

React Nivel Avanzado  
Sesión sincrónica 7

# Rendimiento de la aplicación



# Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica	Crecimiento postraumático.
Descripción de la práctica	En esta práctica harás un recuento de las situaciones difíciles a las que te has enfrentado y reflexionarás sobre lo positivo que surgió de ellas.
Palabras clave	Resiliencia.
Instrucciones para el aprendedor	<p>La resiliencia es la capacidad de reponerse tras la adversidad, de recuperarse después de vivir experiencias difíciles, dolorosas o traumáticas. Para algunos la resiliencia implica, no solo salir adelante después de una situación muy dura, sino, incluso, crecer o ser mejor a raíz de esta experiencia (Tarragona, como se citó en Palomar. J., y Gaxiola, J., 2012).</p> <p>La siguiente práctica te ayudará a fomentar esta importante cualidad:</p> <ol style="list-style-type: none"><li>1. Escribe acerca de un momento en el que enfrentaste una pérdida o una adversidad significativa.</li><li>2. Primero, escribe acerca de las puertas que se te cerraron debido a esa situación. ¿Qué perdiste?</li><li>3. Después, escribe sobre las puertas que se abrieron como consecuencia de esa pérdida o adversidad.</li><li>4. ¿Hay más probabilidad de que nuevas maneras de actuar, pensar o relacionarse, sucedan ahora?</li></ol>

# Bienvenida y actividad de bienestar

Fuente

- Ejercicio contribuido por Taylor Kreiss del Positive Psychology Center, University of Pennsylvania, basado en la fuente: Peterson, C. (2006). *A Primer in Positive Psychology*. Estados Unidos: Oxford University Press.
- Palomar. J., y Gaxiola, J. *Estudios de resiliencia en América Latina Volumen 1*. México: Universidad de Sonora.

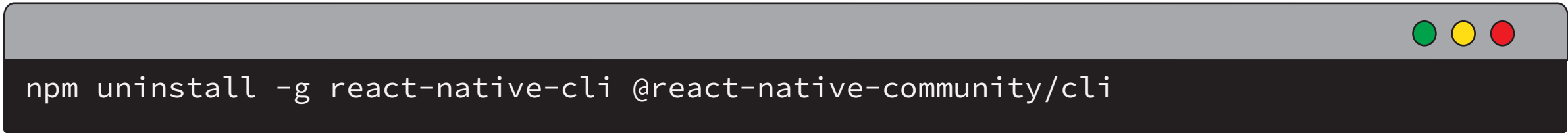
# Actividad guiada

## Parte 1

**Duración: 75 minutos.**

Abre alguna aplicación que venga por default en React native e implementa la nueva arquitectura:

1. Elimina el paquete react-native-cli con el siguiente comando:



```
npm uninstall -g react-native-cli @react-native-community/cli
```

2. Para Android:

- a. Cambia la propiedad newArchEnabled a verdadero:
  - i. Cambiando la línea correspondiente en android/gradle.properties
  - ii. Configuración de la variable de entorno `ORG_GRADLE_PROJECT_newArchEnabled=true`
- b. Compila y ejecuta la aplicación:



```
yarn Android
```



## Actividad guiada

### Parte 1

c. Verás el siguiente resultado:

```
      Welcome to Metro!  
    Fast – Scalable – Integrated  
  
To reload the app press "r"  
To open developer menu press "d"  
  
BUNDLE ./index.js  
  
LOG Running "MyApp" with {"fabric":true,"initialProps":null,  
"rootTag":1}
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora,  
para fines educativos.

## Actividad guiada

### Parte 1

3. Ahora, habilita el módulo turbo en Android:

a. Habilita el NDK y el constructor nativo:

i. Edita el archivo build.gradle para incluir 2 bloques externos nativos:

```
android {
    defaultConfig {
        applicationId "com.awesomeproject"
        // ...

        // Add this block
        externalNativeBuild {
            cmake {
                arguments "-DPROJECT_BUILD_DIR=$buildDir",
                    "-DREACT_ANDROID_DIR=$rootDir/../node_modules/react-native/ReactAndroid",
                    "-DREACT_ANDROID_BUILD_DIR=$rootDir/../node_modules/react-native/ReactAndroid/build",
                    "-DNODE_MODULES_DIR=$rootDir/../node_modules",
                    "-DANDROID_STL=c++_shared"
            }
        }
    }

    // Add this block
    externalNativeBuild {
        cmake {
            path "$projectDir/src/main/jni/CMakeLists.txt"
        }
    }
}
```

# Actividad guiada

## Parte 1

ii. En el mismo archivo, dentro de Android, {} debes añadir el siguiente código:

```
android {  
    // ...  
  
    def reactAndroidProjectDir = project(':ReactAndroid').projectDir  
    def packageReactNdkLibs = tasks.register("packageReactNdkLibs", Copy) {  
        dependsOn(":ReactAndroid:packageReactNdkLibsForBuck")  
        dependsOn("generateCodegenArtifactsFromSchema")  
        from("$reactAndroidProjectDir/src/main/jni/prebuilt/lib")  
        into("$buildDir/react-ndk/exported")  
    }  
  
    afterEvaluate {  
        preBuild.dependsOn(packageReactNdkLibs)  
        configureCMakeRelWithDebInfo.dependsOn(preReleaseBuild)  
        configureCMakeDebug.dependsOn(preDebugBuild)  
    }  
  
    packagingOptions {  
        pickFirst '**/libhermes.so'  
        pickFirst '**/libjsc.so'  
    }  
}
```

# Actividad guiada

## Parte 1

iii. Crea un archivo CMake dentro del folder src/main/jni llamado CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.13)

# Define the library name here.
project(myapplication_appmodules)

# This file includes all the necessary to let you build your application
with the New Architecture.
include(${REACT_ANDROID_DIR}/cmake-utils/ReactNative-application.cmake)
```

iv. Ejecuta la aplicación:

```
yarn react-native run-android
```



# Actividad guiada

## Parte 1

- b. Crea la subclase ReactPackageTurboModuleManagerDelegate:
  - i. Java:

```
package com.awesomeproject;

import com.facebook.jni.HybridData;
import com.facebook.react.ReactPackage;
import com.facebook.react.ReactPackageTurboModuleManagerDelegate;
import com.facebook.react.bridge.ReactApplicationContext;
import com.facebook.soloader.Soloader;

import java.util.List;

public class MyApplicationTurboModuleManagerDelegate extends ReactPackageTurboModuleManagerDelegate {

    private static volatile boolean sIsSoLibraryLoaded;

    protected MyApplicationTurboModuleManagerDelegate(ReactApplicationContext reactApplicationContext, List<ReactPackage> packages) {
        super(reactApplicationContext, packages);
    }

    protected native HybridData initHybrid();

    public static class Builder extends ReactPackageTurboModuleManagerDelegate.Builder {
        protected MyApplicationTurboModuleManagerDelegate build(
            ReactApplicationContext context, List<ReactPackage> packages) {
            return new MyApplicationTurboModuleManagerDelegate(context, packages);
        }
    }
}
```

# Actividad guiada

## Parte 1

```
}  
  
@Override  
protected synchronized void maybeLoadOtherSoLibraries() {  
    // Prevents issues with initializer interruptions.  
    if (!sIsSoLibraryLoaded) {  
        SoLoader.loadLibrary("myapplication_appmodules");  
        sIsSoLibraryLoaded = true;  
    }  
}  
}
```

ii. Kotlin:

```
package com.awesomeproject  
  
import com.facebook.jni.HybridData  
import com.facebook.react.ReactPackage  
import com.facebook.react.ReactPackageTurboModuleManagerDelegate  
import com.facebook.react.bridge.ReactApplicationContext  
import com.facebook.soloader.Soloader  
  
class MyApplicationTurboModuleManagerDelegate  
protected constructor(  
    reactApplicationContext: ReactApplicationContext,  
    packages: List<ReactPackage>  
) : ReactPackageTurboModuleManagerDelegate(reactApplicationContext, packages) {  
  
    override protected external fun initHybrid(): HybridData?  
    class Builder : ReactPackageTurboModuleManagerDelegate.Builder() {
```

# Actividad guiada

## Parte 1

```
        override protected fun build(
            context: ReactApplicationContext,
            packages: List<ReactPackage>
        ): MyApplicationTurboModuleManagerDelegate =
            MyApplicationTurboModuleManagerDelegate(context, packages)
    }

    @Synchronized
    override protected fun maybeLoadOtherSoLibraries() {
        // Prevents issues with initializer interruptions.
        if (!isSoLibraryLoaded) {
            SoLoader.loadLibrary("myapplication_appmodules")
            isSoLibraryLoaded = true
        }
    }

    companion object {
        @Volatile private var isSoLibraryLoaded = false
    }
}
```

# Actividad guiada

## Parte 1

- c. Adapta el host de ReactNative para usar ReactPackageTurboModuleManagerDelegate:
  - i. Java:

```
public class MyApplication extends Application implements ReactApplication {  
    private final ReactNativeHost mReactNativeHost =  
        new ReactNativeHost(this) {  
            @Override  
            public boolean getUseDeveloperSupport() { /* ... */ }  
  
            @Override  
            protected List<ReactPackage> getPackages() { /* ... */ }  
  
            @Override  
            protected String getJSMainModuleName() { /* ... */ }  
  
            @NonNull  
            @Override  
            protected ReactPackageTurboModuleManagerDelegate.Builder getRe-  
actPackageTurboModuleManagerDelegateBuilder() {  
                return new MyApplicationTurboModuleManagerDelegate.Build-  
er();  
            }  
        };  
}
```

# Actividad guiada

## Parte 1

ii. Kotlin:

```
class MyApplication : Application(), ReactApplication {
    private val reactNativeHost: ReactNativeHost =
        object : ReactNativeHost(this) {

            override fun getUseDeveloperSupport(): Boolean {
                /* ... */
            }

            override fun getPackages(): List<ReactPackage?>? {
                /* ... */
            }

            override fun getJSMainModuleName(): String? {
                /* ... */
            }

            @NonNull
            override fun getReactPackageTurboModuleManagerDelegateBuilder()
=
                ReactPackageTurboModuleManagerDelegate.Builder()
        }
}
```



# Actividad guiada

## Parte 1

- d. Extiende la función getPackages() del Host de react native para usar el módulo turbo:  
i. Java:

```
public class MyApplication extends Application implements ReactApplication {  
  
    private final ReactNativeHost mReactNativeHost =  
        new ReactNativeHost(this) {  
            @Override  
            public boolean getUseDeveloperSupport() { /* ... */ }  
  
            @Override  
            protected List<ReactPackage> getPackages() {  
                List<ReactPackage> packages = new PackageList(this).getPackages();  
  
                // Add those lines  
                packages.add(new TurboReactPackage() {  
                    @Nullable  
                    @Override  
                    public NativeModule getModule(String name, ReactApplicationContext reactContext) {  
                        if (name.equals(NativeAwesomeManager.NAME)) {  
                            return new NativeAwesomeManager(reactContext);  
                        } else {  
                            return null;  
                        }  
                    }  
                })  
            }  
  
            @Override  
            public ReactModuleInfoProvider getReactModuleInfoProvider() {  
                return () -> {  
                    final Map<String, ReactModuleInfo> moduleInfos = new HashMap<>();  
                    moduleInfos.put(  
                        NativeAwesomeManager.NAME,  
                        new ReactModuleInfo(  

```

# Actividad guiada

## Parte 1

```
NativeAwesomeManager.NAME,  
"NativeAwesomeManager",  
false, // canOverrideExistingModule  
false, // needsEagerInit  
true, // hasConstants  
false, // isCxxModule  
true // isTurboModule  
    )  
    );  
    return moduleInfos;  
};  
}  
});  
return packages;  
}  
  
@Override  
protected String getJSMainModuleName() { /* ... */ }  
  
@NonNull  
@Override  
protected ReactPackageTurboModuleManagerDelegate.Builder getReactPackageTurboModuleManager-  
DelegateBuilder() {  
    return new MyApplicationTurboModuleManagerDelegate.Builder();  
}  
};  
}
```

# Actividad guiada

## Parte 1

ii. Kotlin:

```
class MyApplication() : Application(), ReactApplication {

    private val reactNativeHost: ReactNativeHost =
        object : ReactNativeHost(this) {
            override fun getUseDeveloperSupport(): Boolean {
                /* ... */
            }

            override protected fun getPackages(): List<ReactPackage>? {
                val packages: MutableList<ReactPackage> = PackageList(this).getPackages()

                // Add those lines
                packages.add(
                    object : TurboReactPackage() {
                        @Nullable
                        override fun getModule(
                            name: String,
                            reactContext: ReactApplicationContext?
                        ): NativeModule? =
                            if ((name == NativeAwesomeManager.NAME)) {
                                NativeAwesomeManager(reactContext)
                            } else {
                                null
                            }
                    }
                )

                override fun getReactModuleInfoProvider() =
                    mutableMapOf<String, ReactModuleInfo>(
                        NativeAwesomeManager.NAME,
                        ReactModuleInfo(
                            NativeAwesomeManager.NAME,
```

# Actividad guiada

## Parte 1

```
        "NativeAwesomeManager",
        false, // canOverrideExistingModule
        false, // needsEagerInit
        true, // hasConstants
        false, // isCxxModule
        true // isTurboModule
    )
    )
    }
    )
    return packages
}

override protected fun getJSMainModuleName(): String? {
    /* ... */
}

@NonNull
override protected fun getReactPackageTurboModuleManagerDelegateBuilder():
    ReactPackageTurboModuleManagerDelegate.Builder? {
    return Builder()
}
}
```

# Actividad guiada

## Parte 1

### Consideraciones:

Debes:

1. Tener instalado React Native.
2. Haber instalado Android Studio.
3. Saber programar en JavaScript.
4. Realizar los ejercicios en orden.



# Receso

Duración: 10 minutos.

React Nivel Avanzado





# Actividad guiada

## Parte 2

**Duración: 75 minutos.**

4. Brinda una implementación nativa para los métodos TurboModuleDelegate:
  - a. MyApplicationTurboModuleManagerDelegate.h

```
#include <memory>
#include <string>

#include <ReactCommon/TurboModuleManagerDelegate.h>
#include <fbjni/fbjni.h>

namespace facebook {
namespace react {

class MyApplicationTurboModuleManagerDelegate : public jni::Hybrid-
Class<MyApplicationTurboModuleManagerDelegate, TurboModuleManagerDelegate>
{
public:
    // Adapt it to the package you used for your Java class.
    static constexpr auto kJavaDescriptor =
        "Lcom/awesomeproject/MyApplicationTurboModuleManagerDelegate;";

    static jni::local_ref<jhybriddata> initHybrid(jni::alias_
ref<jhybridobject>);

    static void registerNatives();

    std::shared_ptr<TurboModule> getTurboModule(const std::string name, const
std::shared_ptr<CallInvoker> jsInvoker) override;
    std::shared_ptr<TurboModule> getTurboModule(const std::string name, const
JavaTurboModule::InitParams &params) override;
```

## Actividad guiada Parte 2

```
private:
    friend HybridBase;
    using HybridBase::HybridBase;

};

} // namespace react
} // namespace Facebook
```

b. MyApplicationTurboModuleManagerDelegate.cpp:

```
#include "MyApplicationTurboModuleManagerDelegate.h"
#include "MyApplicationModuleProvider.h"

namespace facebook {
namespace react {

jni::local_ref<MyApplicationTurboModuleManagerDelegate::jhybriddata>
MyApplicationTurboModuleManagerDelegate::initHybrid(-
jni::alias_ref<jhybridobject>) {
    return makeCxxInstance();
}

void MyApplicationTurboModuleManagerDelegate::registerNatives() {
    registerHybrid({
        makeNativeMethod("initHybrid", MyApplicationTurboModuleManager
Delegate::initHybrid),
    });
}

std::shared_ptr<TurboModule> MyApplicationTurboModuleManagerDele-
```

## Actividad guiada Parte 2

```
gate::getTurboModule(const std::string name, const std::shared_ptr<
CallInvoker> jsInvoker) {
    // Not implemented yet: provide pure-C++ NativeModules here.
    return nullptr;
}

std::shared_ptr<TurboModule> MyApplicationTurboModuleManagerDele-
gate::getTurboModule(const std::string name, const JavaTurboModule::Init-
Params &params) {
    return MyApplicationModuleProvider(name, params);
}

} // namespace react
} // namespace Facebook
```

c. MyApplicationModuleProvider.h:

```
#pragma once

#include <memory>
#include <string>

#include <ReactCommon/JavaTurboModule.h>

namespace facebook {
namespace react {

std::shared_ptr<TurboModule> MyApplicationModuleProvider(const st-
d::string moduleName, const JavaTurboModule::InitParams &params);

} // namespace react
} // namespace Facebook
```

## Actividad guiada Parte 2

d. MyApplicationModuleProvider.cpp:

```
#include "MyApplicationModuleProvider.h"

#include <rncore.h>
#include <samplelibrary.h>

namespace facebook {
namespace react {

std::shared_ptr<TurboModule> MyApplicationModuleProvider(const std::string moduleName, const JavaTurboModule::InitParams &params) {
    auto module = samplelibrary_ModuleProvider(moduleName, params);
    if (module != nullptr) {
        return module;
    }

    return rncore_ModuleProvider(moduleName, params);
}

} // namespace react
} // namespace Facebook
```



## Actividad guiada Parte 2

e. OnLoad.cpp:

```
#include <fbjni/fbjni.h>
#include "MyApplicationTurboModuleManagerDelegate.h"

JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM *vm, void *) {
    return facebook::jni::initialize(vm, [] {
        facebook::react::MyApplicationTurboModuleManagerDelegate::re-
        gisterNatives();
    });
}
```

5. Habilita la bandera useTurboModules en el método onCreate:

f. Java:

```
public class MyApplication extends Application implements ReactApplica-
tion {

    @Override
    public void onCreate() {
        ReactFeatureFlags.useTurboModules = true;
        //...
    }
}
```

## Actividad guiada Parte 2

g. Kotlin:

```
class MyApplication : Application(), ReactApplication {  
    override fun onCreate() {  
        ReactFeatureFlags.useTurboModules = true  
        // ...  
    }  
}
```

6. Ejecuta tu aplicación:

```
yarn react-native run-android
```

# Actividad guiada

## Parte 2

React Nivel Avanzado

### Consideraciones:

Debes:

1. Tener instalado React Native.
2. Haber instalado Android Studio.
3. Saber programar en JavaScript.
4. Realizar los ejercicios en orden.

## Cierre

**Duración: 10 minutos.**

Conforme va evolucionando la tecnología, el software también evoluciona, ya que, si no se actualiza, se volverá obsoleto y los desarrolladores, así como los usuarios, ya no querrán nada que tenga que ver con dicho software o aplicación.

La nueva arquitectura de React Native te permitirá que esto no ocurra y que tu desarrollo de las aplicaciones se vuelva más sencillo.

La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor.

El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.