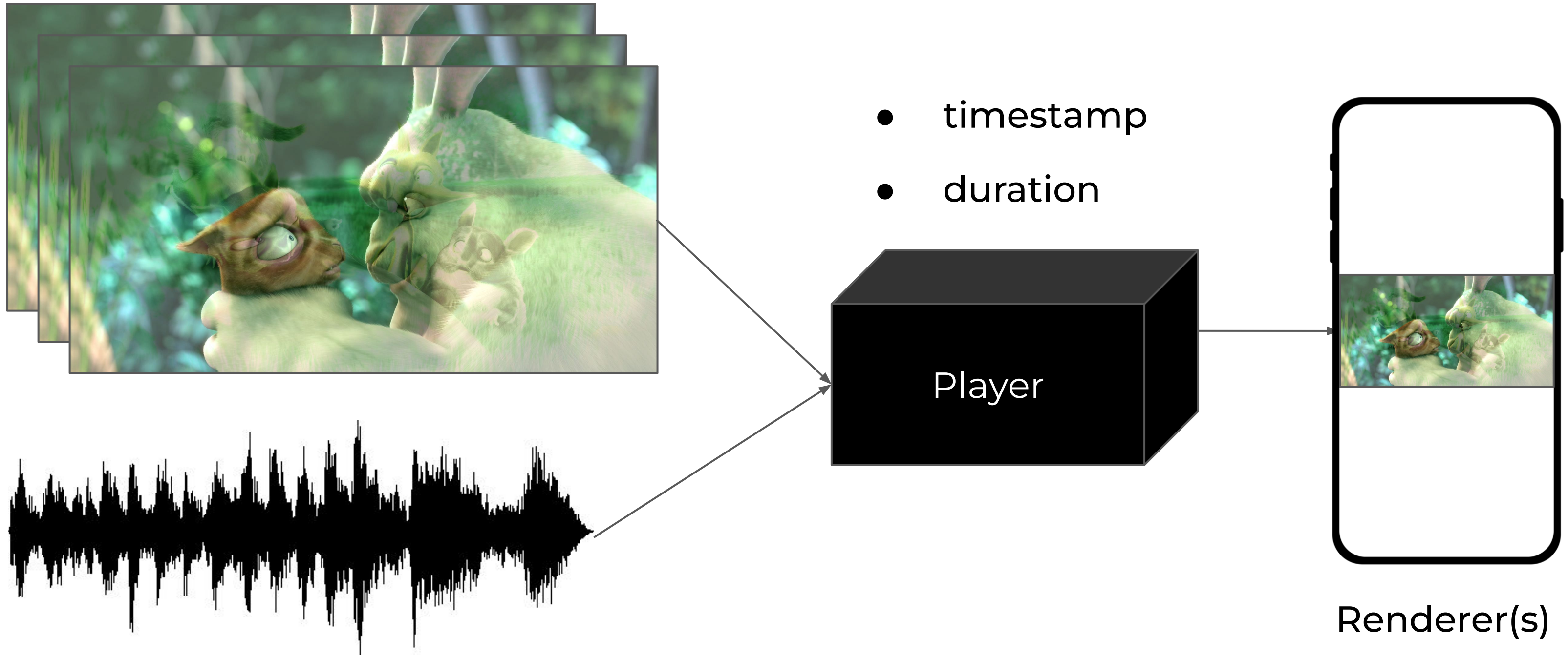
Обработка мультимедиа Pipeline



Обработка мультимедиа Pipeline



Обработка мультимедиа Pipeline



Обработка мультимедиа Pipeline

HD, 1280 x 720 = 921600 px

32bit RGBA 🟥🟩🟦▨

≈ 3,5 МБ (1 кадр)

1 сек / 25 fps = 40 мсек / кадр

Обработка мультимедиа Выводы (общие)

1. **Контракты** должны быть общими:
   - parser
   - decoder
   - renderer
2. **Реализации** могут быть общими:
   - stream (video/audio)
   - player
   - modifier (effect/filter)
3. **Память**, расходуемая на работу с мультимедиа, **должна контролироваться**.

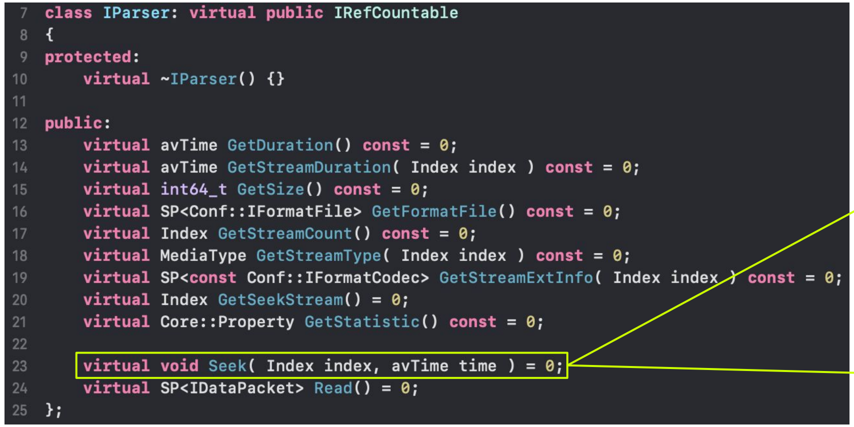
Обработка мультимедиа
Мультимедийное ядро на C++

Обработка мультимедиа Выводы (частные)

1. **Реализации** должны быть частными:
   - parser
   - decoder
   - renderer
2. **Работа с видео** должна происходить **на GPU** (по возможности).

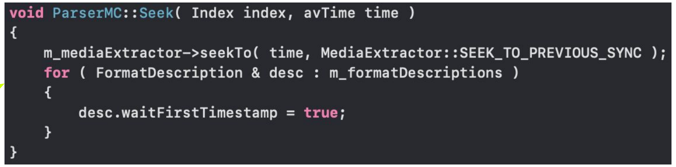
Обработка мультимедиа Мультимедийное ядро на C++

Контракт



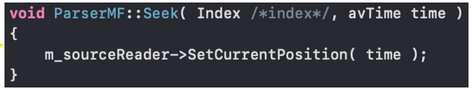
```cpp
class IParser: virtual public IRefCountable
{
protected:
    virtual ~IParser() {}

public:
    virtual avTime GetDuration() const = 0;
    virtual avTime GetStreamDuration( Index index ) const = 0;
    virtual int64_t GetSize() const = 0;
    virtual SP<Conf::IFormatFile> GetFormatFile() const = 0;
    virtual Index GetStreamCount() const = 0;
    virtual MediaType GetStreamType( Index index ) const = 0;
    virtual SP<const Conf::IFormatCodec> GetStreamExtInfo( Index index ) const = 0;
    virtual Index GetSeekStream() = 0;
    virtual Core::Property GetStatistic() const = 0;

    virtual void Seek( Index index, avTime time ) = 0;
    virtual SP<IDataPacket> Read() = 0;
};
```

Android реализация



```cpp
void ParserMC::Seek( Index index, avTime time )
{
    m_mediaExtractor->seekTo( time, MediaExtractor::SEEK_TO_PREVIOUS_SYNC );
    for ( FormatDescription & desc : m_formatDescriptions )
    {
        desc.waitFirstTimestamp = true;
    }
}
```

Windows реализация



```cpp
void ParserMF::Seek( Index /*index*/, avTime time )
{
    m_sourceReader->SetCurrentPosition( time );
}
```

Обработка мультимедиа Мультимедийное ядро на C++ Технологии

MediaFoundation

CUDA

VideoToolbox

NDK

OpenGL

Metal

MediaExtractor

AVFramework

MediaCodec

AudioToolbox

Обработка мультимедиа
Мультимедийное ядро на С++
Технологии

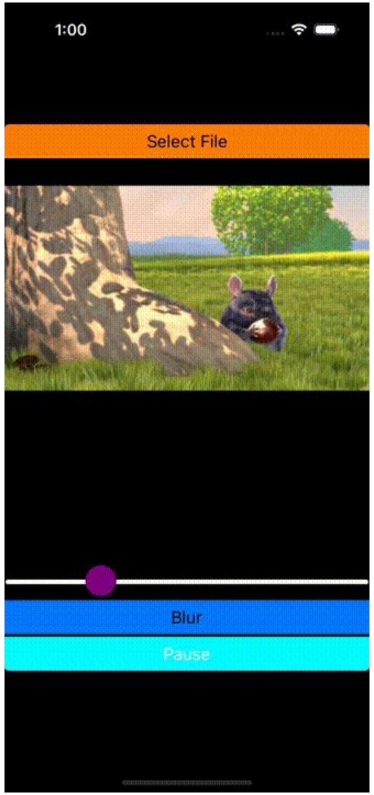
iOS

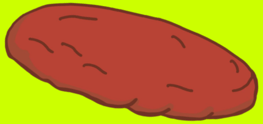
Обработка мультимедиа Портирование на iOS

1. **Реализации** должны быть частными:
    - decoder - Video Toolbox
    - renderer - CAEAGLLayer
2. **Работа с видео** должна происходить **на GPU** - OpenGL + Metal

Обработка мультимедиа Портирование на iOS Демо-приложение

- Проверка работоспособности.
- Замеры производительности.
- Упрощённая интеграция и отладка эффектов.



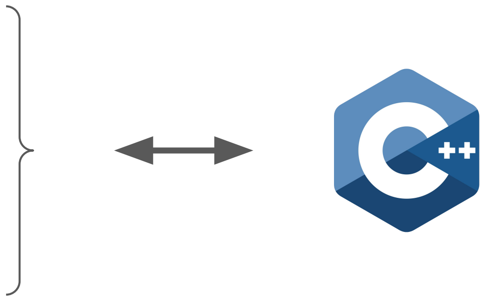


Биндинги между C++ и Kotlin

Биндинги для Kotlin/Native

Kotlin/Native supports the following platforms:

- macOS
- iOS, tvOS, watchOS
- Linux
- Windows (MinGW)
- Android NDK



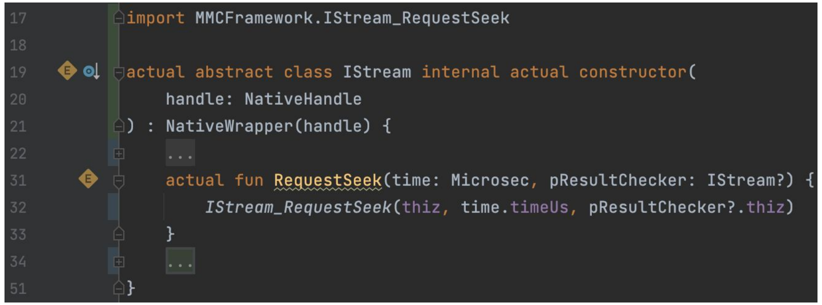
Биндинги для Kotlin/Native (iOS)

Interoperability:

- with C
- with Swift/Objective-C

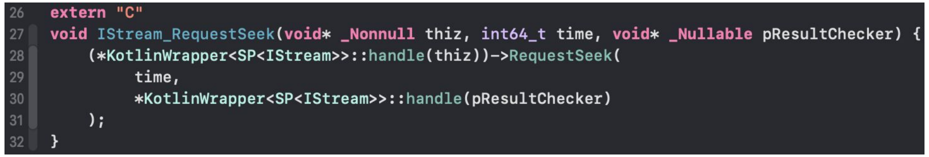
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

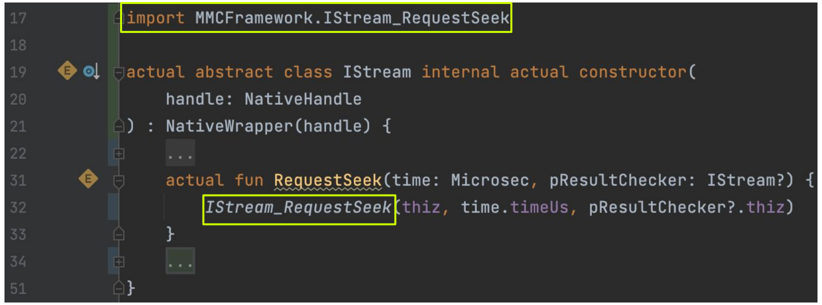
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
        time,
        *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

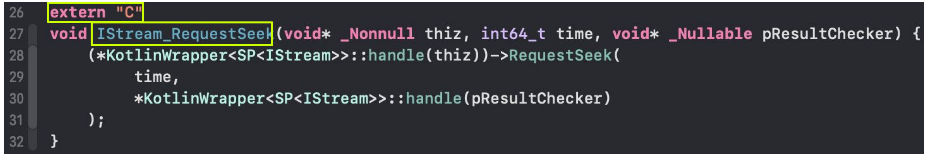
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

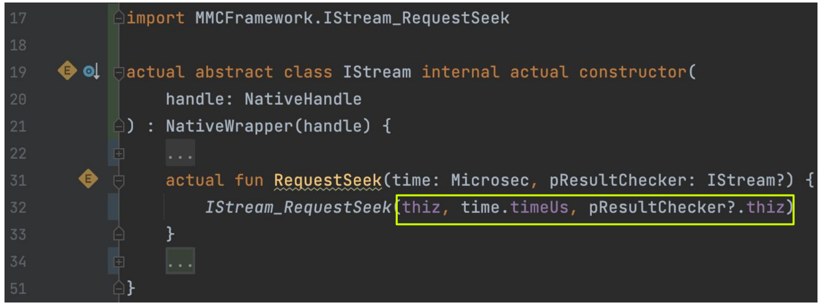
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
        time,
        *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

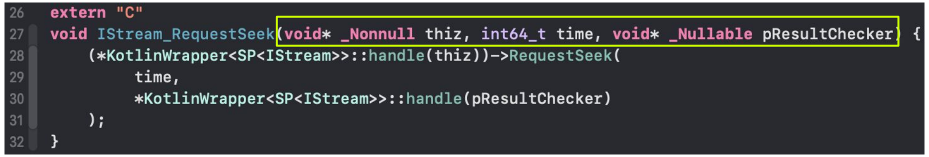
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

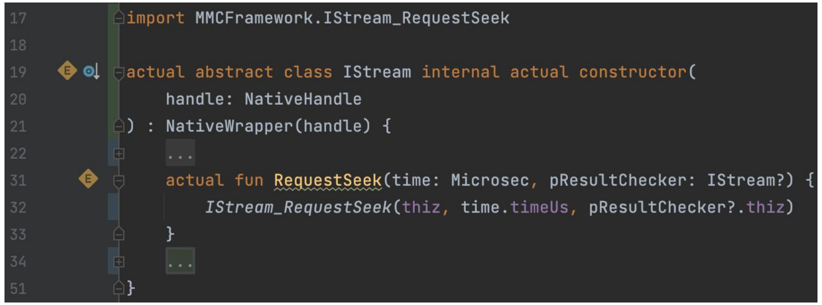
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
        time,
        *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

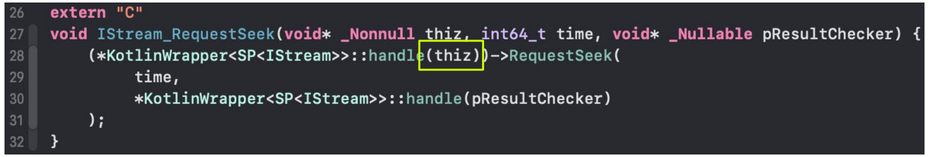
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

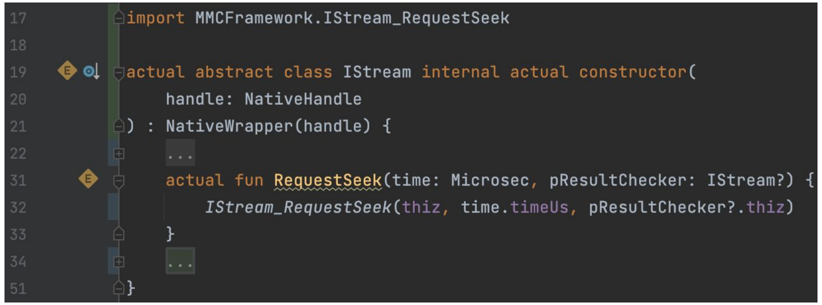
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
        time,
        *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

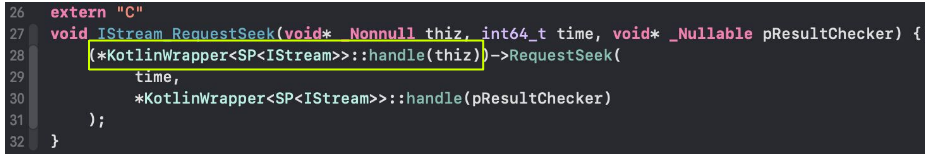
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

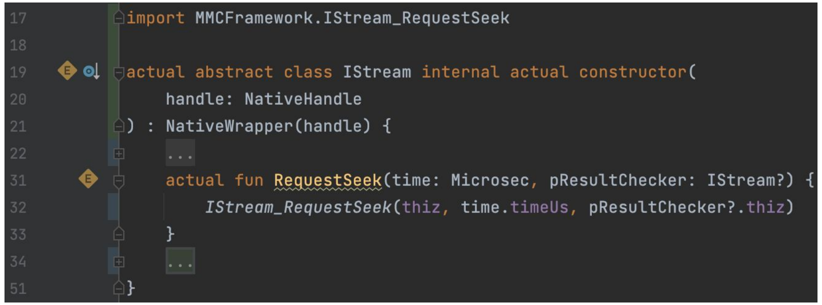
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
        time,
        *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

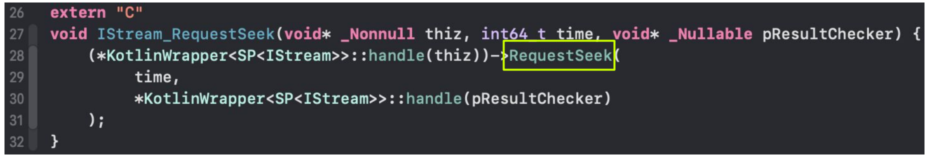
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

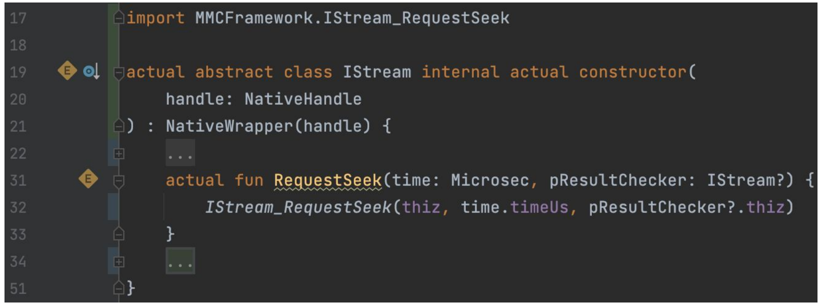
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
        time,
        *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

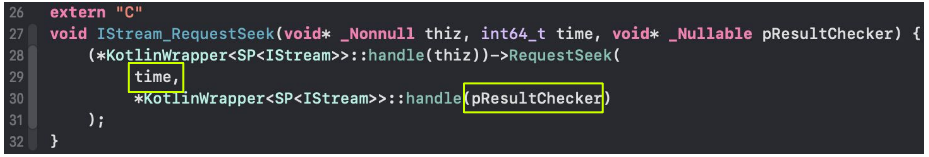
Биндинги для Kotlin/Native (iOS) Interoperability with C

Kotlin



```kotlin
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

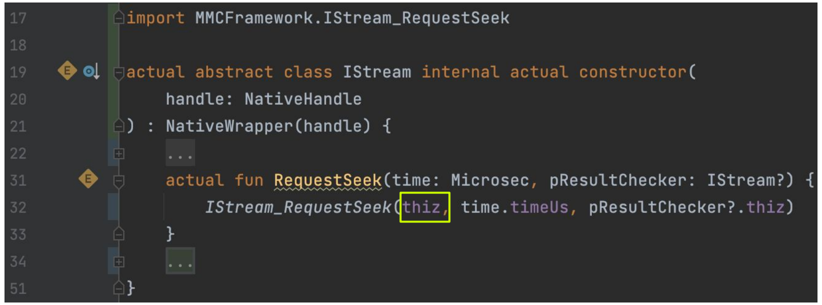
C++



```cpp
extern "C"
void IStream_RequestSeek(void* _Nonnull thiz, int64_t time, void* _Nullable pResultChecker) {
    (*KotlinWrapper<SP<IStream>>::handle(thiz))->RequestSeek(
            time,
            *KotlinWrapper<SP<IStream>>::handle(pResultChecker)
    );
}
```

Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

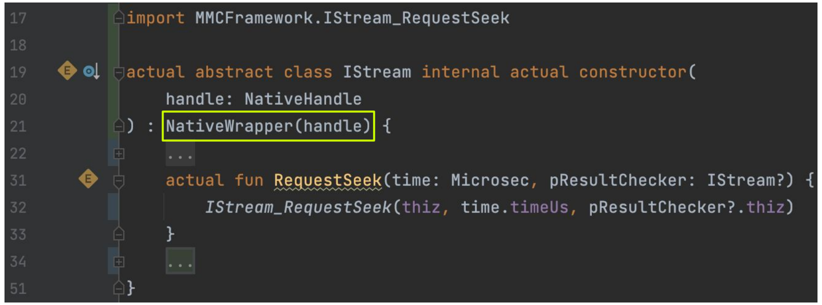
Kotlin



```
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

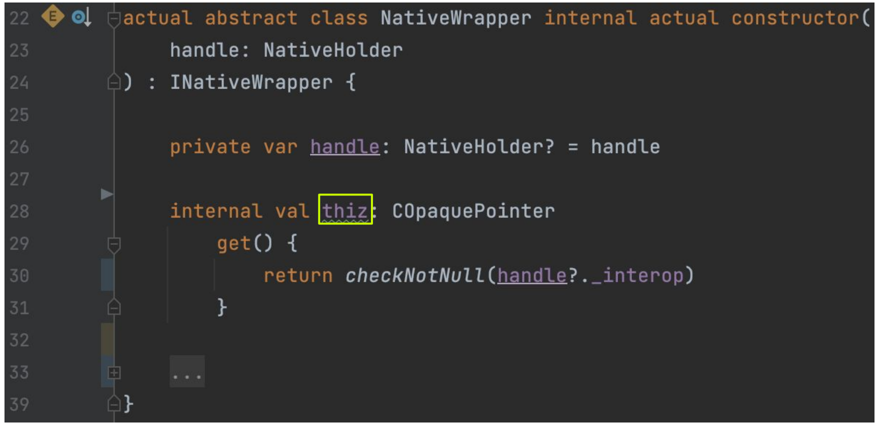
Kotlin



```
import MMCFramework.IStream_RequestSeek

actual abstract class IStream internal actual constructor(
    handle: NativeHandle
) : NativeWrapper(handle) {
    ...
    actual fun RequestSeek(time: Microsec, pResultChecker: IStream?) {
        IStream_RequestSeek(thiz, time.timeUs, pResultChecker?.thiz)
    }
    ...
}
```

Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

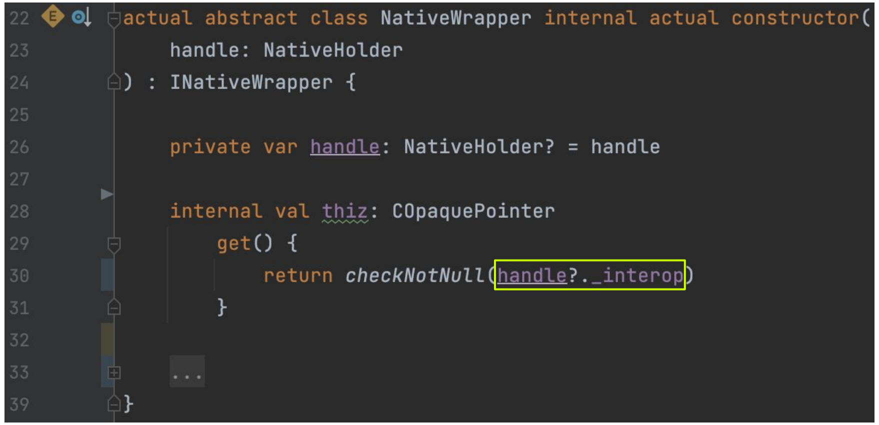
Kotlin



```
actual abstract class NativeWrapper internal actual constructor(
    handle: NativeHolder
) : INativeWrapper {

    private var handle: NativeHolder? = handle

    internal val thiz: COpaquePointer
        get() {
            return checkNotNull(handle?._interop)
        }

    ...
}
```

Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

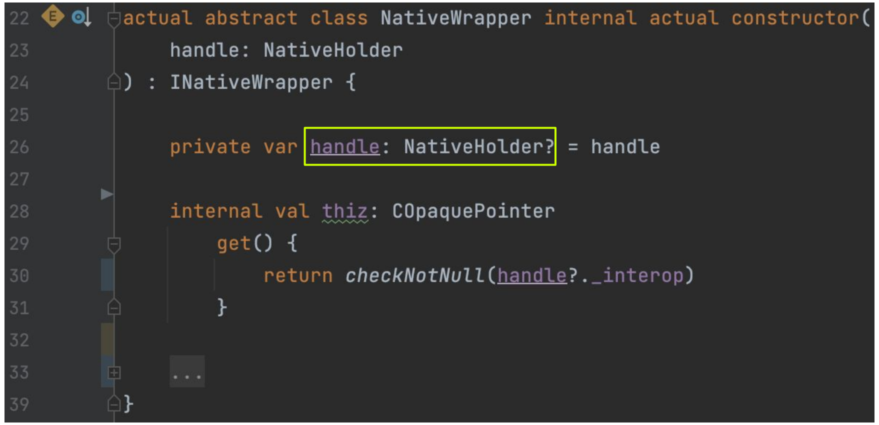
Kotlin



```
actual abstract class NativeWrapper internal actual constructor(
        handle: NativeHolder
) : INativeWrapper {

        private var handle: NativeHolder? = handle

        internal val thiz: COpaquePointer
            get() {
                return checkNotNull(handle?._interop)
            }

        ...
}
```

Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

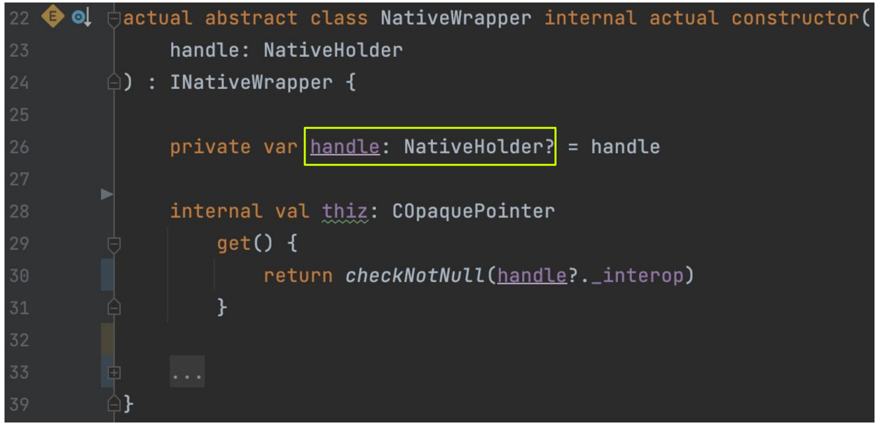
Kotlin



```
actual abstract class NativeWrapper internal actual constructor(
    handle: NativeHolder
) : INativeWrapper {

    private var handle: NativeHolder? = handle

    internal val thiz: COpaquePointer
        get() {
            return checkNotNull(handle?._interop)
        }

    ...
}
```

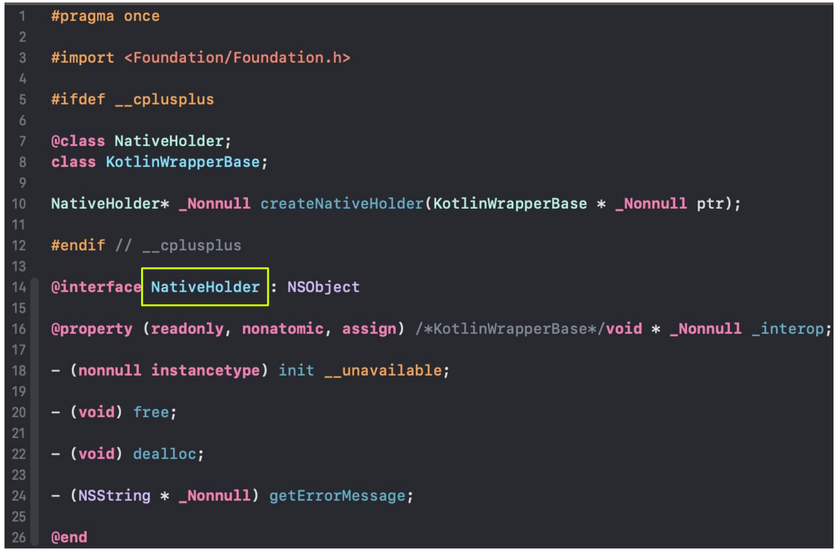
Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

Kotlin



```kotlin
actual abstract class NativeWrapper internal actual constructor(
    handle: NativeHolder
) : INativeWrapper {

    private var handle: NativeHolder? = handle

    internal val thiz: COpaquePointer
        get() {
            return checkNotNull(handle?._interop)
        }

    ...
}
```

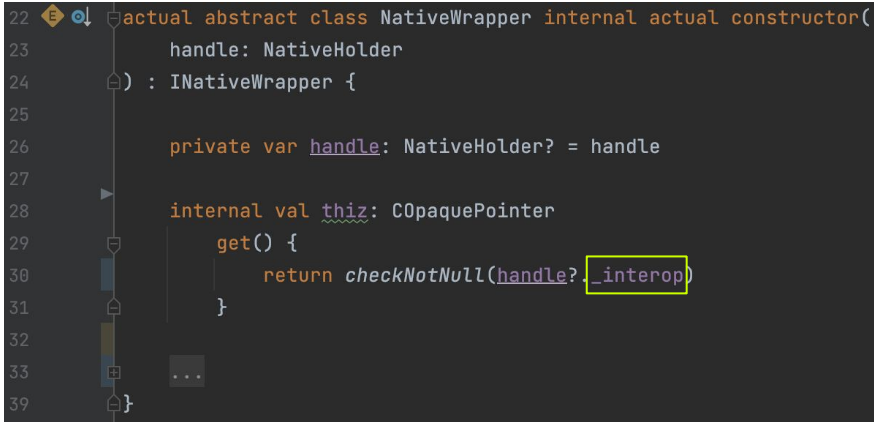
Objective-C



```objc
#pragma once

#import <Foundation/Foundation.h>

#ifdef __cplusplus

@class NativeHolder;
class KotlinWrapperBase;

NativeHolder* _Nonnull createNativeHolder(KotlinWrapperBase * _Nonnull ptr);

#endif // __cplusplus

@interface NativeHolder : NSObject

@property (readonly, nonatomic, assign) /*KotlinWrapperBase*/void * _Nonnull _interop;

- (nonnull instancetype) init __unavailable;

- (void) free;

- (void) dealloc;

- (NSString * _Nonnull) getErrorMessage;

@end
```

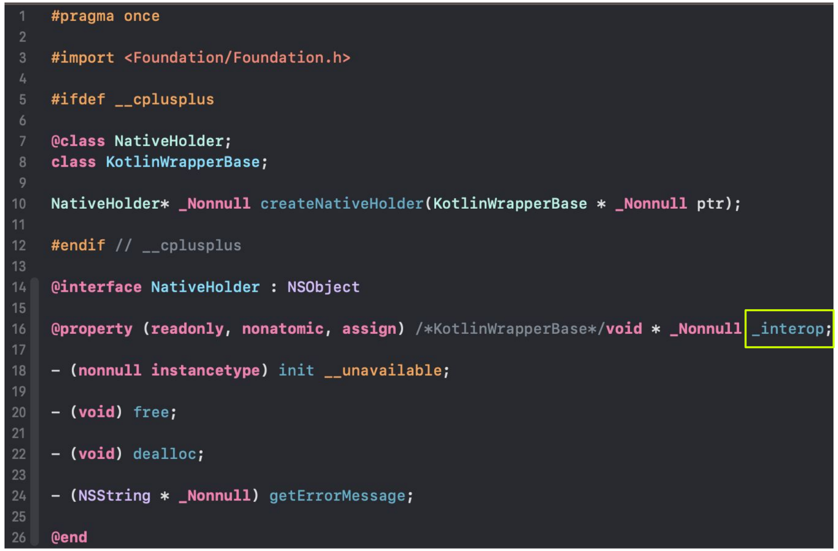
Биндинги для Kotlin/Native (iOS) Interoperability with Objective-C

Kotlin



```kotlin
actual abstract class NativeWrapper internal actual constructor(
    handle: NativeHolder
) : INativeWrapper {

    private var handle: NativeHolder? = handle

    internal val thiz: COpaquePointer
        get() {
            return checkNotNull(handle?._interop)
        }

    ...
}
```

Objective-C

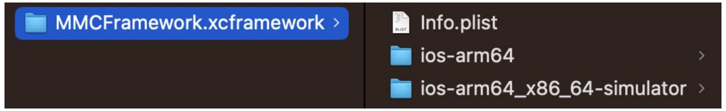


```objc
#pragma once

#import <Foundation/Foundation.h>

#ifdef __cplusplus

@class NativeHolder;
class KotlinWrapperBase;

NativeHolder* _Nonnull createNativeHolder(KotlinWrapperBase * _Nonnull ptr);

#endif // __cplusplus

@interface NativeHolder : NSObject

@property (readonly, nonatomic, assign) /*KotlinWrapperBase*/void * _Nonnull _interop;

- (nonnull instancetype) init __unavailable;

- (void) free;

- (void) dealloc;

- (NSString * _Nonnull) getErrorMessage;

@end
```

Биндинги для Kotlin/Native (iOS)

Dependency delivery:

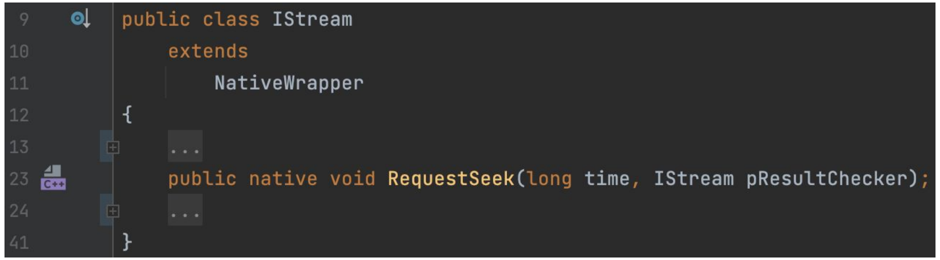
- CocoaPods
- .def files



```
MMCFramework.xcframework >   Info.plist
                             ios-arm64 >
                             ios-arm64_x86_64-simulator >
```

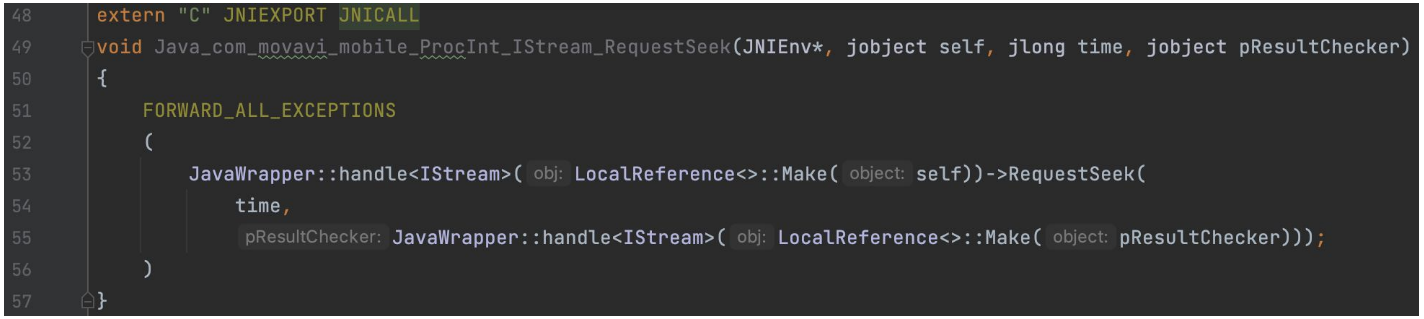
Биндинги для Kotlin, Java (Android) Interoperability with C

Java



```java
public class IStream
    extends
        NativeWrapper
{
    ...
    public native void RequestSeek(long time, IStream pResultChecker);
    ...
}
```

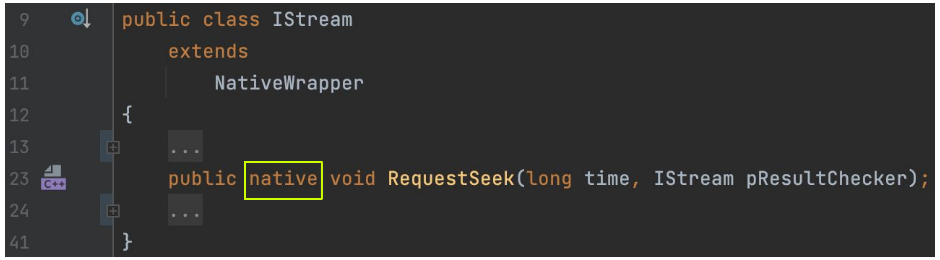
C++



```cpp
extern "C" JNIEXPORT JNICALL
void Java_com_movavi_mobile_ProcInt_IStream_RequestSeek(JNIEnv*, jobject self, jlong time, jobject pResultChecker)
{
    FORWARD_ALL_EXCEPTIONS
    (
        JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: self))->RequestSeek(
            time,
            pResultChecker: JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: pResultChecker)));
    )
}
```

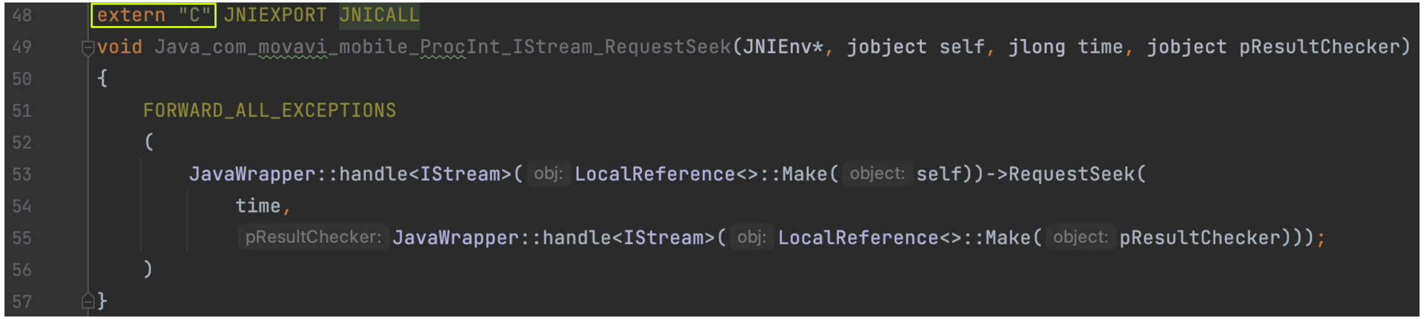
Биндинги для Kotlin, Java (Android) Interoperability with C

Java



```java
public class IStream
    extends
        NativeWrapper
{
    ...
    public native void RequestSeek(long time, IStream pResultChecker);
    ...
}
```

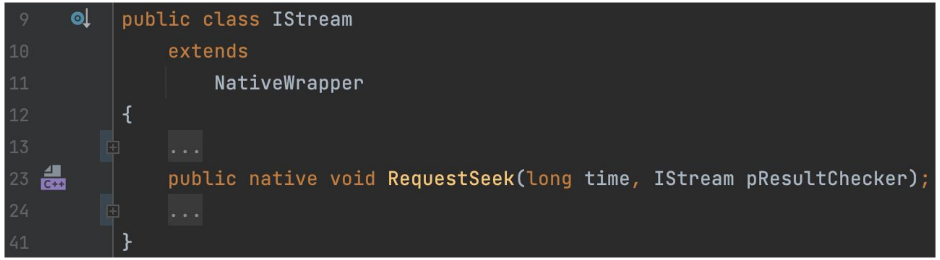
C++



```cpp
extern "C" JNIEXPORT JNICALL
void Java_com_movavi_mobile_ProcInt_IStream_RequestSeek(JNIEnv*, jobject self, jlong time, jobject pResultChecker)
{
    FORWARD_ALL_EXCEPTIONS
    (
        JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: self))->RequestSeek(
            time,
            pResultChecker: JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: pResultChecker)));
    )
}
```

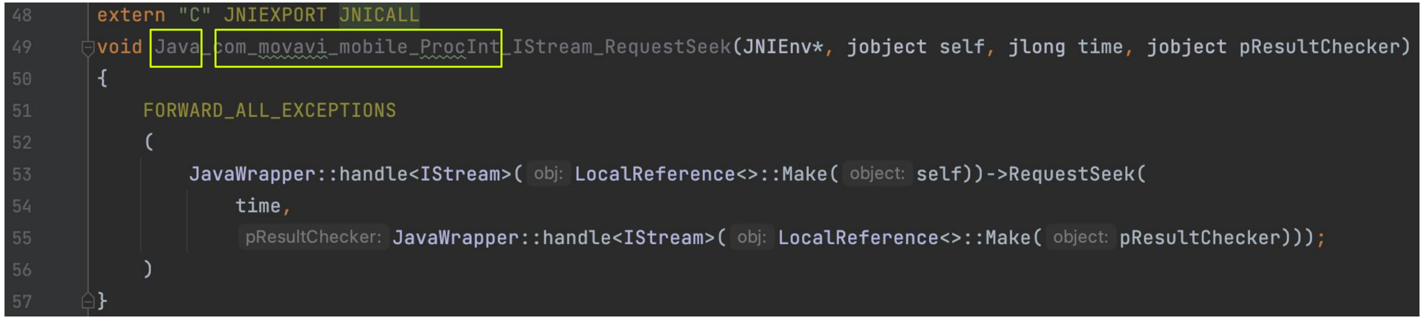
Биндинги для Kotlin, Java (Android) Interoperability with C

Java



```java
public class IStream
    extends
        NativeWrapper
{
    ...
    public native void RequestSeek(long time, IStream pResultChecker);
    ...
}
```

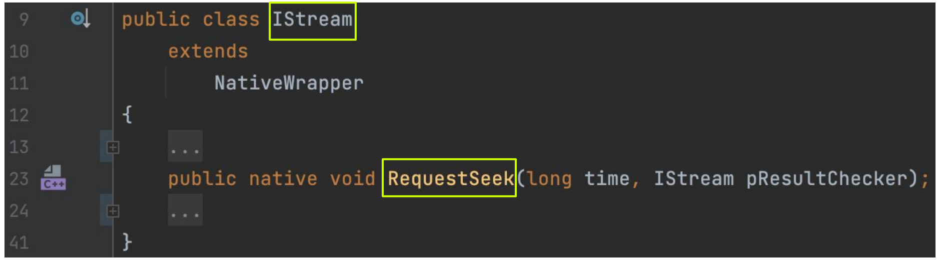
C++



```cpp
extern "C" JNIEXPORT JNICALL
void Java_com_movavi_mobile_ProcInt_IStream_RequestSeek(JNIEnv*, jobject self, jlong time, jobject pResultChecker)
{
    FORWARD_ALL_EXCEPTIONS
    (
        JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: self))->RequestSeek(
            time,
            pResultChecker: JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: pResultChecker)));
    )
}
```

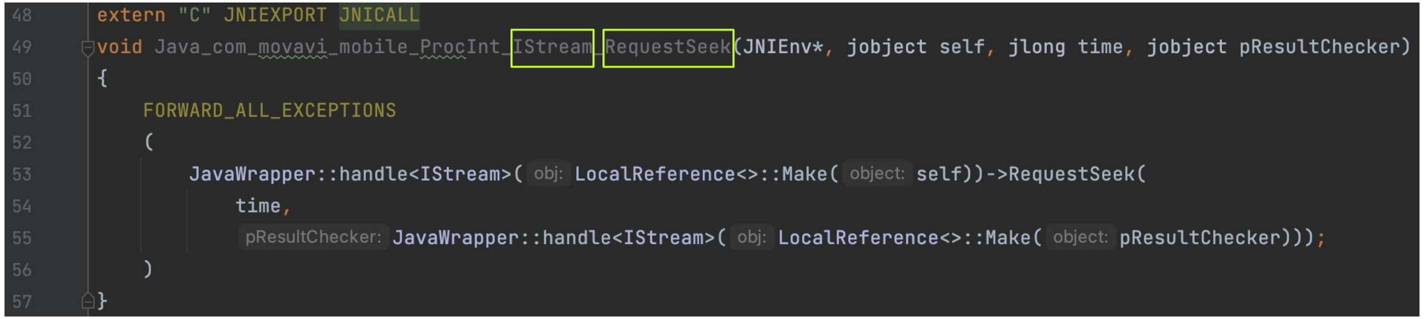
Биндинги для Kotlin, Java (Android) Interoperability with C

Java



```java
public class IStream
    extends
        NativeWrapper
{
    ...
    public native void RequestSeek(long time, IStream pResultChecker);
    ...
}
```

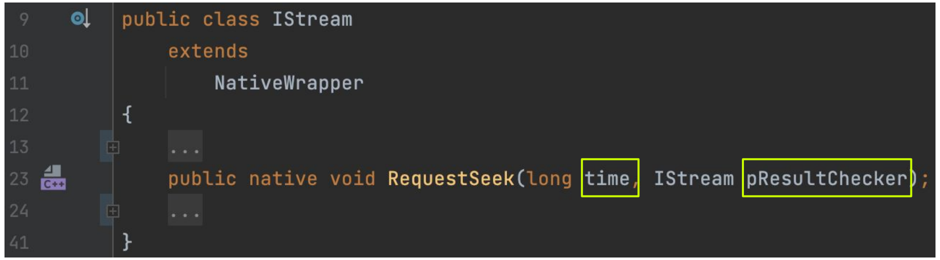
C++



```cpp
extern "C" JNIEXPORT JNICALL
void Java_com_movavi_mobile_ProcInt_IStream_RequestSeek(JNIEnv*, jobject self, jlong time, jobject pResultChecker)
{
    FORWARD_ALL_EXCEPTIONS
    (
        JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: self))->RequestSeek(
            time,
            pResultChecker: JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: pResultChecker)));
    )
}
```

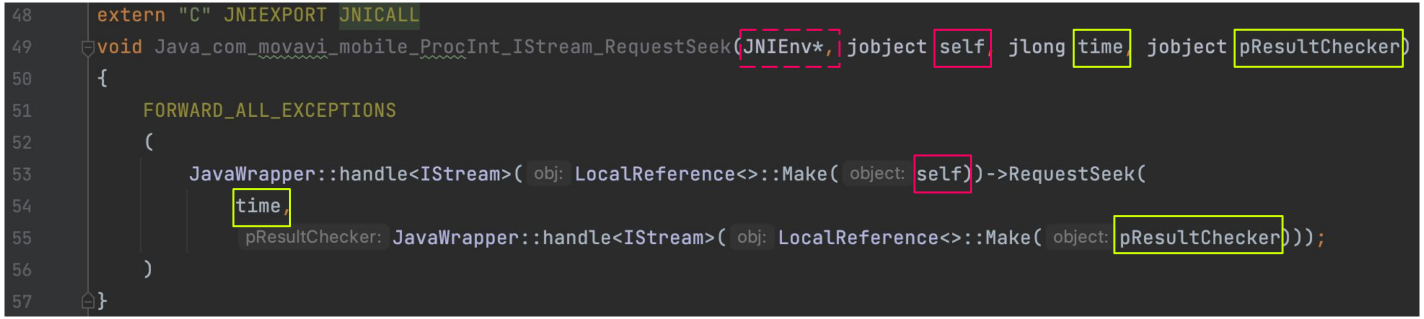
Биндинги для Kotlin, Java (Android) Interoperability with C

Java



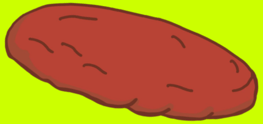
```java
public class IStream
    extends
        NativeWrapper
{
    ...
    public native void RequestSeek(long time, IStream pResultChecker);
    ...
}
```

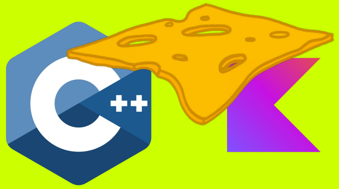
C++



```cpp
extern "C" JNIEXPORT JNICALL
void Java_com_movavi_mobile_ProcInt_IStream_RequestSeek(JNIEnv*, jobject self, jlong time, jobject pResultChecker)
{
    FORWARD_ALL_EXCEPTIONS
    (
        JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: self))->RequestSeek(
            time,
            pResultChecker: JavaWrapper::handle<IStream>( obj: LocalReference<>::Make( object: pResultChecker)));
    )
}
```

Управление памятью

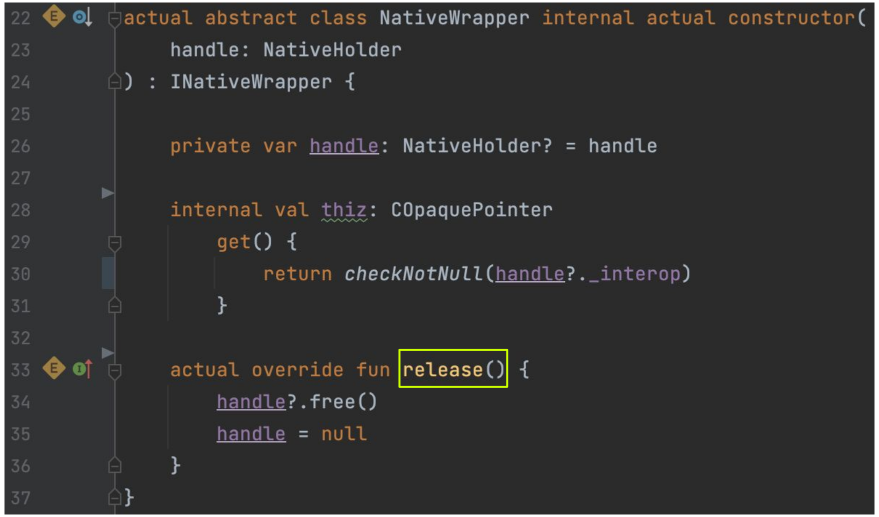




Kotlin

Биндинги для Kotlin/Native (iOS) Управление памятью

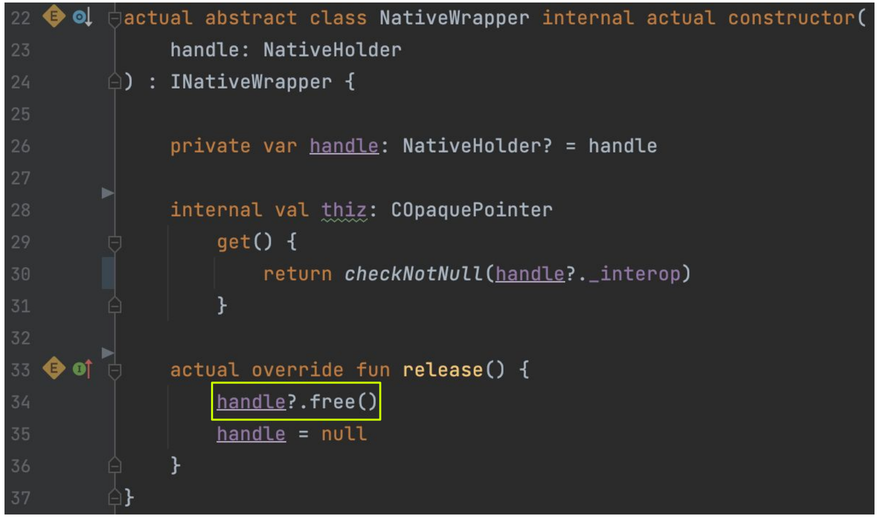
Kotlin



```kotlin
actual abstract class NativeWrapper internal actual constructor(
    handle: NativeHolder
) : INativeWrapper {

    private var handle: NativeHolder? = handle

    internal val thiz: COpaquePointer
        get() {
            return checkNotNull(handle?._interop)
        }

    actual override fun release() {
        handle?.free()
        handle = null
    }
}
```

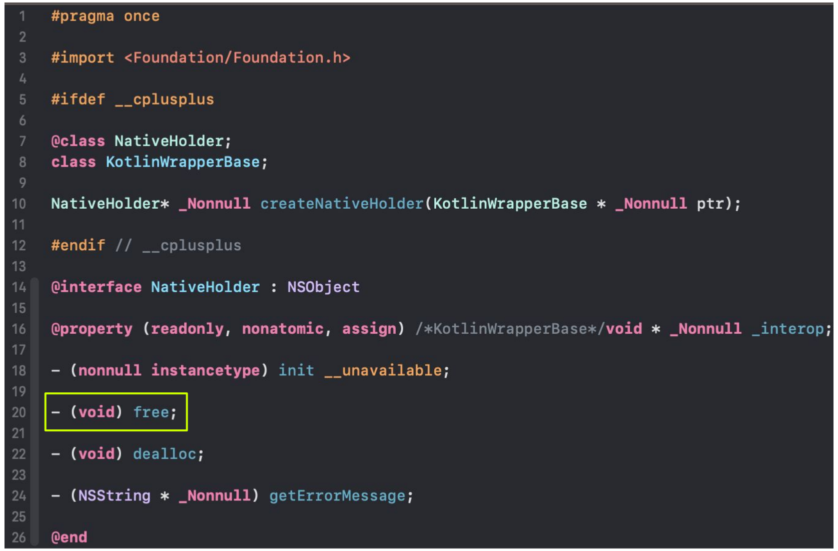
Биндинги для Kotlin/Native (iOS) Управление памятью

Kotlin



```kotlin
actual abstract class NativeWrapper internal actual constructor(
        handle: NativeHolder
) : INativeWrapper {

        private var handle: NativeHolder? = handle

        internal val thiz: COpaquePointer
            get() {
                return checkNotNull(handle?._interop)
            }

        actual override fun release() {
            handle?.free()
            handle = null
        }
}
```

Objective-C



```objc
#pragma once

#import <Foundation/Foundation.h>

#ifdef __cplusplus

@class NativeHolder;
class KotlinWrapperBase;

NativeHolder* _Nonnull createNativeHolder(KotlinWrapperBase * _Nonnull ptr);

#endif // __cplusplus

@interface NativeHolder : NSObject

@property (readonly, nonatomic, assign) /*KotlinWrapperBase*/void * _Nonnull _interop;

- (nonnull instancetype) init __unavailable;

- (void) free;

- (void) dealloc;

- (NSString * _Nonnull) getErrorMessage;

@end
```

Биндинги для Kotlin/Native Управление памятью

C++
manual smart-pointers
destructors

Биндинги для Kotlin/Native
Управление памятью

C++	Objective-C
manual smart-pointers	ARC MRC
destructors	dealloc

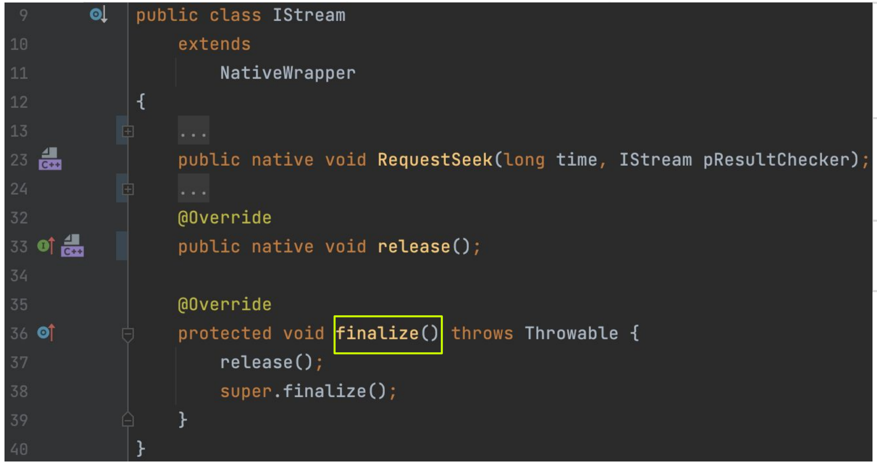
Биндинги для Kotlin/Native
Управление памятью

C++	Objective-C	Swift
manual smart-pointers	ARC MRC	ARC
destructors	dealloc	deinit

Биндинги для Kotlin/Native
Управление памятью

C++	Objective-C	Swift	Java, Kotlin/JVM
manual smart-pointers	ARC MRC	ARC	GC
destructors	dealloc	deinit	finalize

Биндинги для Kotlin/Native
Управление памятью



```
public class IStream
    extends
        NativeWrapper
{
    ...
    public native void RequestSeek(long time, IStream pResultChecker);
    ...
    @Override
    public native void release();

    @Override
    protected void finalize() throws Throwable {
        release();
        super.finalize();
    }
}
```

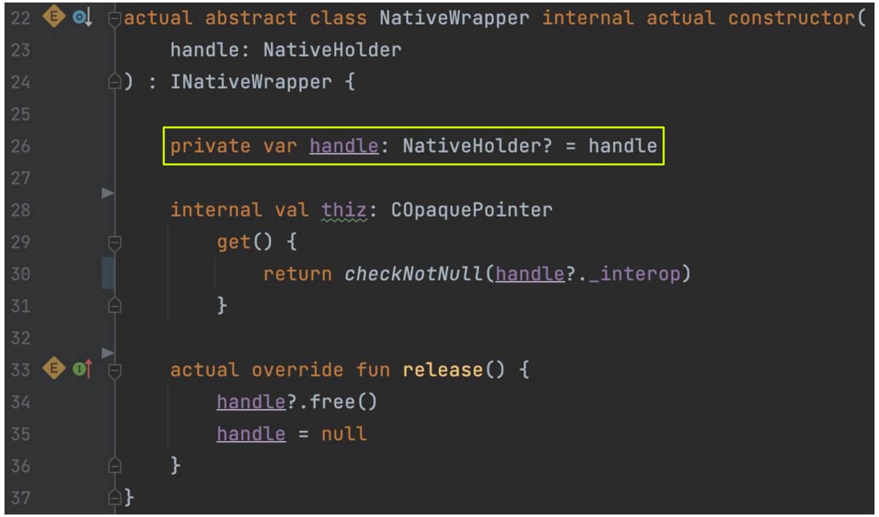
Java, Kotlin/JVM
GC
finalize

Биндинги для Kotlin/Native
Управление памятью

C++	Objective-C	Swift	Java, Kotlin/JVM	Kotlin/Native
manual smart-pointers	ARC MRC	ARC	GC	GC
destructors	dealloc	deinit	finalize	

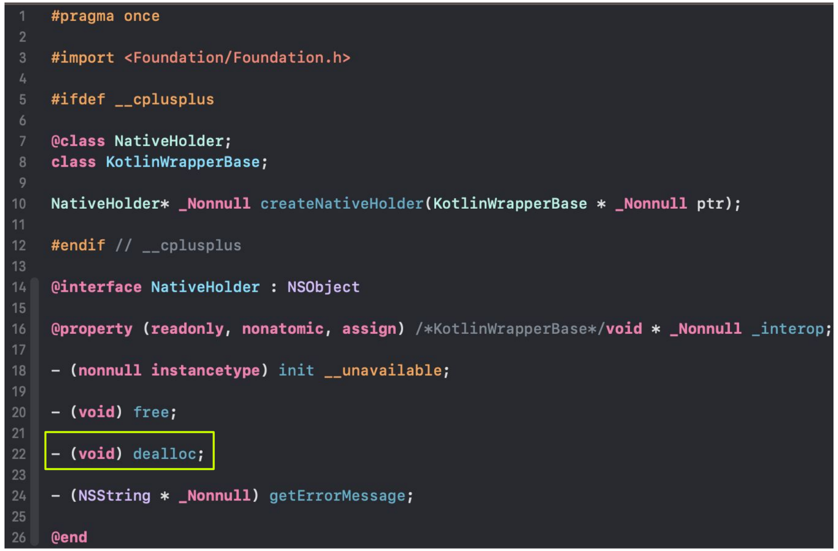
Биндинги для Kotlin/Native (iOS) Управление памятью

Kotlin



```kotlin
actual abstract class NativeWrapper internal actual constructor(
    handle: NativeHolder
) : INativeWrapper {

    private var handle: NativeHolder? = handle

    internal val thiz: COpaquePointer
        get() {
            return checkNotNull(handle?._interop)
        }

    actual override fun release() {
        handle?.free()
        handle = null
    }
}
```

Objective-C



```objectivec
#pragma once

#import <Foundation/Foundation.h>

#ifdef __cplusplus

@class NativeHolder;
class KotlinWrapperBase;

NativeHolder* _Nonnull createNativeHolder(KotlinWrapperBase * _Nonnull ptr);

#endif // __cplusplus

@interface NativeHolder : NSObject

@property (readonly, nonatomic, assign) /*KotlinWrapperBase*/void * _Nonnull _interop;

- (nonnull instancetype) init __unavailable;

- (void) free;

- (void) dealloc;

- (NSString * _Nonnull) getErrorMessage;

@end
```

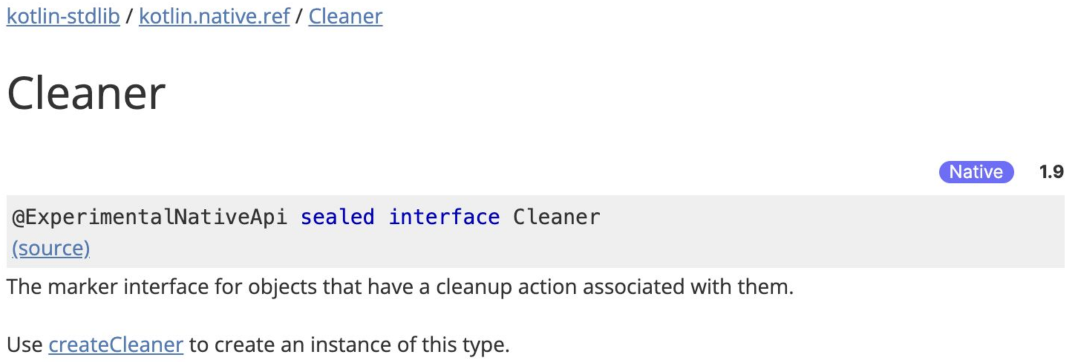
Биндинги для Kotlin/Native
Управление памятью

C++	Objective-C	Swift	Java, Kotlin/JVM	Kotlin/Native
manual smart-pointers	ARC MRC	ARC	GC	GC
destructors	dealloc	deinit	finalize	*

* < Kotlin 1.9

Биндинги для Kotlin/Native
Управление памятью

C++	Objective-C	Swift	Java, Kotlin/JVM	Kotlin/Native
manual smart-pointers	ARC MRC	ARC	GC	GC
destructors	dealloc	deinit	finalize	Cleaner



kotlin-stdlib / kotlin.native.ref / Cleaner

Cleaner

Native 1.9

```
@ExperimentalNativeApi sealed interface Cleaner
```
(source)

The marker interface for objects that have a cleanup action associated with them.

Use createCleaner to create an instance of this type.

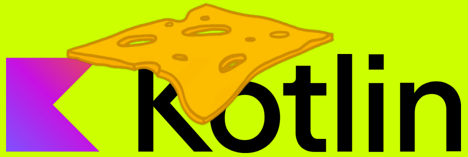
Биндинги для Kotlin/Native Управление памятью

C++	Objective-C	Swift	Java, Kotlin/JVM	Kotlin/Native
manual smart-pointers	ARC MRC	ARC	GC	GC
destructors	dealloc	deinit	~~finalize~~ Cleaner*	Cleaner

* Android: added in API level 33

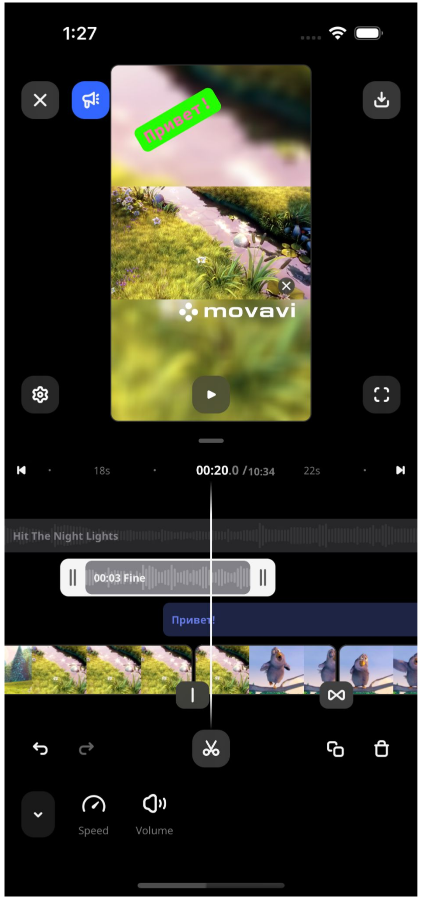
Java 9

Модель данных и бизнес-логика



Модель данных Timeline

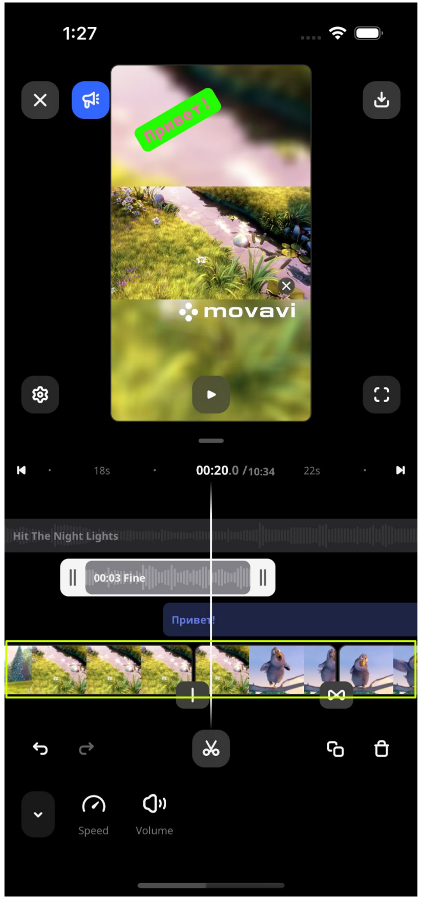
Основной экран редактирования видео



Модель данных Timeline

Основной экран редактирования видео:

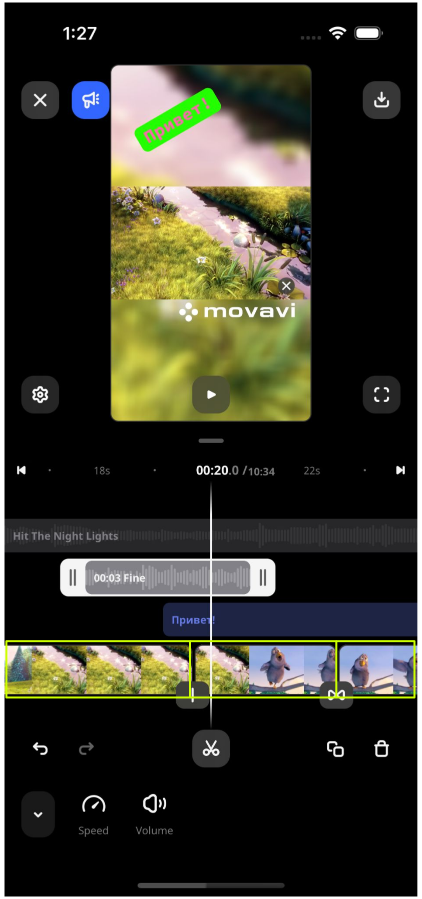
- основная видеодорожка



Модель данных Timeline

Основной экран редактирования видео:

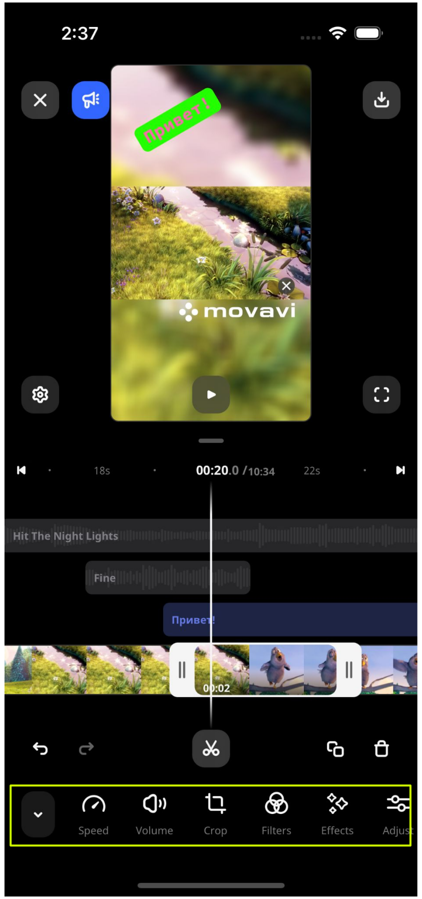
- основная видеодорожка
- элементы на основной видео дорожке



Модель данных Timeline

Основной экран редактирования видео:

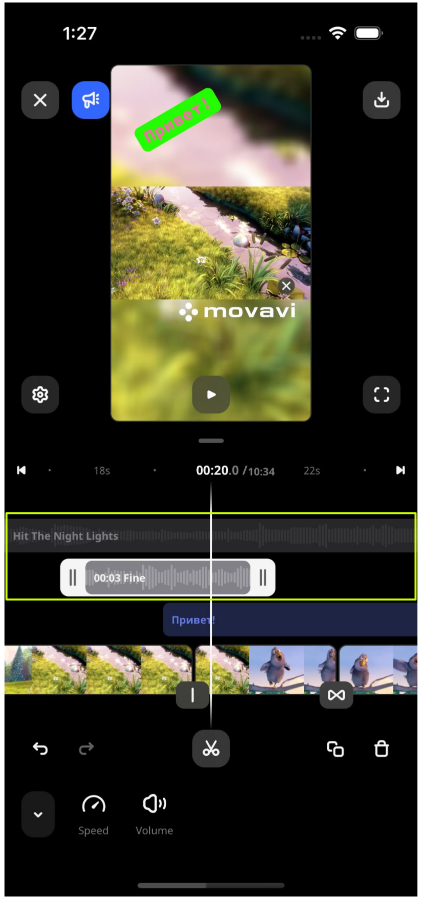
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)
- инструмент подрезки элемента дорожки
- инструмент разрезки
- настройки переходов между видео элементами
- инструменты настройки аудио элемента
- инструменты настройки видео элемента



Модель данных Timeline

Основной экран редактирования видео:

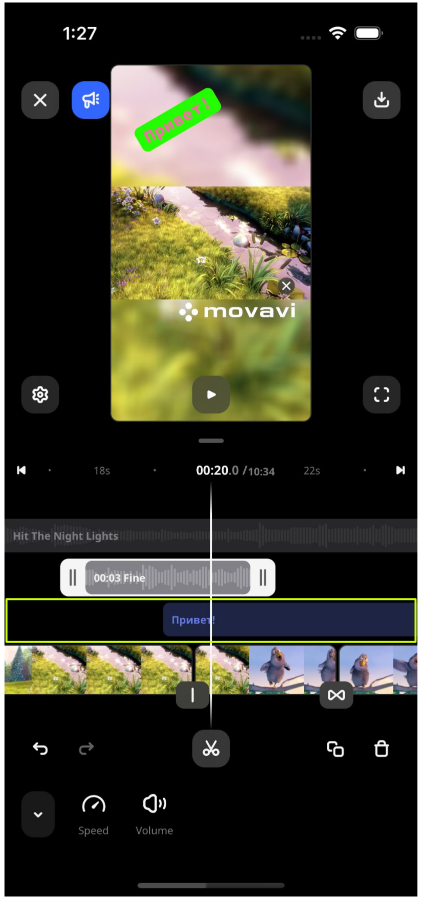
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки



Модель данных Timeline

Основной экран редактирования видео:

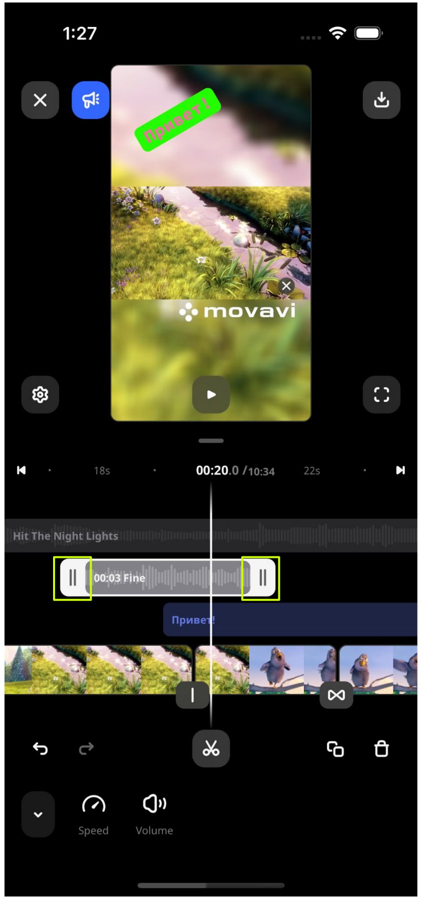
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)



Модель данных Timeline

Основной экран редактирования видео:

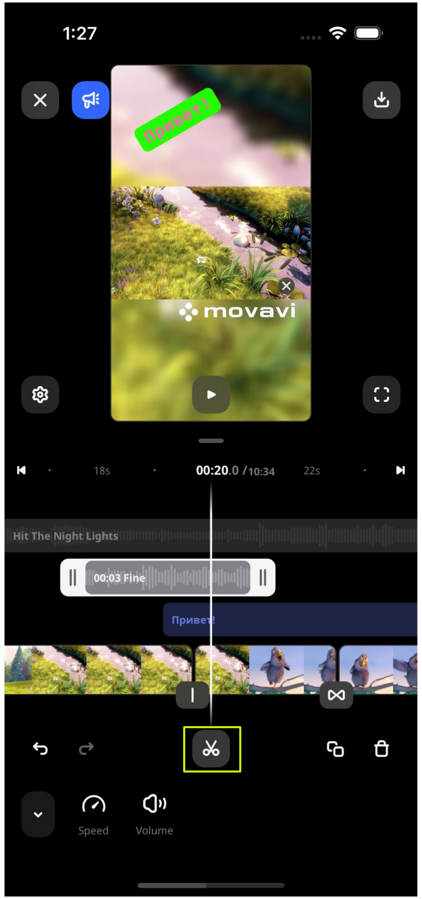
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)
- инструмент подрезки элемента дорожки



Модель данных Timeline

Основной экран редактирования видео:

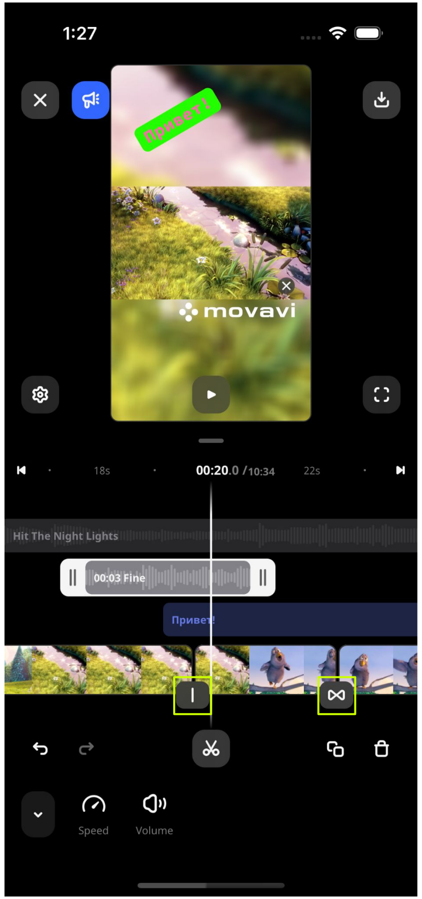
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)
- инструмент подрезки элемента дорожки
- инструмент разрезки



Модель данных Timeline

Основной экран редактирования видео:

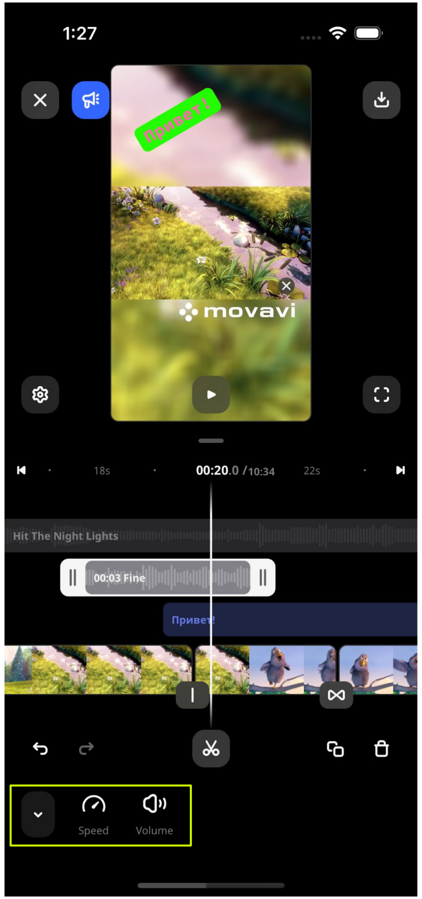
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)
- инструмент подрезки элемента дорожки
- инструмент разрезки
- настройки переходов между видео элементами



Модель данных Timeline

Основной экран редактирования видео:

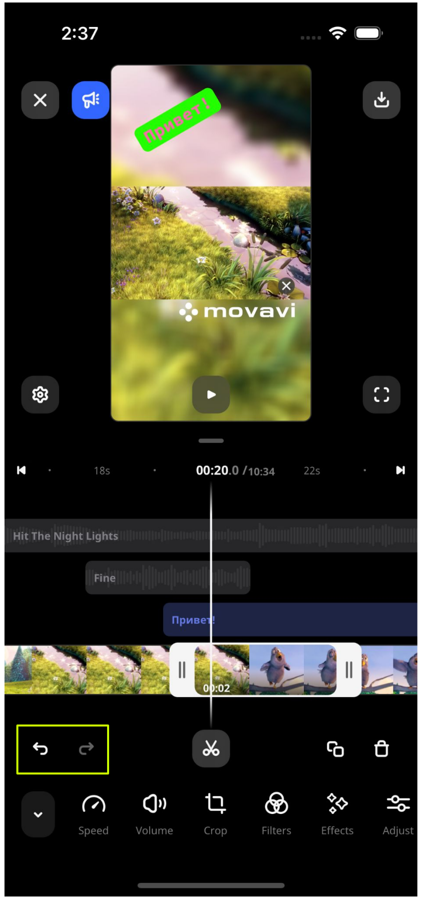
- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)
- инструмент подрезки элемента дорожки
- инструмент разрезки
- настройки переходов между видео элементами
- инструменты настройки аудио элемента



Модель данных Timeline

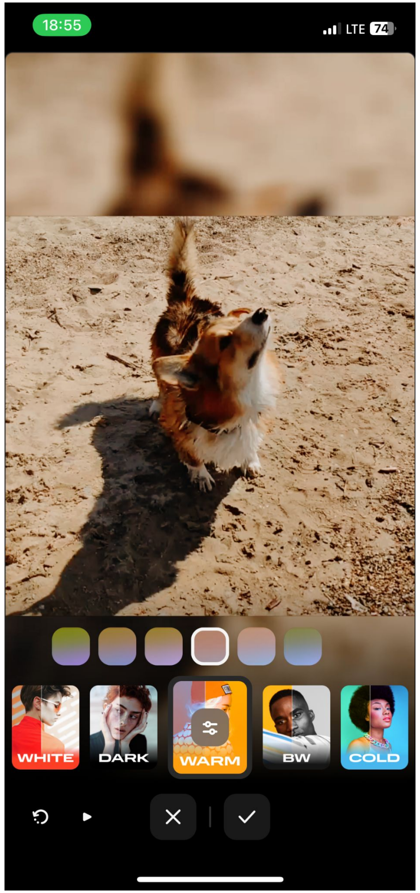
Основной экран редактирования видео:

- основная видеодорожка
- элементы на основной видео дорожке
- дополнительные музыкальные дорожки
- дополнительные дорожки наложения (текст)
- инструмент подрезки элемента дорожки
- инструмент разрезки
- настройки переходов между видео элементами
- инструменты настройки аудио элемента
- инструменты настройки видео элемента
- инструмент undo/redo



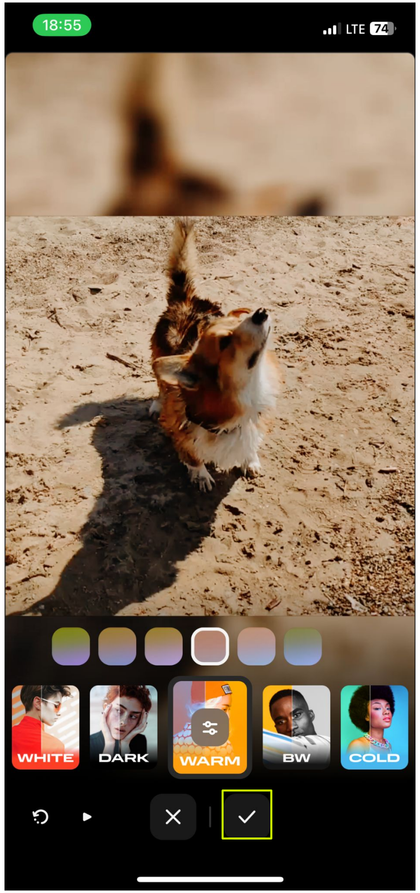
Изменения модели данных

- Coroutines
- Flow



Изменения модели данных

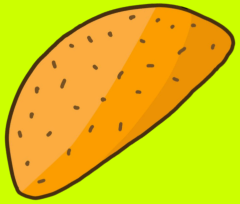
- Coroutines
- Flow

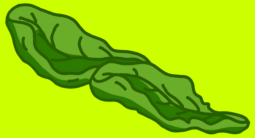


Third-party dependencies

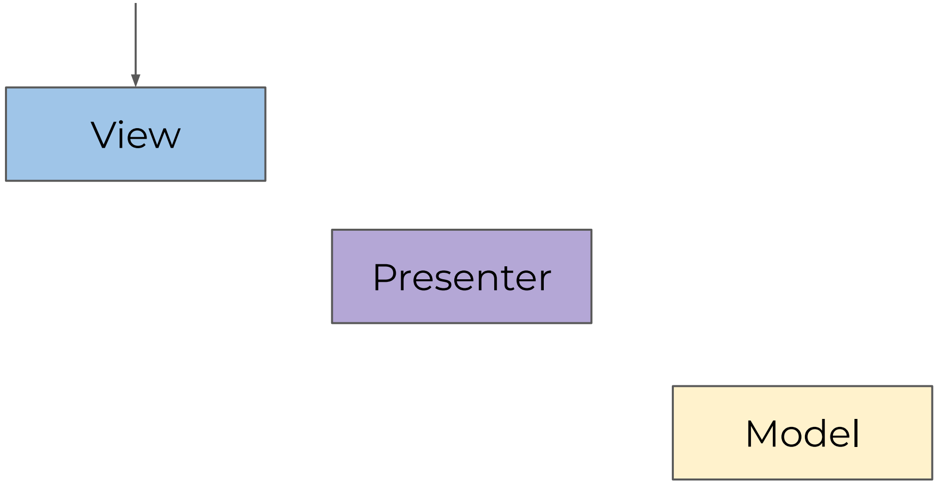
- A Kotlin Multiplatform library for saving simple key-value data
- Kotlin multiplatform / multi-format serialization.
- KotlinX multiplatform date/time library.
- Logging library for Kotlin Multiplatform.
- Dependency injection framework for Kotlin developers.
- A modern I/O library for Android, Java, and Kotlin Multiplatform.
- Kotlin Multiplatform State Library.

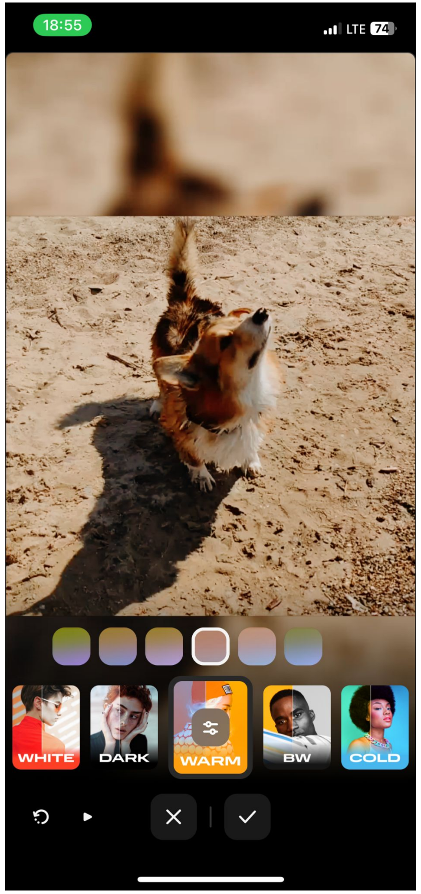
Архитектура MV*



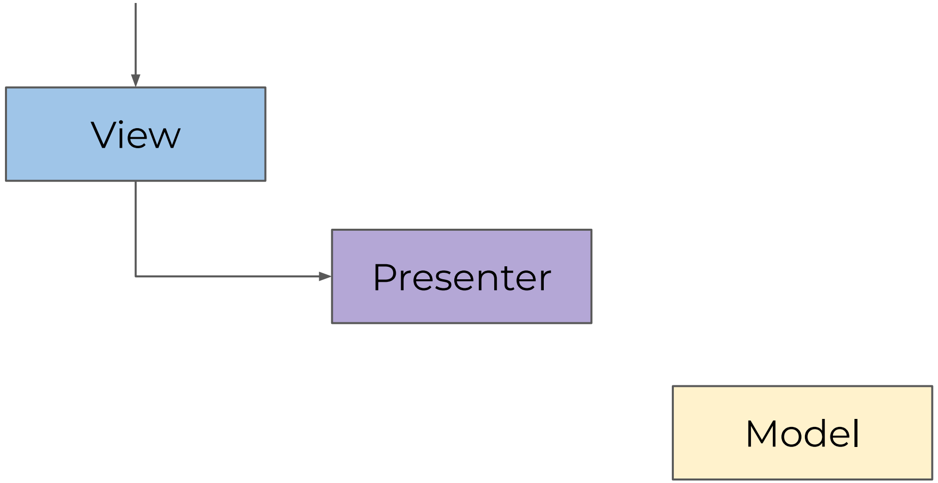


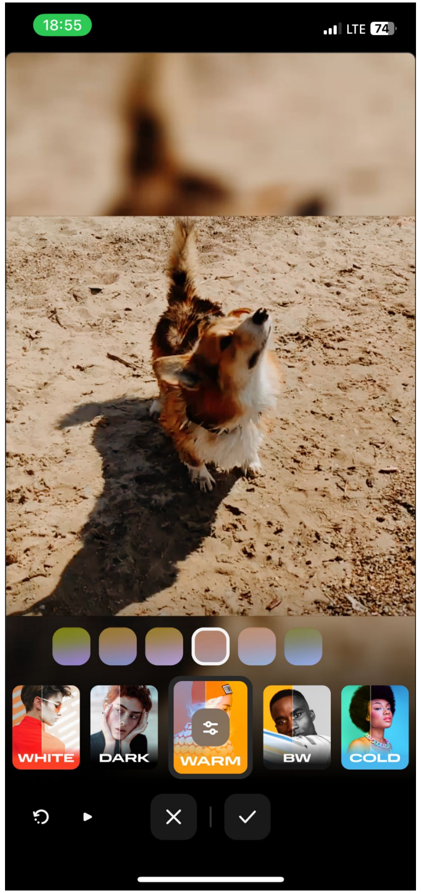
Архитектура MVP



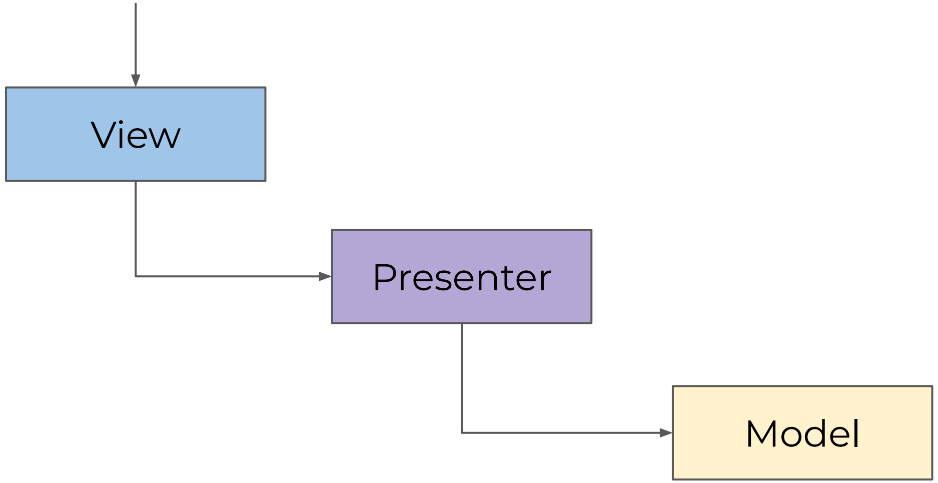


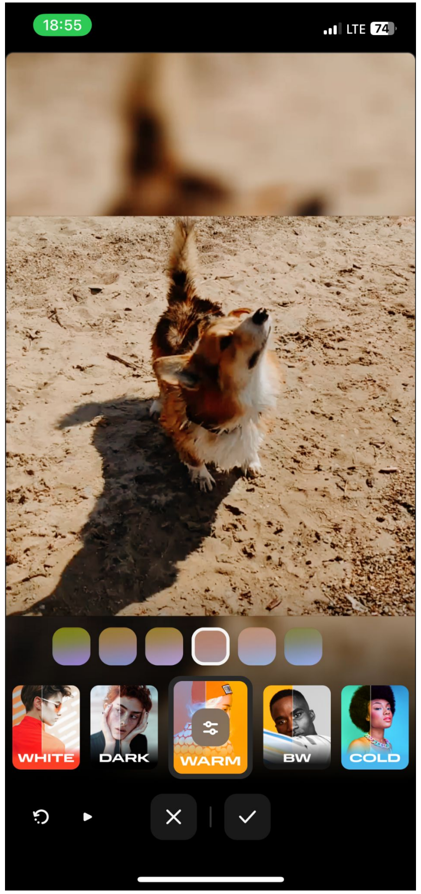
Архитектура MVP



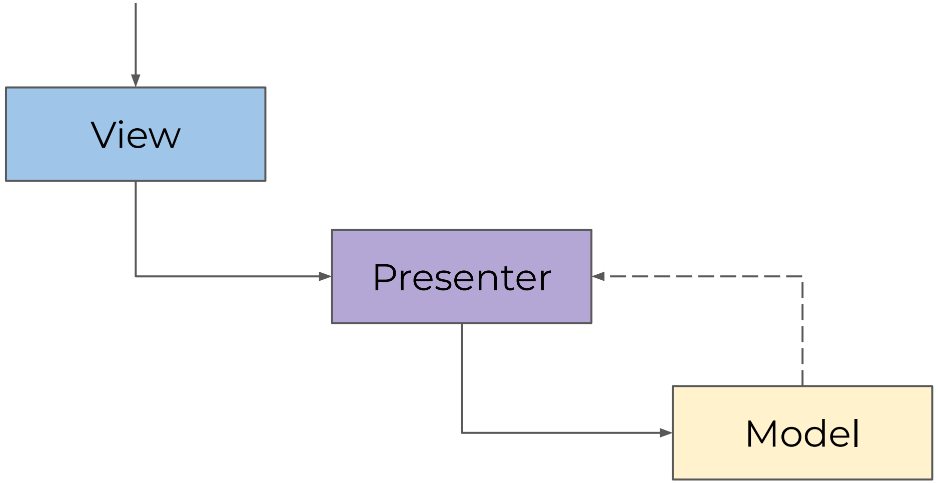


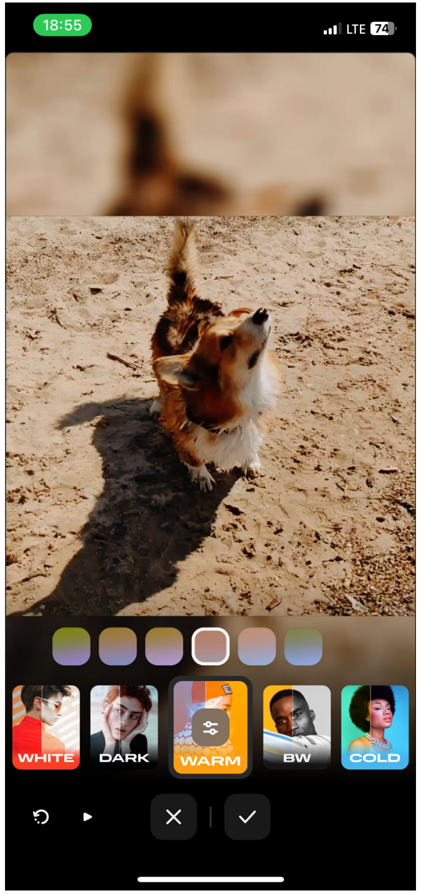
Архитектура MVP


$$\begin{tikzcd}
{} \arrow[d] & & \\
\text{View} \arrow[r] & \text{Presenter} \arrow[r] & \text{Model}
\end{tikzcd}$$

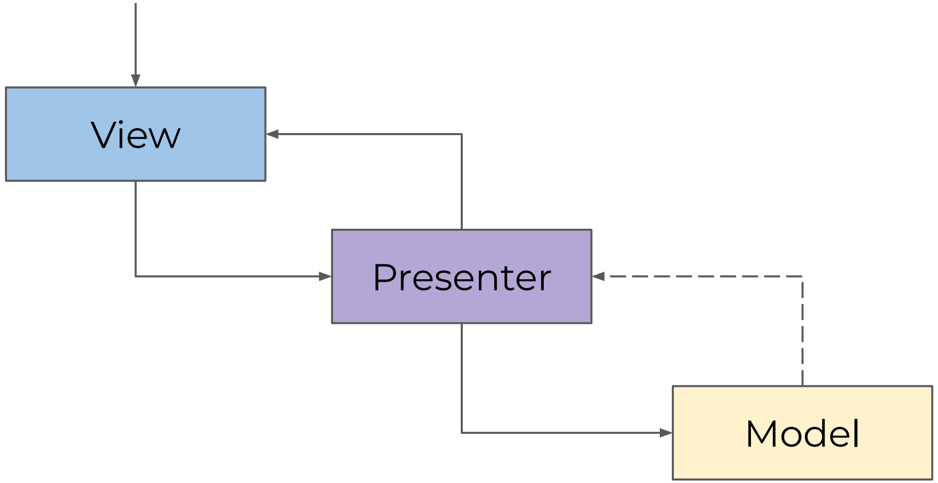


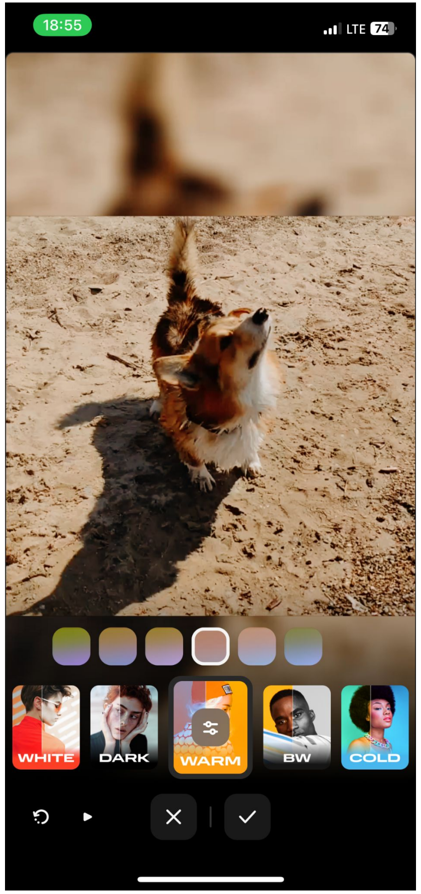
Архитектура MVP



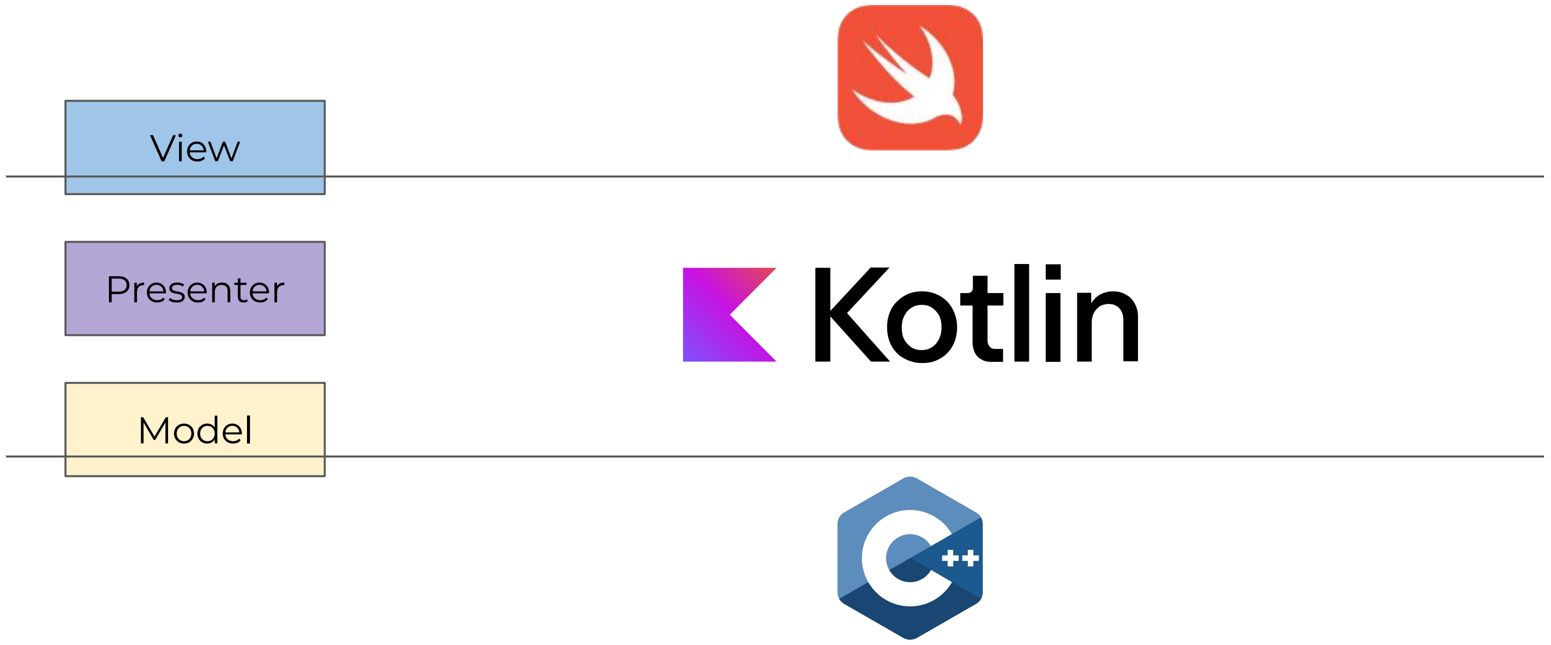


Архитектура MVP

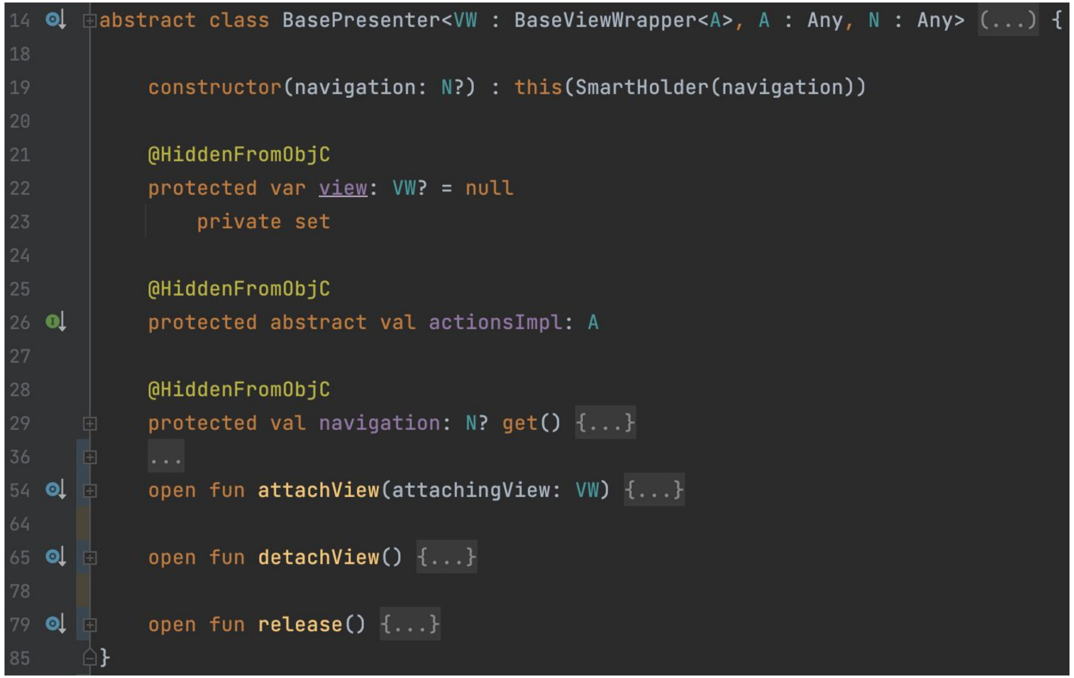




Архитектура MVP

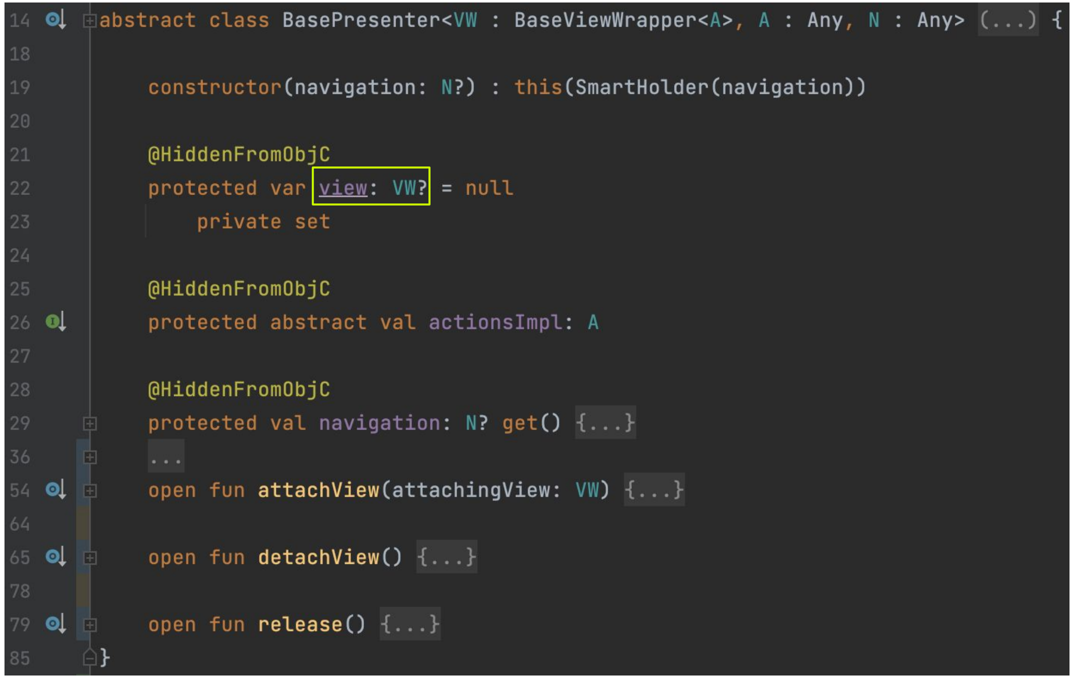


Архитектура MVP Presenter



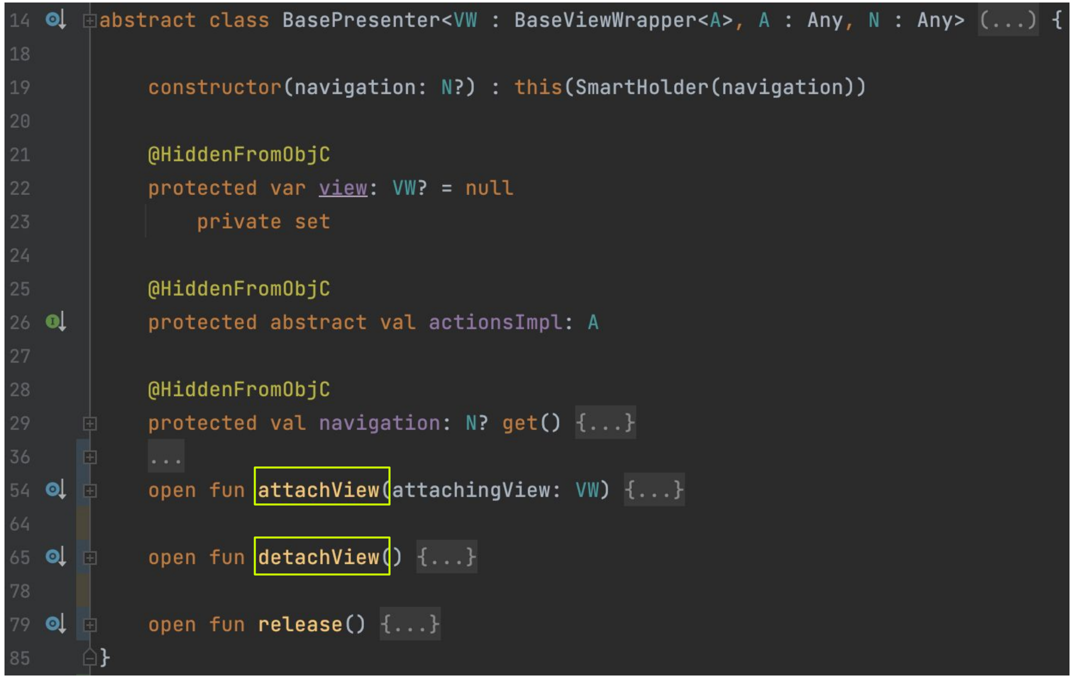
```
abstract class BasePresenter<VW : BaseViewWrapper<A>, A : Any, N : Any> (...) {

    constructor(navigation: N?) : this(SmartHolder(navigation))

    @HiddenFromObjC
    protected var view: VW? = null
        private set

    @HiddenFromObjC
    protected abstract val actionsImpl: A

    @HiddenFromObjC
    protected val navigation: N? get() {...}
    ...
    open fun attachView(attachingView: VW) {...}

    open fun detachView() {...}

    open fun release() {...}
}
```

Архитектура MVP
Presenter



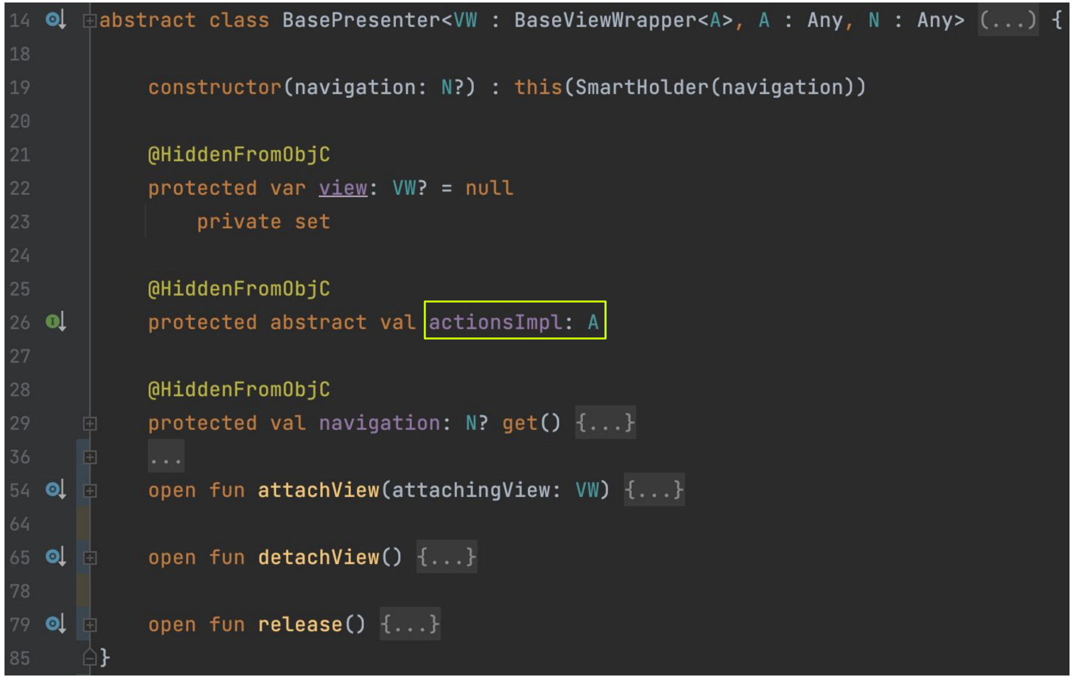
```
abstract class BasePresenter<VW : BaseViewWrapper<A>, A : Any, N : Any> (...) {

    constructor(navigation: N?) : this(SmartHolder(navigation))

    @HiddenFromObjC
    protected var view: VW? = null
        private set

    @HiddenFromObjC
    protected abstract val actionsImpl: A

    @HiddenFromObjC
    protected val navigation: N? get() {...}
    ...
    open fun attachView(attachingView: VW) {...}

    open fun detachView() {...}

    open fun release() {...}
}
```

Архитектура MVP Presenter



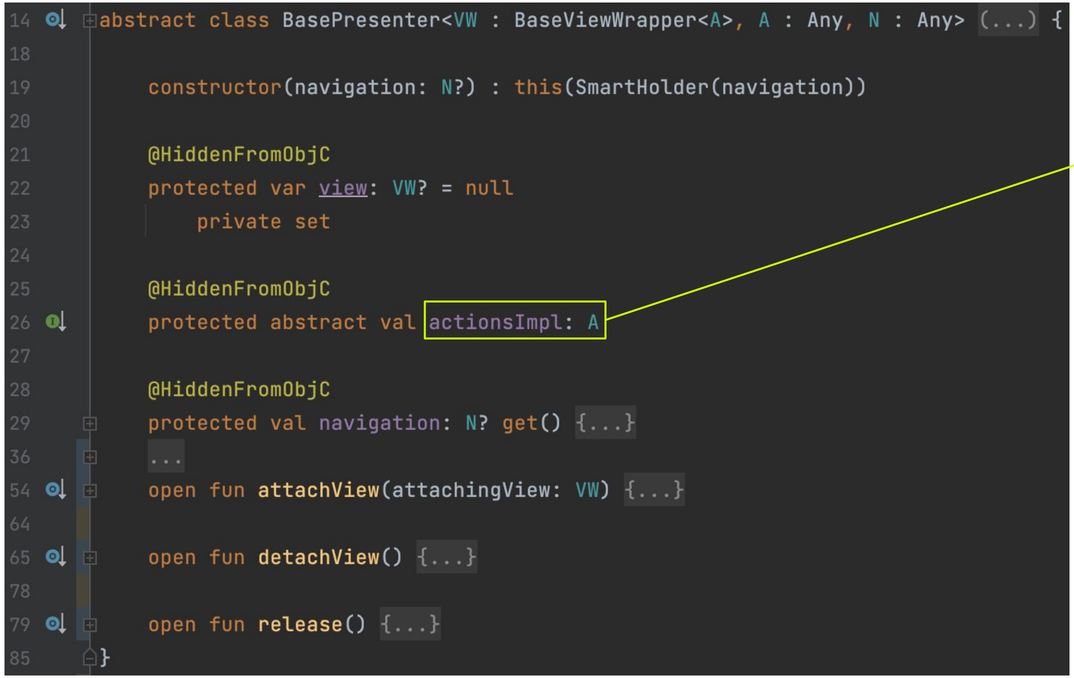
```
abstract class BaseViewWrapper<VW : BaseViewWrapper<A>, A : Any, N : Any> (...) {

    constructor(navigation: N?) : this(SmartHolder(navigation))

    @HiddenFromObjC
    protected var view: VW? = null
        private set

    @HiddenFromObjC
    protected abstract val actionsImpl: A

    @HiddenFromObjC
    protected val navigation: N? get() {...}
    ...
    open fun attachView(attachingView: VW) {...}

    open fun detachView() {...}

    open fun release() {...}
}
```

Архитектура MVP
Presenter

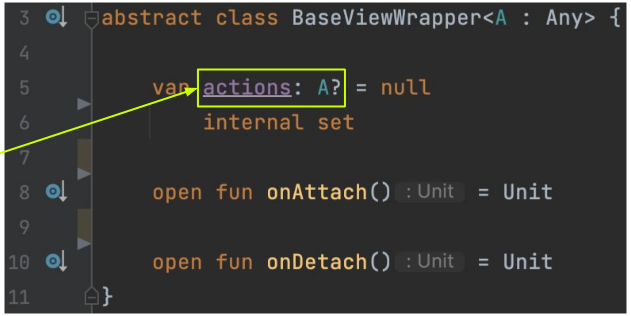


```
abstract class BasePresenter<VW : BaseViewWrapper<A>, A : Any, N : Any> (...) {

    constructor(navigation: N?) : this(SmartHolder(navigation))

    @HiddenFromObjC
    protected var view: VW? = null
        private set

    @HiddenFromObjC
    protected abstract val actionsImpl: A

    @HiddenFromObjC
    protected val navigation: N? get() {...}
    ...
    open fun attachView(attachingView: VW) {...}

    open fun detachView() {...}

    open fun release() {...}
}
```

Архитектура MVP
Presenter

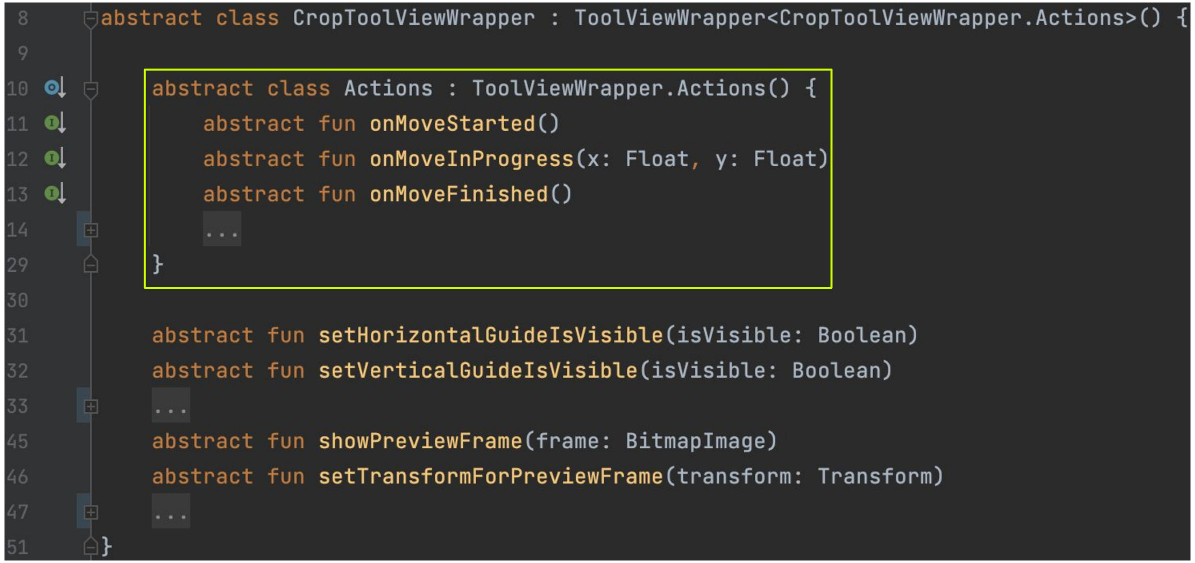


```
abstract class BasePresenter<VW : BaseViewWrapper<A>, A : Any, N : Any> (...) {

    constructor(navigation: N?) : this(SmartHolder(navigation))

    @HiddenFromObjC
    protected var view: VW? = null
        private set

    @HiddenFromObjC
    protected abstract val actionsImpl: A

    @HiddenFromObjC
    protected val navigation: N? get() {...}
    ...
    open fun attachView(attachingView: VW) {...}

    open fun detachView() {...}

    open fun release() {...}
}
```

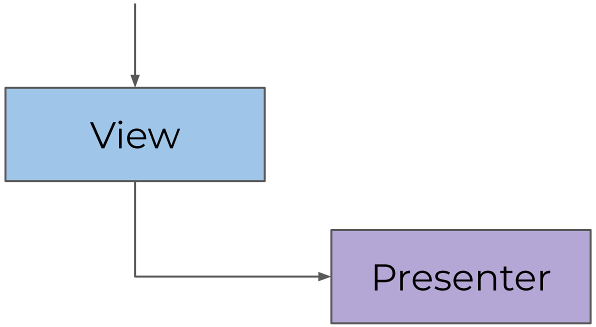


```
abstract class BaseViewWrapper<A : Any> {

    var actions: A? = null
        internal set

    open fun onAttach() : Unit = Unit

    open fun onDetach() : Unit = Unit
}
```

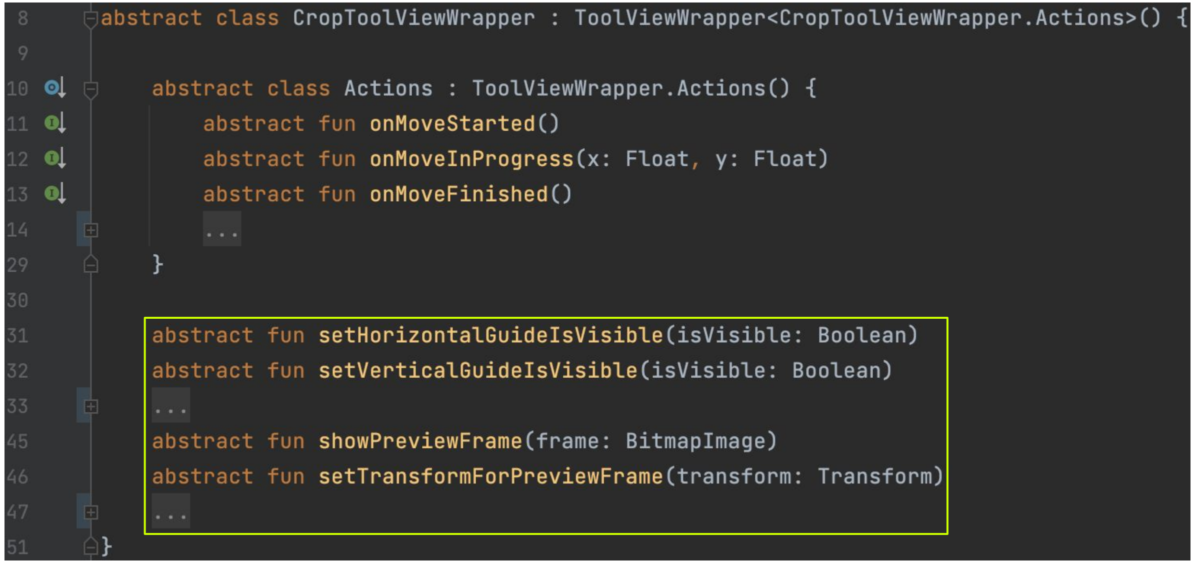
Архитектура MVP ViewWrapper



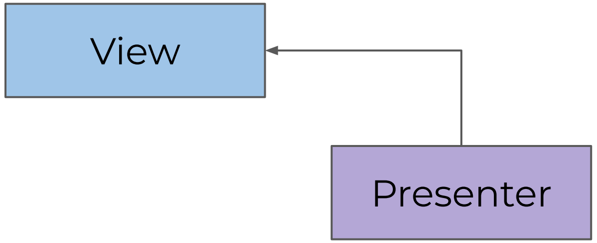
```kotlin
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```



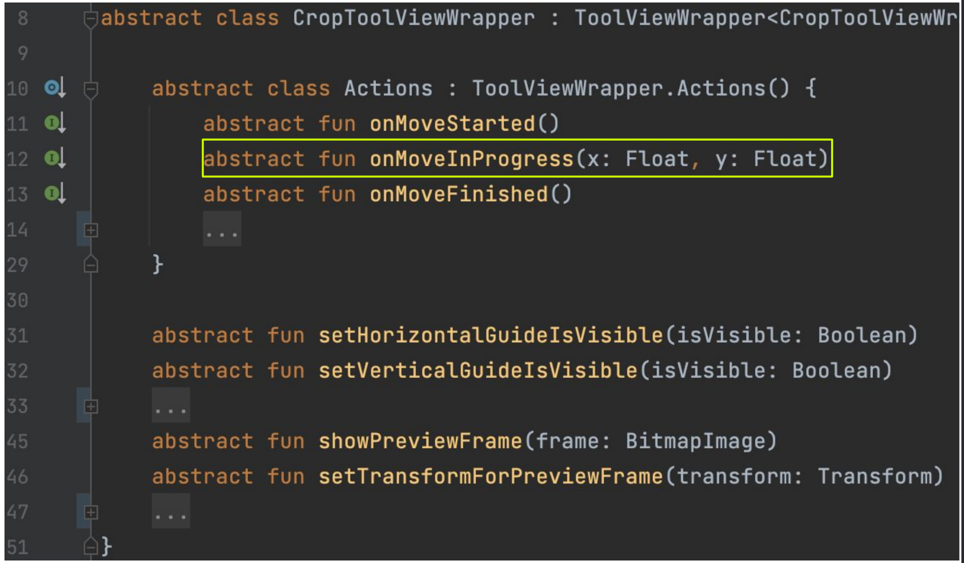
Архитектура MVP ViewWrapper



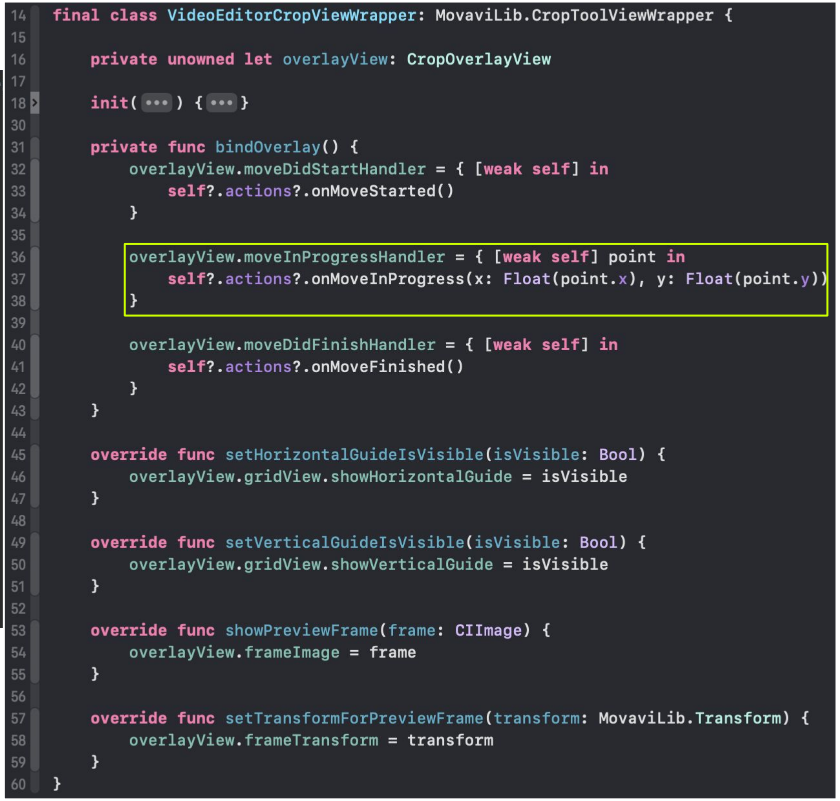
```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```



Архитектура MVP ViewWrapper

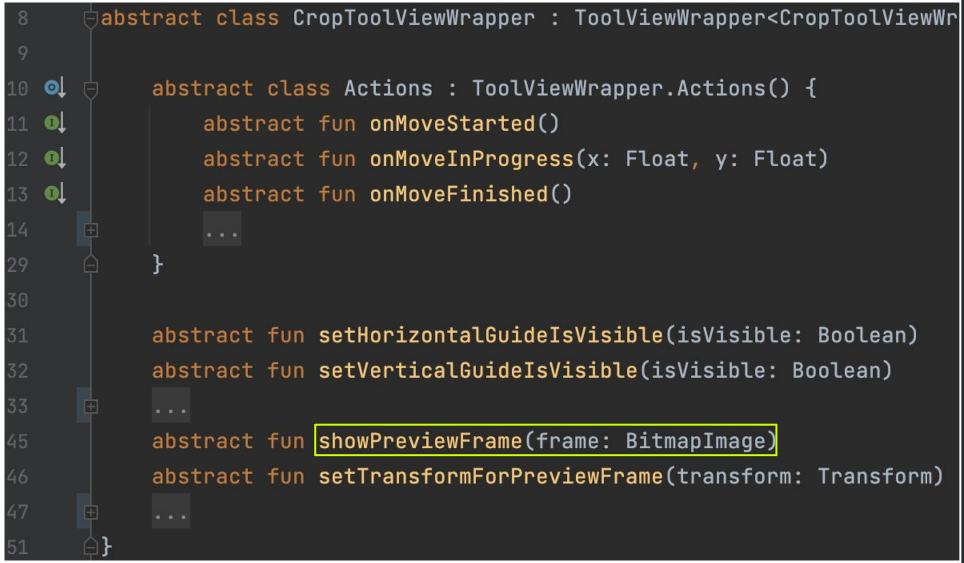


```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWr

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

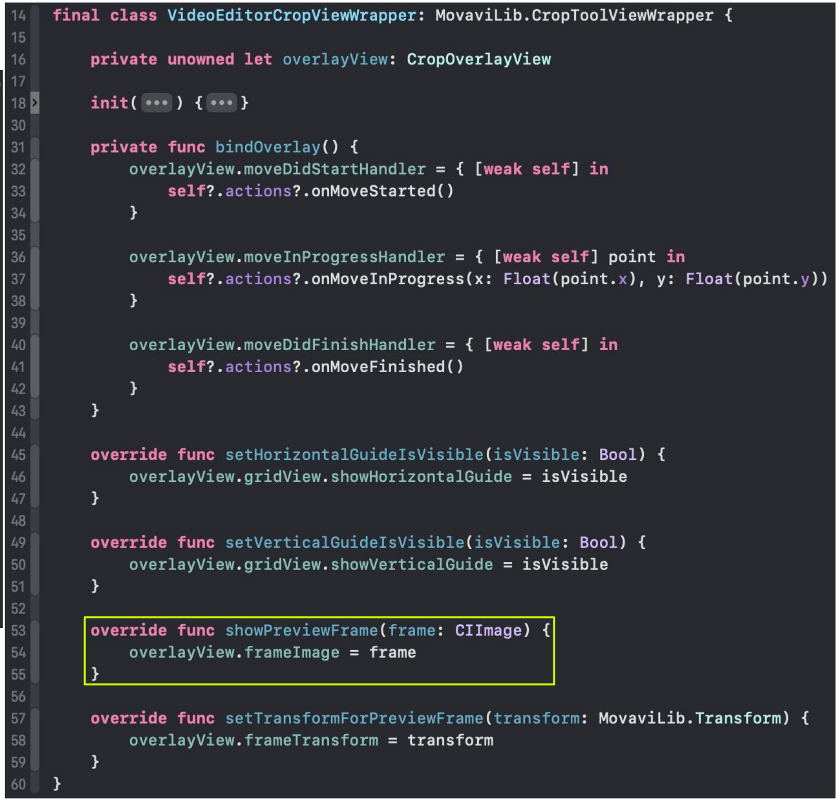


```
final class VideoEditorCropViewWrapper: MovaviLib.CropToolViewWrapper {

    private unowned let overlayView: CropOverlayView

    init(•••) {•••}

    private func bindOverlay() {
        overlayView.moveDidStartHandler = { [weak self] in
            self?.actions?.onMoveStarted()
        }

        overlayView.moveInProgressHandler = { [weak self] point in
            self?.actions?.onMoveInProgress(x: Float(point.x), y: Float(point.y))
        }

        overlayView.moveDidFinishHandler = { [weak self] in
            self?.actions?.onMoveFinished()
        }
    }

    override func setHorizontalGuideIsVisible(isVisible: Bool) {
        overlayView.gridView.showHorizontalGuide = isVisible
    }

    override func setVerticalGuideIsVisible(isVisible: Bool) {
        overlayView.gridView.showVerticalGuide = isVisible
    }

    override func showPreviewFrame(frame: CIImage) {
        overlayView.frameImage = frame
    }

    override func setTransformForPreviewFrame(transform: MovaviLib.Transform) {
        overlayView.frameTransform = transform
    }
}
```

Архитектура MVP ViewWrapper

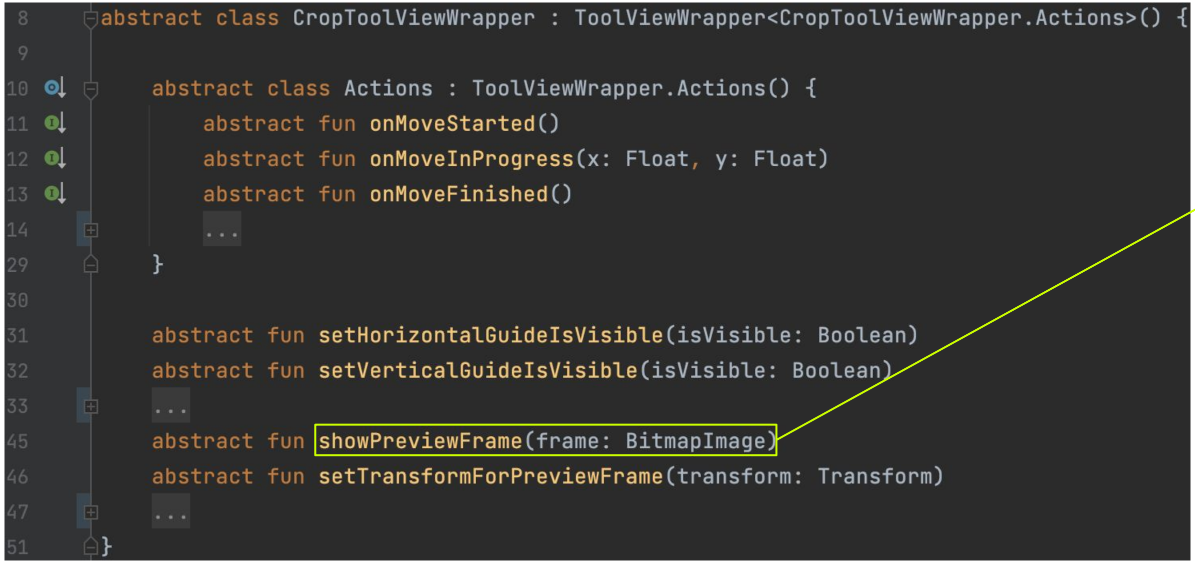


```kotlin
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWr

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

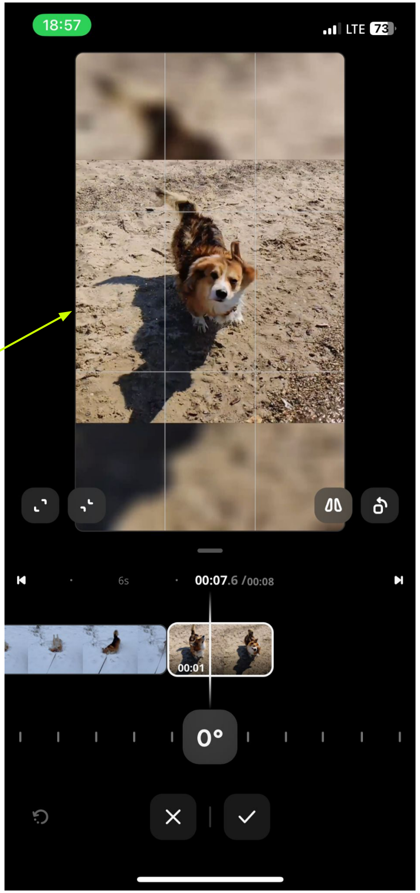


```swift
final class VideoEditorCropViewWrapper: MovaviLib.CropToolViewWrapper {

    private unowned let overlayView: CropOverlayView

    init(•••) {•••}

    private func bindOverlay() {
        overlayView.moveDidStartHandler = { [weak self] in
            self?.actions?.onMoveStarted()
        }

        overlayView.moveInProgressHandler = { [weak self] point in
            self?.actions?.onMoveInProgress(x: Float(point.x), y: Float(point.y))
        }

        overlayView.moveDidFinishHandler = { [weak self] in
            self?.actions?.onMoveFinished()
        }
    }

    override func setHorizontalGuideIsVisible(isVisible: Bool) {
        overlayView.gridView.showHorizontalGuide = isVisible
    }

    override func setVerticalGuideIsVisible(isVisible: Bool) {
        overlayView.gridView.showVerticalGuide = isVisible
    }

    override func showPreviewFrame(frame: CIImage) {
        overlayView.frameImage = frame
    }

    override func setTransformForPreviewFrame(transform: MovaviLib.Transform) {
        overlayView.frameTransform = transform
    }
}
```

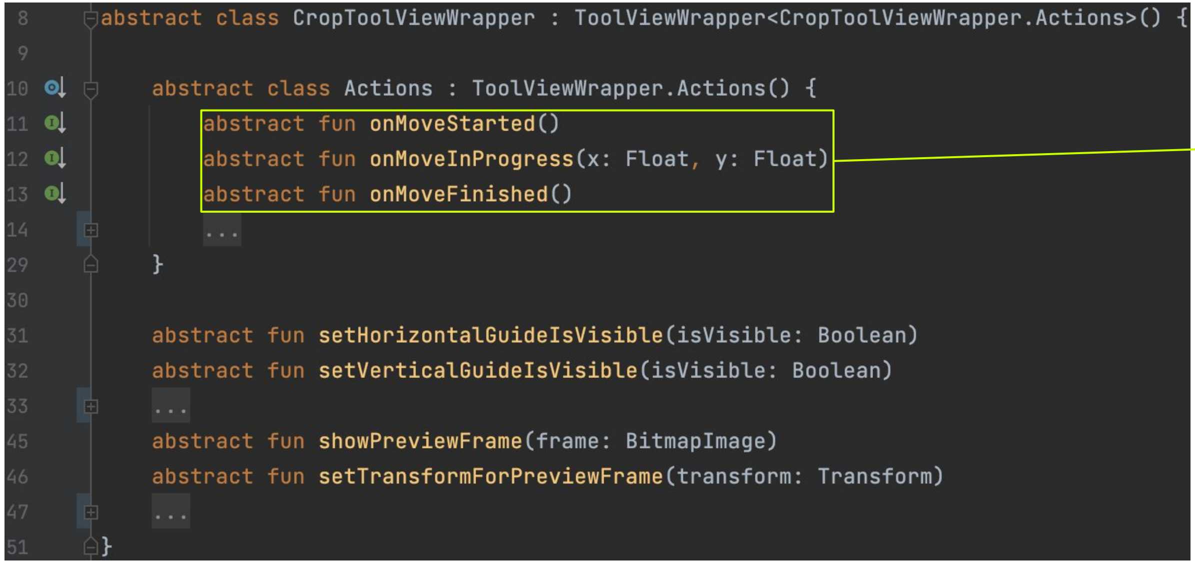
Архитектура MVP ViewWrapper



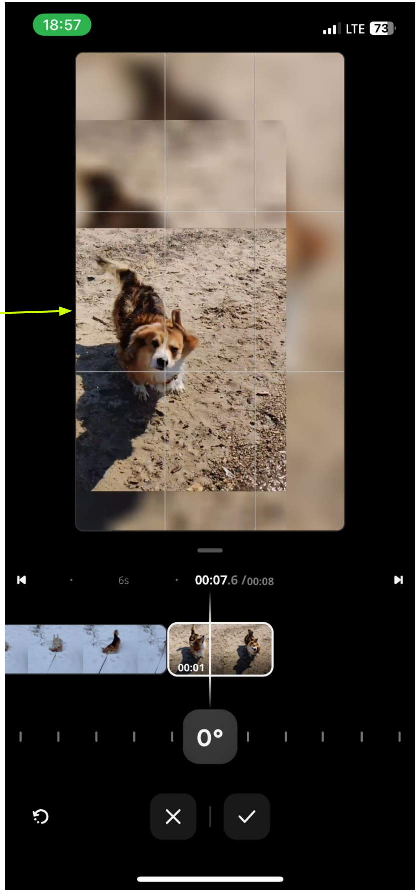
```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```



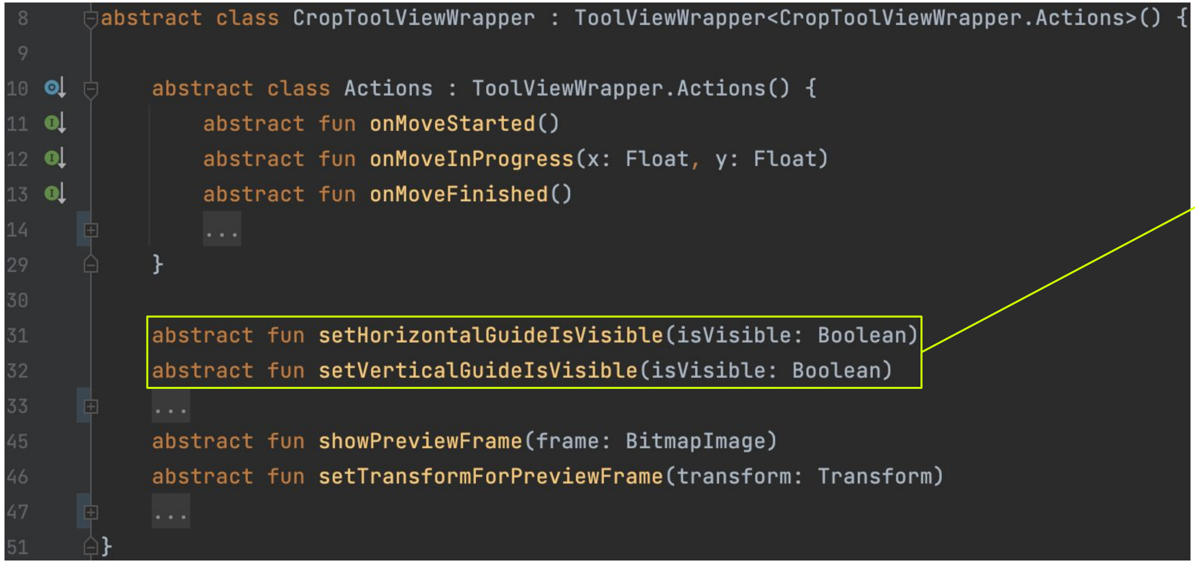
Архитектура MVP ViewWrapper



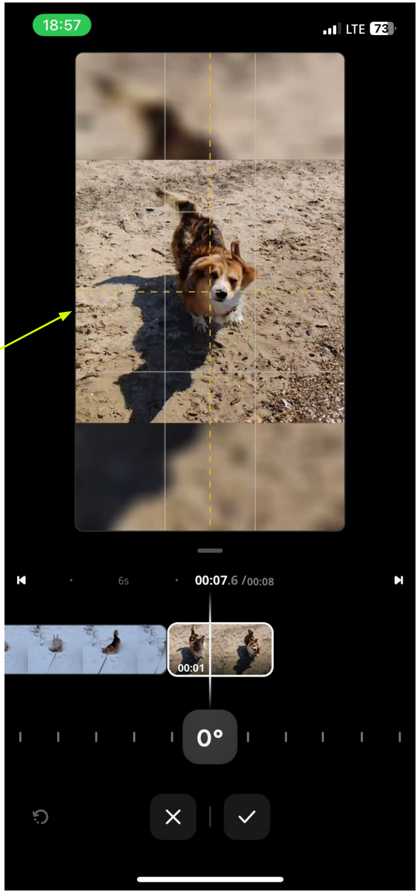
```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```



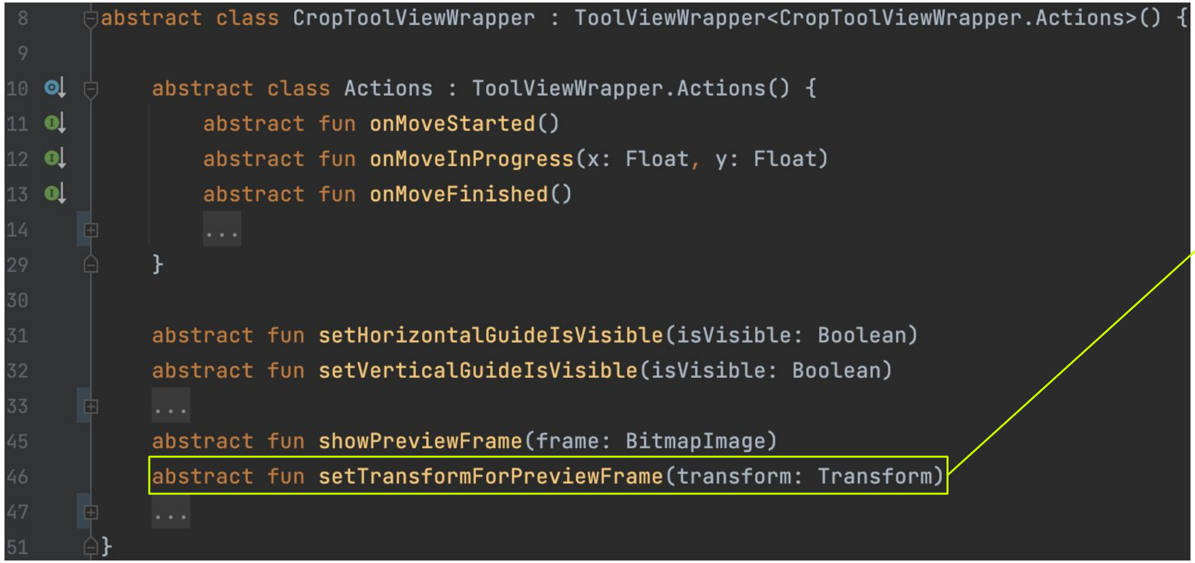
Архитектура MVP ViewWrapper



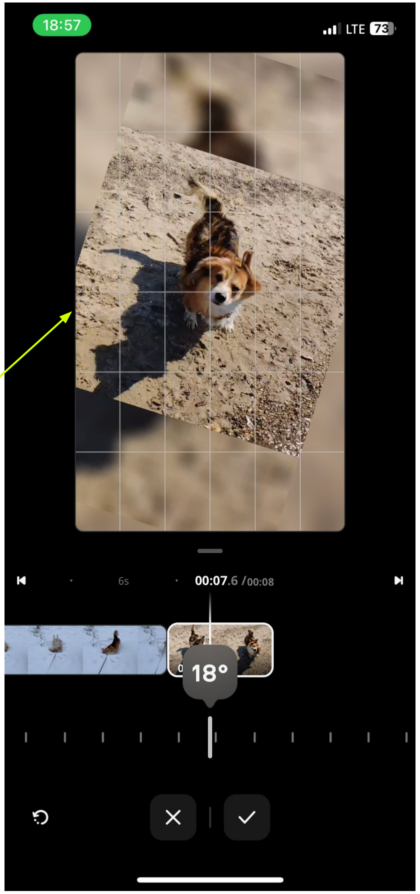
```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```



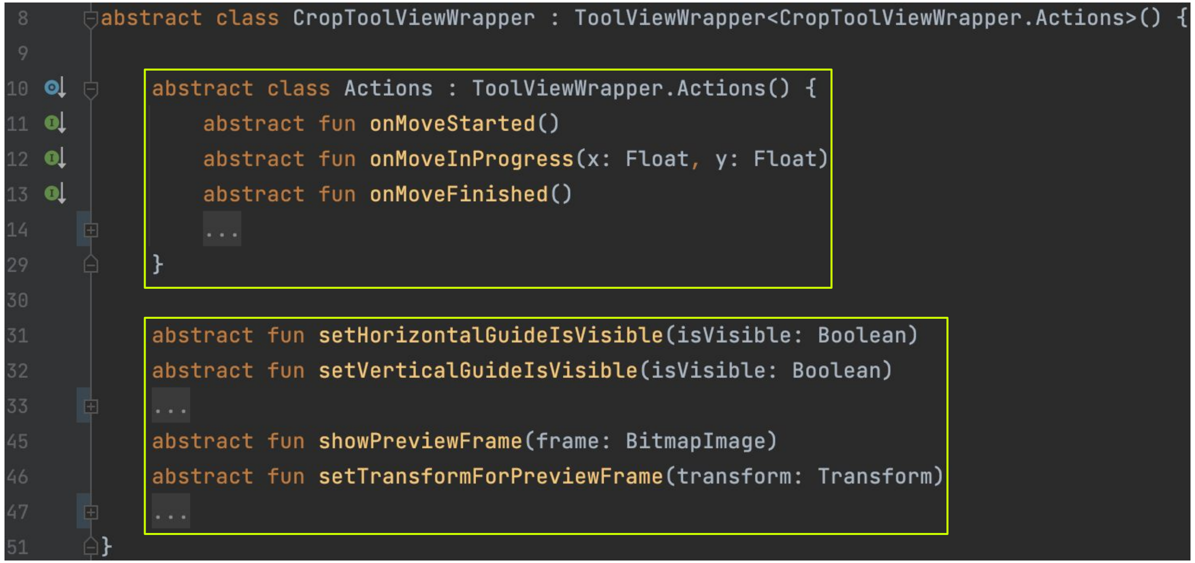
Архитектура MVP ViewWrapper



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```



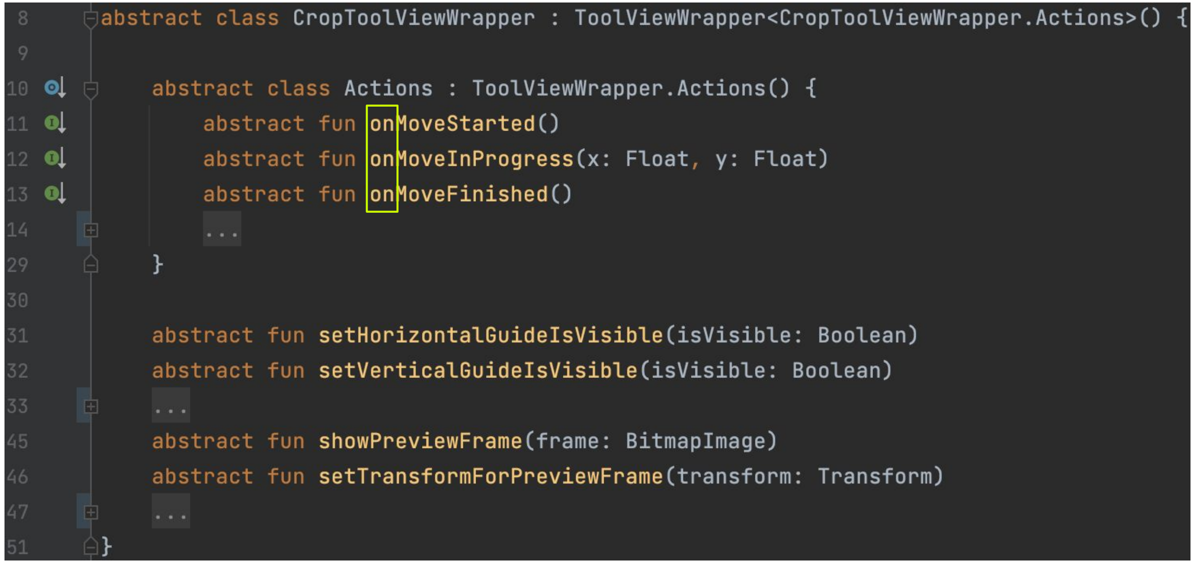
Архитектура MVP
Важные моменты



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

- UI Thread

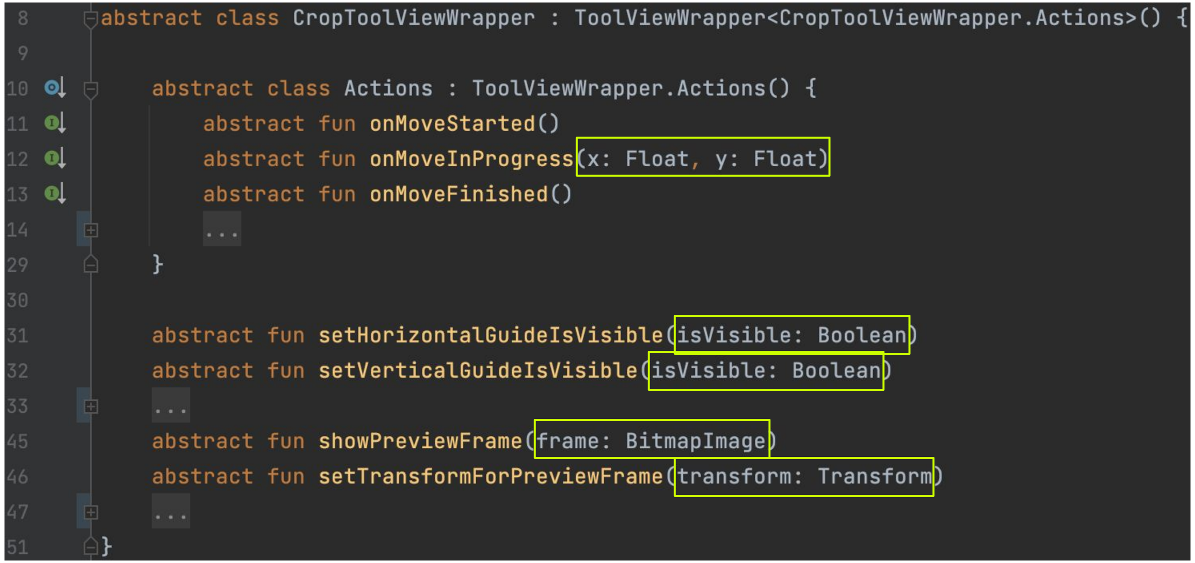
Архитектура MVP
Важные моменты



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

- UI Thread
- Actions.on...()

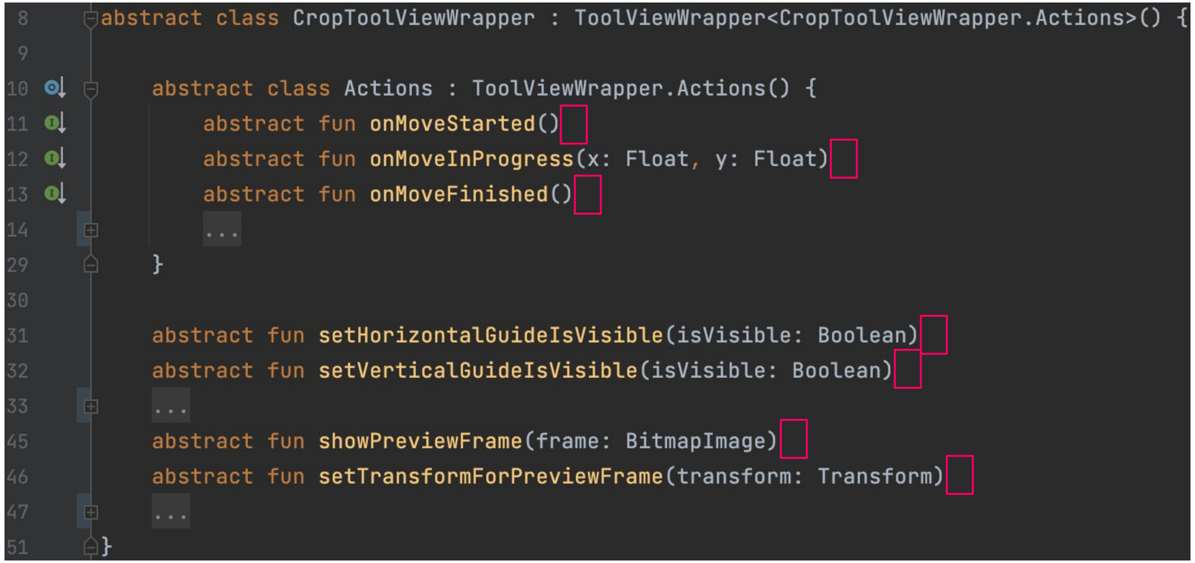
Архитектура MVP
Важные моменты



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

- UI Thread
- Actions.on...()
- fun ...(args)

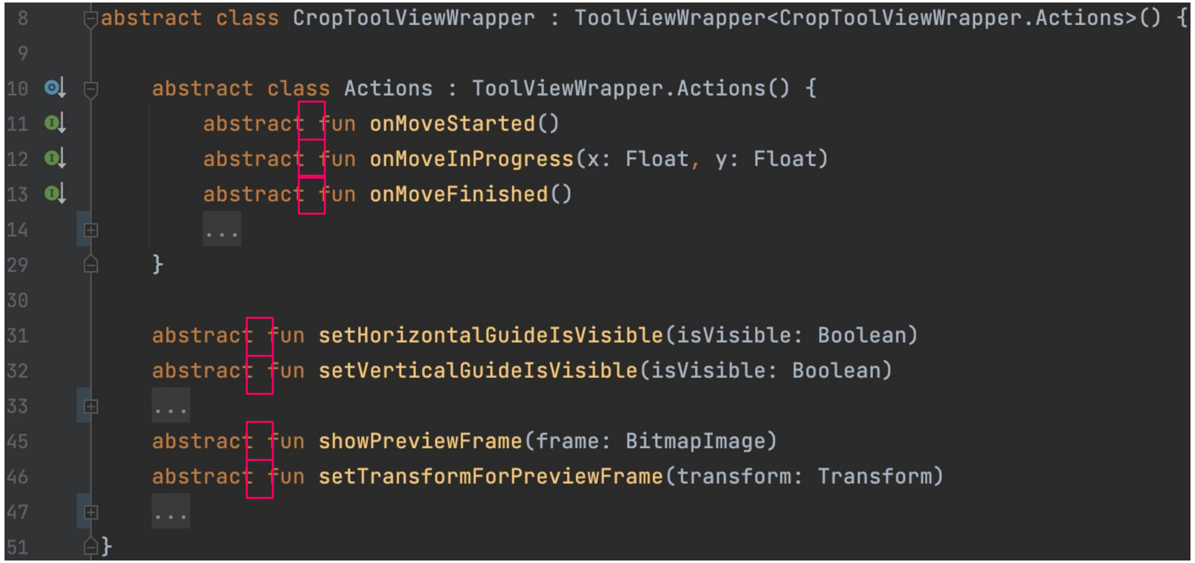
Архитектура MVP
Важные моменты



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

- UI Thread
- Actions.on...()
- fun ...(args)
- ~~return value~~

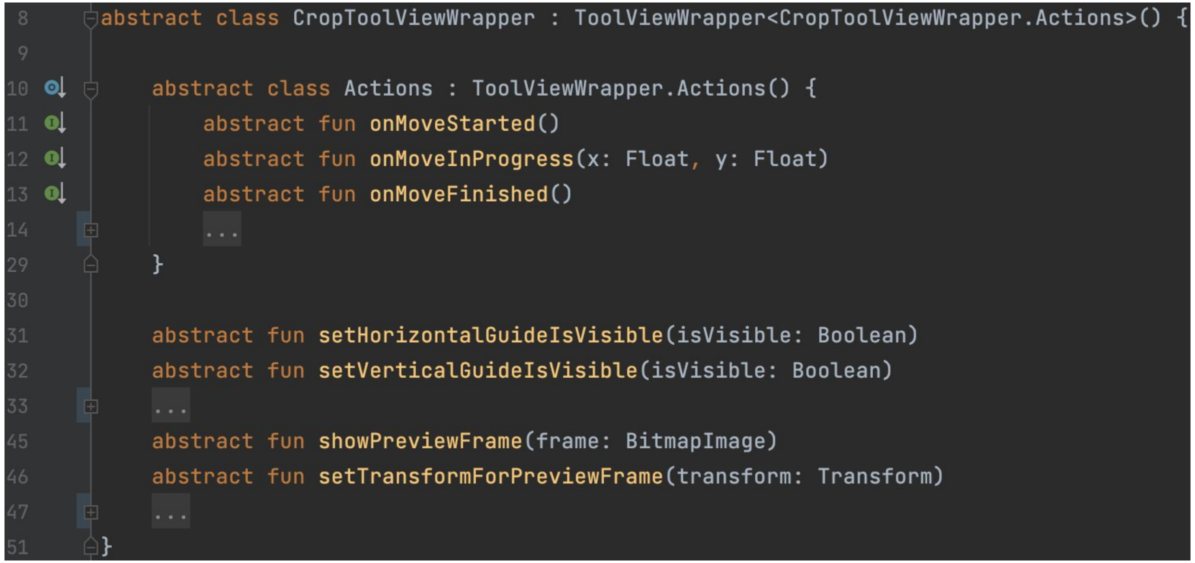
Архитектура MVP
Важные моменты



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

- UI Thread
- Actions.on...()
- fun ...(args)
- ~~return value~~
- ~~suspend fun~~

Архитектура MVP
Важные моменты



```
abstract class CropToolViewWrapper : ToolViewWrapper<CropToolViewWrapper.Actions>() {

    abstract class Actions : ToolViewWrapper.Actions() {
        abstract fun onMoveStarted()
        abstract fun onMoveInProgress(x: Float, y: Float)
        abstract fun onMoveFinished()
        ...
    }

    abstract fun setHorizontalGuideIsVisible(isVisible: Boolean)
    abstract fun setVerticalGuideIsVisible(isVisible: Boolean)
    ...
    abstract fun showPreviewFrame(frame: BitmapImage)
    abstract fun setTransformForPreviewFrame(transform: Transform)
    ...
}
```

- UI Thread
- Actions.on…()
- fun …(args)
- ~~return value~~
- ~~suspend fun~~
- ~~@Throws~~

Kotlin Multiplatform Release Notes

Kotlin

Kotlin Multiplatform Versions

1.6.20

An update on the new memory manager

> The new Kotlin/Native memory manager is in Alpha. It may change incompatibly and require manual migration in the future. We would appreciate your feedback on it in YouTrack ↗.

With Kotlin 1.6.20, you can try the Alpha version of the new Kotlin/Native memory manager. It eliminates the differences between the JVM and Native platforms to provide a consistent developer experience in multiplatform projects. For example, you'll have a much easier time creating new cross-platform mobile applications that work on both Android and iOS.

The new Kotlin/Native memory manager lifts restrictions on object-sharing between threads. It also provides leak-free concurrent programming primitives that are safe and don't require any special management or annotations.

The new memory manager will become the default in future versions, so we encourage you to try it now. Check out our blog post ↗ to learn more about the new memory manager and explore demo projects, or jump right to the migration instructions ↗ to try it yourself.

Try using the new memory manager on your projects to see how it works and share feedback in our issue tracker, YouTrack ↗.

Performance improvements

We are working hard on Kotlin/Native to speed up the compilation process ↗ and improve your developing experience.

Kotlin 1.6.20 brings some performance updates and bug fixes that affect the LLVM IR that Kotlin generates. According to the benchmarks on our internal projects, we achieved the following performance boosts on average:

- 15% reduction in execution time
- 20% reduction in the code size of both release and debug binaries
- 26% reduction in the compilation time of release binaries

These changes also provide a 10% reduction in compilation time for a debug binary on a large internal project.

To achieve this, we've implemented static initialization for some of the compiler-generated synthetic objects, improved the way we structure LLVM IR for every function, and optimized the compiler caches.

Kotlin Multiplatform Versions

1.6.20 1.7.0

Prohibited undeclared exceptions through Objective-C bridges

When you call Kotlin code from Swift/Objective-C code (or vice versa) and this code throws an exception, it should be handled by the code where the exception occurred, unless you specifically allowed the forwarding of exceptions between languages with proper conversion (for example, using the `@Throws` annotation).

Previously, Kotlin had another unintended behavior where undeclared exceptions could "leak" from one language to another in some cases. Kotlin 1.7.0 fixes that issue, and now such cases lead to program termination.

So, for example, if you have a `{ throw Exception() }` lambda in Kotlin and call it from Swift, in Kotlin 1.7.0 it will terminate as soon as the exception reaches the Swift code. In previous Kotlin versions, such an exception could leak to the Swift code.

The `@Throws` annotation continues to work as before.

Improved CocoaPods integration

Starting with Kotlin 1.7.0, you no longer need to install the `cocoapods-generate` plugin if you want to integrate CocoaPods in your projects.

Previously, you needed to install both the CocoaPods dependency manager and the `cocoapods-generate` plugin to use CocoaPods, for example, to handle iOS dependencies in Kotlin Multiplatform Mobile projects.

Now setting up the CocoaPods integration is easier, and we've resolved the issue when `cocoapods-generate` couldn't be installed on Ruby 3 and later. Now the newest Ruby versions that work better on Apple M1 are also supported.

See how to set up the initial CocoaPods integration.

Kotlin Multiplatform Versions

1.6.20
1.7.0
1.7.20

The new Kotlin/Native memory manager enabled by default

This release brings further stability and performance improvements to the new memory manager, allowing us to promote the new memory manager to Beta.

The previous memory manager complicated writing concurrent and asynchronous code, including issues with implementing the `kotlinx.coroutines` library. This blocked the adoption of Kotlin Multiplatform Mobile because concurrency limitations created problems with sharing Kotlin code between iOS and Android platforms. The new memory manager finally paves the way to promote Kotlin Multiplatform Mobile to Beta ↗.

The new memory manager also supports the compiler cache that makes compilation times comparable to previous releases. For more on the benefits of the new memory manager, see our original blog post ↗ for the preview version. You can find more technical details in the documentation.

Kotlin Multiplatform Versions

1.6.20
1.7.0
1.7.20
1.8.0

Improved Objective-C/Swift interoperability

To make Kotlin more interoperable with Objective-C and Swift, three new annotations were added:

- `@ObjCName` ↗ allows you to specify a more idiomatic name in Swift or Objective-C, instead of renaming the Kotlin declaration.

  The annotation instructs the Kotlin compiler to use a custom Objective-C and Swift name for this class, property, parameter, or function:

  ```
  @ObjCName(swiftName = "MySwiftArray")
  class MyKotlinArray {
      @ObjCName("index")
      fun indexOf(@ObjCName("of") element: String): Int = TODO()
  }

  // Usage with the ObjCName annotations
  let array = MySwiftArray()
  let index = array.index(of: "element")
  ```

- `@HiddenFromObjC` ↗ allows you to hide a Kotlin declaration from Objective-C.

  The annotation instructs the Kotlin compiler not to export a function or property to Objective-C and, consequently, Swift. This can make your Kotlin code more Objective-C/Swift-friendly.

Kotlin Multiplatform Versions

1.6.20
1.7.0
1.7.20
1.8.0
1.8.20

Deprecation of the legacy memory manager

Starting with 1.8.20, the legacy memory manager is deprecated and will be removed in 1.9.20. The new memory manager was enabled by default in 1.7.20 and has been receiving further stability updates and performance improvements.

If you're still using the legacy memory manager, remove the `kotlin.native.binary.memoryModel=strict` option from your `gradle.properties` and follow our Migration guide to make the necessary changes.

The new memory manager doesn't support the `wasm32` target. This target is also deprecated starting with this release and will be removed in 1.9.20.

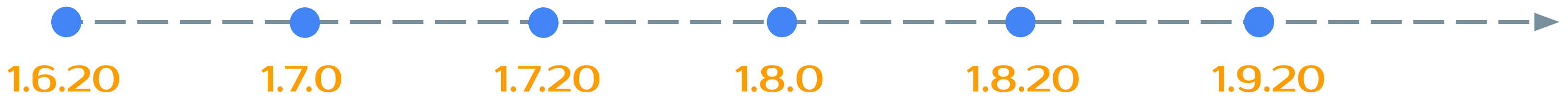
Reimplementation of compiler cache management in the compiler

To speed up the evolution of compiler caches, we've moved compiler cache management from the Kotlin Gradle plugin to the Kotlin/Native compiler. This unblocks work on several important improvements, including those to do with compilation times and compiler cache flexibility.

If you encounter some problem and need to return to the old behavior, use the `kotlin.native.cacheOrchestration=gradle` Gradle property.

We would appreciate your feedback on this in YouTrack ↗.

Kotlin Multiplatform Versions

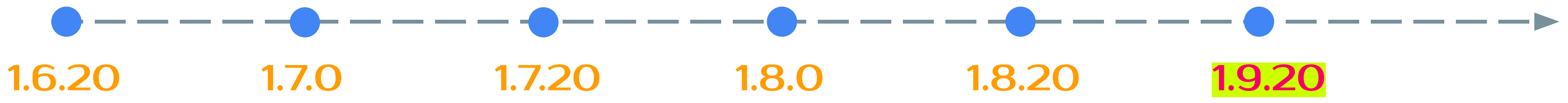


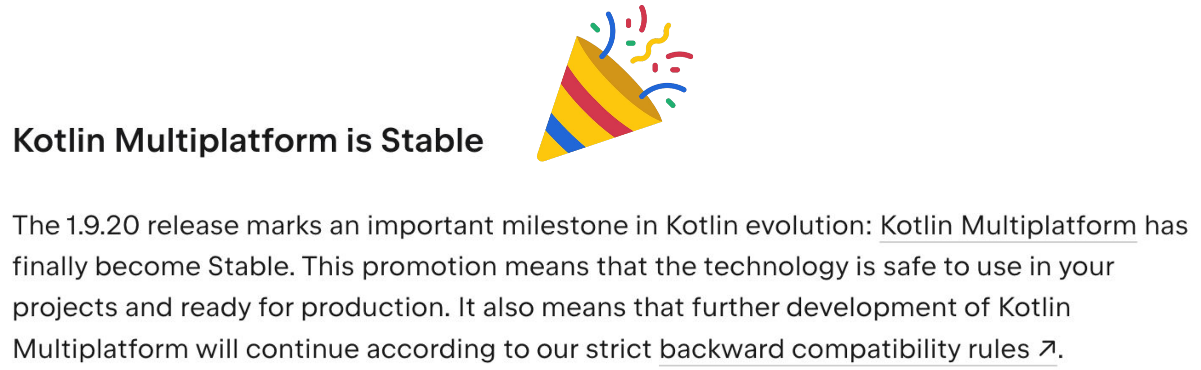
Performance improvements for the garbage collector

The Kotlin team continues to improve the performance and stability of the new Kotlin/Native memory manager. This release brings a number of significant changes to the garbage collector (GC), including the following 1.9.20 highlights:

- Full parallel mark to reduce the pause time for the GC
- Tracking memory in big chunks to improve the allocation performance

Kotlin Multiplatform Versions





Kotlin Multiplatform is Stable 🎉

The 1.9.20 release marks an important milestone in Kotlin evolution: Kotlin Multiplatform has finally become Stable. This promotion means that the technology is safe to use in your projects and ready for production. It also means that further development of Kotlin Multiplatform will continue according to our strict backward compatibility rules ↗.

Kotlin Multiplatform Roadmap

Kotlin

Kotlin Multiplatform Точки роста

- **Скорость сборки**

  Apple M2 Max (32ГБ)

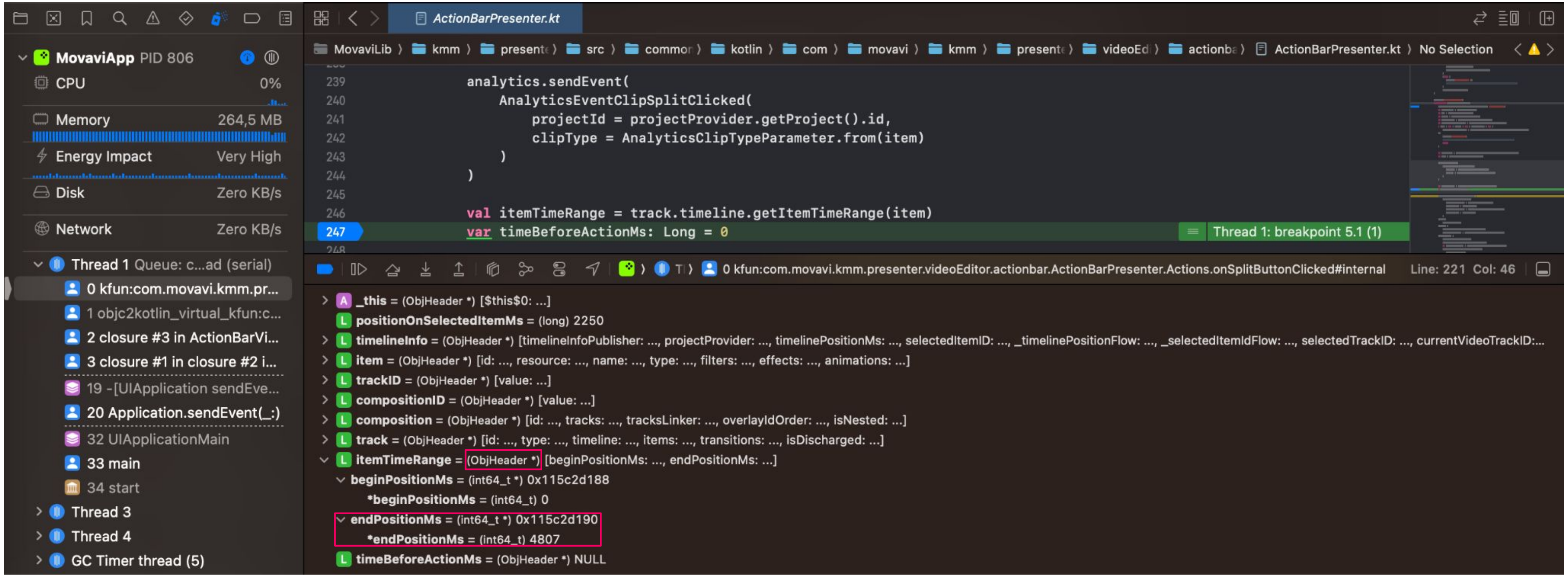
  - всего на проект: ~140 сек
  - из них KMP часть: ~90 сек

Kotlin Multiplatform Точки роста

- **Скорость сборки**

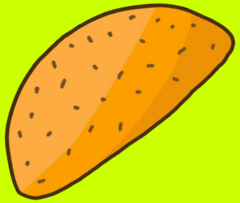
  Apple M2 Max (32ГБ)
  - всего на проект: ~140 сек
  - из них KMP часть: ~90 сек
- **Отладка**
  - Есть возможность отлаживать в Android Studio
  - По умолчанию нет возможности отлаживать KMP часть в Xcode
  - Есть “Kotlin Native Xcode Plugin”: xcode-kotlin

Kotlin Multiplatform Отладка в Xcode


ActionBarPresenter.kt
MovaviLib ⟩ kmm ⟩ presente ⟩ src ⟩ commor ⟩ kotlin ⟩ com ⟩ movavi ⟩ kmm ⟩ presente ⟩ videoEd ⟩ actionba ⟩ ActionBarPresenter.kt ⟩ No Selection
MovaviApp PID 806
CPU 0%
Memory 264,5 MB
Energy Impact Very High
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: c...ad (serial)
0 kfun:com.movavi.kmm.pr...
1 objc2kotlin_virtual_kfun:c...
2 closure #3 in ActionBarVi...
3 closure #1 in closure #2 i...
19 -[UIApplication sendEve...
20 Application.sendEvent(_:)
32 UIApplicationMain
33 main
34 start
Thread 3
Thread 4
GC Timer thread (5)
Thread 1: breakpoint 5.1 (1)
0 kfun:com.movavi.kmm.presenter.videoEditor.actionbar.ActionBarPresenter.Actions.onSplitButtonClicked#internal
Line: 221 Col: 46

```
            analytics.sendEvent(
                AnalyticsEventClipSplitClicked(
                    projectId = projectProvider.getProject().id,
                    clipType = AnalyticsClipTypeParameter.from(item)
                )
            )

            val itemTimeRange = track.timeline.getItemTimeRange(item)
            var timeBeforeActionMs: Long = 0
```

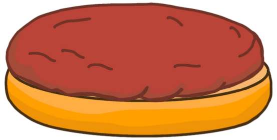
```
> A _this = (ObjHeader *) [$this$0: ...]
  L positionOnSelectedItemMs = (long) 2250
> L timelineInfo = (ObjHeader *) [timelineInfoPublisher: ..., projectProvider: ..., timelinePositionMs: ..., selectedItemID: ..., _timelinePositionFlow: ..., _selectedItemIdFlow: ..., selectedTrackID: ..., currentVideoTrackID:...
> L item = (ObjHeader *) [id: ..., resource: ..., name: ..., type: ..., filters: ..., effects: ..., animations: ...]
> L trackID = (ObjHeader *) [value: ...]
> L compositionID = (ObjHeader *) [value: ...]
> L composition = (ObjHeader *) [id: ..., tracks: ..., tracksLinker: ..., overlayIdOrder: ..., isNested: ...]
> L track = (ObjHeader *) [id: ..., type: ..., timeline: ..., items: ..., transitions: ..., isDischarged: ...]
v L itemTimeRange = (ObjHeader *) [beginPositionMs: ..., endPositionMs: ...]
   v beginPositionMs = (int64_t *) 0x115c2d188
       *beginPositionMs = (int64_t) 0
   v endPositionMs = (int64_t *) 0x115c2d190
       *endPositionMs = (int64_t) 4807
  L timeBeforeActionMs = (ObjHeader *) NULL
```



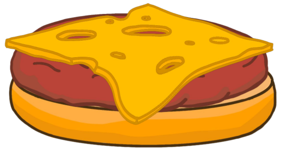
Рецепт приготовления кроссплатформенного мобильного видеоредактора

Kotlin

Рецепт

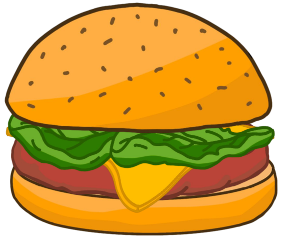


Рецепт



Kotlin

Рецепт



Спасибо за внимание