El caso judicial

Oracle America, INC

contra

Google, LLC

No. C 10-03561 WHA

El caso judicial ***Oracle America, Inc. c. Google, LLC.***
No. C 10-03561 WHA

CONTENIDO

Estructura, secuencia y organización

No importa que tan creativo o imaginativo pueda ser un método de Java, cualquiera tiene derecho a utilizar la misma especificación del método (inputs, outputs, parámetros) siempre y cuando y línea por línea, el código fuente implementado sea diferente

Los métodos de Java como "métodos de operación" podrían ser protegidos bajo el sistema de patentes de invención

Doctrina de la fusión ("*merge doctrine*")

Interoperabilidad

Fragmentación (o "*balkanization*"). Los casos SONY y SEGA

Razones por las cuales el caso *American Dental Association (ADA) v. Delta Dental Plans Association* (decisión judicial del Circuito Séptimo) no controla el presente caso judicial

Porque no es correcta la visión de Oracle respecto del caso *Johnson Controls v. Phoenix Control Sys* (Circuito Noveno, del año 1989)

Porcentajes

Elementos particulares del caso

CAPITULO VI. RECURSO DE APELACIÓN INTERPUESTO POR ORACLE CONTRA LA

PROLOGO

El objetivo del presente trabajo es narrar los hechos a través de los dichos expresados por las partes litigantes y los tribunales intervinientes durante el juicio de manera tal que los lectores puedan recorrer de forma simple y suficiente los hechos expuestos por cada una de las partes en las distintas presentaciones judiciales, como también los fundamentos brindados por los jueces en sus resoluciones judiciales tanto en su etapa de protección de derechos de autor como en la etapa del juicio referida al análisis de uso justo ("*fair use"*).

El presente caso judicial constituye uno de los litigios extranjeros más complejos, extensos y relevantes en materia de programas de computación que ha existido hasta la fecha a nivel global en la industria del software, no sólo por el objeto novedoso bajo debate, sino también por la complejidad técnica y jurídica que plantea y sus diferentes puntos de interconexión.

Por lo tanto, no es objetivo del presente trabajo emitir una conclusión personal del autor al respecto de si lo debatido en el juicio corresponde o no estar protegido por un sistema de derechos de autor.

Muy por el contrario, lo que se intenta ofrecer al lector no es una conclusión *propiamente dicha,* sino que se ofrece un listado de preguntas "*no exhaustivas, objetivas, como de mínima y obligatorias"* con la finalidad de ayudar a buscar en cada lector su propia conclusión.

Por otra parte, tales preguntas están formuladas con independencia a las diferentes posibles jurisdicciones y leyes aplicables de derechos de autor de cada uno de los lectores, de

manera tal, que una vez finalizada la lectura de este trabajo las preguntas colaboraran a cada lector a obtener una conclusión propia en relación a lo debatido en este pleito y en su caso de cuales podrían ser sus efectos bajo su jurisdicción.

Por lo cual sin importar la jurisdicción y ley aplicable del lector las hipótesis de partida de las preguntas son válidas para colaborar con los lectores a elaborar su propia conclusión.

Gustavo G. Mármol Alioto[1].

[1] Abogado, 1998, Buenos Aires, Argentina. Actualmente se desempeña como abogado consultor especialista en negociación internacional, privacidad, licenciamiento de software binario y *open-source*, firmware, sistemas operativos, comercialización de hardware, sistemas de almacenamiento, cumplimiento y gobierno corporativo, y litigios. También es investigador independiente en asuntos relacionados a estándares abiertos e interoperabilidad. Anteriormente fue director de legales, cumplimiento y privacidad de la subsidiaria de Oracle Chile en Santiago de Chile desde enero de 2014 a septiembre de 2017. También fue responsable del área privacidad y seguridad de Oracle para América Latina. Asimismo, fue designado responsable legal de la unidad de negocios *open-source* de Oracle llamada Linux, MySQL y virtualización para Latinoamérica. Anteriormente desde agosto de 2010 a diciembre de 2013 fue abogado senior en la subsidiaria de Oracle Argentina en la ciudad de Buenos Aires. También, desde enero de 2007 a agosto de 2010 fue abogado corporativo y de privacidad de la subsidiaria de Sun Microsystems en Argentina, cumpliendo funciones regionales para América del Sur y países de Centro América y Caribe. Sun Microsystems fue adquirida por Oracle Corporación en enero de 2010. Con anterioridad a ello, desde 1999 hasta el año 2006 se desempeñó como abogado asesor y litigante habiendo representado a clientes nacionales e internacionales en diferentes industrias, entre ellas de software, hardware, cinematografía, televisión por cable, televisión satelital, gráfica, indumentaria, aviación, química, gastronomía & entretenimiento, de pedidos de informes, agro y financiero.

CAPITULO I

PRELIMINAR

Desde los mismos escritos judiciales presentados por las partes, y desde las mismas resoluciones judiciales emitidas en el pleito en las diferentes instancias procesales**,** se narra el pleito iniciado en los EE.UU. en agosto de 2010 por parte de la empresa *Oracle America, Inc*. contra la empresa *Google LLC*[2].

Se recorren los argumentos expuestos por las partes en sus escritos de inicio y contestación de demanda, la sentencia de la Corte de Distrito del Norte de California, las sentencias del Tribunal de Apelaciones del Circuito Federal del año 2014 y 2018, las distintas apelaciones interpuestas por las partes, y finalmente se narran los argumentos expuestos por la Corte Suprema de los EE.UU. tanto los del voto de la mayoría, como los de la minoría.

El litigio refiere a la infracción de derechos de autor y patentes de invención, y respecto de lo primero se trata de:

> *(a)* un *tipo* de código fuente denominado en el trámite del juicio como "código fuente declarado" (en inglés *"declaring code")* o "archivos de cabecera" (en inglés "*header files*") o "declaraciones" (en inglés *"declarations"*) de 37 paquetes de la librería de software estándar de Java (en inglés *Java standard software library*)

[2] Google cambió su denominación social en el transcurso del juicio de Google INC (así es la denominación que poseía a la fecha de la demanda de Oracle) a Google, LLC.

o librería de clase (en inglés "*library class"*) denominados en el juicio como las API de Java,

(b) de la estructura, secuencia y organización de las API de Java (en inglés *"sequence, structure and organization"* o su sigla de "*SSO"*).

En efecto, el pasado 5 de abril del 2021 en una votación dividida de 6 votos contra 2 la Corte Suprema de Justicia de los Estados Unidos de América, dictó sentencia a favor de la empresa Google aplicando la teoría del "uso justo" o legitimo (en inglés *"fair use"*).

Para decidir la cuestión, la Corte Suprema de EE.UU. entendió que:

(i) no debía resolver si las API de Java y su estructura, secuencia y organización era protegible conforme al sistema de derechos de autor de los EE.UU. (sistema del "Copyright"), por lo tanto, *no modificó* el fallo dictado por el Tribunal de Apelaciones del Circuito Federal (en su sentencia I de mayo 9 del 2014) la cual había reconocido la protección bajo la ley de derecho de autor de EE.UU. de las API de Java y de su estructura, secuencia y organización;

(ii) Por el contrario, la Corte Suprema decidió tratar *sólo aquello que consideró como necesario para resolver la disputa* entre las partes, y así *revirtió el fallo* dictado por el Tribunal de Apelaciones del Circuito Federal (sentencia Nro. II de marzo de 2018) referida a la limitación del derecho de autor conocido como uso justo o legítimo (en inglés *"fair use").*

Al respecto, la Corte Suprema dijo en su conclusión que la utilización de las API y su estructura, secuencia y organización por parte de Google había sido “justo o legítimo”.

Ello ya que Google en la *reimplementación* de una interface de usuario *sólo había tomado aquello que era necesario para permitir a los usuarios poner sus talentos adquiridos a trabajar en un nuevo programa*.

CAPITULO II

LA TECNOLOGÍA JAVA

Temario. *La plataforma Java y el lenguaje de programación Java. La librería de software estándar de Java. Los paquetes, clases y métodos de Java. Diagrama de una API de JAVA de Sun Microsystems (según la Corte Suprema de EE.UU. en su sentencia de abril 5 de 2021)*

LA PLATAFORMA JAVA Y EL LENGUAJE DE PROGRAMACIÓN JAVA[3]

Java, por un lado, constituye una plataforma de software que permite ejecutar programas de software escritos en el lenguaje de programación Java[4].

[3] Uno de los puntos que sobresalieron durante el trámite del pleito fue si la librería de software estándar de Java, que es uno de los componentes de la plataforma Java (más específicamente de lo que se conoce como ambiente de ejecución o en inglés "*run-time environment"*) formaba parte o no del lenguaje de programación Java, y a su vez, si la implementación de un lenguaje de programación debía ser protegido como un programa de computación bajo las leyes de derechos de autor de EE.UU. Las partes no estuvieron de acuerdo al respecto.

[4] Podría decirse que cada lenguaje de programación posee su propia sintaxis. Así, el lenguaje de programación Java posee su propia sintaxis. La sintaxis constituye las "reglas" establecidas que deben seguirse, ya que de cualquier otra forma se producirían errores o simplemente lo escrito en código fuente en dicho lenguaje no produciría ningún resultado. Podría asimilárselo a reglas gramaticales. La diferencia radica en que un compilador (que es otro programa de computación) sólo podría entender una sintaxis correcta del código escrito, por ello si la sintaxis no es correcta el compilador -o el

Y, por otro lado, Java es el lenguaje de programación mediante el cual un programador escribe código fuente [5] en lenguaje Java.

La plataforma Java[6] posee varios componentes, que son los siguientes:

(i) herramientas para asistir en el desarrollo de código fuente escrito en el lenguaje de programación Java (en inglés "*Java development kit"* o "JDK"),

(ii) el compilador de Java (en inglés "*javac compiler"*) que convierte declaraciones en lenguaje de programación Java en declaraciones de código byte de Java,

(iii) el ambiente de ejecución de Java (en inglés "*runtime environment"*) que consiste en:

intérprete dependiendo del tipo de lenguaje implementado- no sería capaz de entender lo escrito, y, por tanto, o produciría un error o no arrojaría resultado alguno. Entonces, la sintaxis de Java posee varios elementos como ser (i) separadores (por ejemplo {;}, ;) (ii) palabras claves (por ejemplo: *if*, *else*, *while*, *return*), (iii) valores literales (por ejemplo, ´x¨, Foo), (iv) operadores (por ejemplo *, /, +, -) etc. Entonces cuando se vean "estos símbolos" en el presente trabajo se trata de la sintaxis del lenguaje de programación Java.

[5] Código fuente constituye la versión del programa de computación que es legible para un ser humano.

[6] Véase en "Brief of former Sun Microsystems executive Scott McNealy as *amicus curiae* in support of Oracle. *How Java Works*. https://www.supremecourt.gov/DocketPDF/18/18956/133505/20200219153012820_Brief.pdf

(a) la máquina virtual de Java (en inglés "*Java virtual machine"* o *"JVM*[7]) diseñada para operar en distinto tipo de computadoras y sistemas operativos, y

(b) la librería de software estándar de Java o librería de clases de Java *(Java standard library* o *class library* o también denominada como *"Java packages"* o *"API packages"* en los escritos de las partes y resoluciones judiciales*)*, y, por último

(iv) la documentación para cada una de las ediciones de la plataforma Java.

EL PRINCIPIO "WRITE ONCE, RUN ANYWHERE" (WORA)

Sun Microsystems, Inc [8] desarrolló la plataforma Java Standard Edition ("Java SE") juntamente con otras ediciones de la plataforma Java[9].

[7] The Java® Virtual Machine Specification Java SE 14 Edition. February 2020. https://docs.oracle.com/javase/specs/jvms/se14/jvms14.pdf

[8] Sun Microsystems, Inc. fue adquirida por Oracle Corporation en el mes de enero del año 2010, y por ende Oracle adquirió todos los derechos que su antecesor Sun Microsystems poseía sobre Java.

[9] Por su parte, cada una de estas ediciones de la plataforma Java contiene su ambiente de desarrollo JDK, su compilador Java, su máquina virtual Java, su set de librerías de software estándar de Java, y la documentación describiendo las funcionalidades y operación específica de cada una de las ediciones de la plataforma Java. Si bien existen similitudes en cada una de estas ediciones de Java, una de las diferencias entre ellas es que cada edición de la plataforma Java posee una librería de software estándar diferente, y ello responde al tipo de aplicaciones y de ambientes por el cual cada edición es enfocada.

Su primera edición se publicó en el año 1996.

El objetivo de Sun Microsystems era que Java debía ser *agnóstico* a la tecnología sobre la cual se ejecutara el programa de computación escrito en Java, de manera tal que los programadores no tendrían que escribir diferentes versiones de programas de computación para diferentes sistemas operativos o hardware.

La máquina virtual de Java posee un rol principal en la plataforma Java.

Como se expuso anteriormente[10] el lenguaje de programación Java -posee palabras, símbolos, más ciertas reglas sintácticas y semánticas que deben respetarse para crear instrucciones- es el lenguaje mediante el cual los programadores escriben código fuente, que es la versión de un programa de computación comprensible a los seres humanos.

Ahora bien, para que tales instrucciones creadas puedan ser ejecutadas o llevarse a cabo, deben convertirse (o compilarse) en código objeto, que consisten en 0 y 1 entendibles para una computadora en particular.

En Java, conforme a sus reglas, el código fuente primero se convierte en lo que se denomina código de bytes (o en inglés "*bytecode"*) que es un estadio intermedio antes de que se convierta en código objeto por la máquina virtual de Java.

Entonces, la plataforma Java a través de la utilización de la máquina virtual de Java permite a los desarrolladores escribir programas que son capaces de ejecutarse en diferentes

[10] Ver referencia nro. 4

computadoras o hardware sin tener que reescribir [11] sus programas para cada tipo diferente de computadora.

Esto se conoció como el *"write once, run anywhere"* (WORA).

En definitiva, un programa escrito en Java podría correr o ejecutarse en cualquier sistema operativo o hardware, siempre que tuviese instalado la máquina virtual de Java.

LA LIBRERÍA DE SOFTWARE ESTÁNDAR DE JAVA [12]. LOS PAQUETES, CLASES Y MÉTODOS DE JAVA[13]

La librería de software estándar de Java contiene programas de computación "*pre-escritos*[14]" (denominados o conocidos en

[11] Véase Real Academia Española: 1. tr. Volver a escribir lo ya escrito introduciendo cambios. 2. tr. Volver a escribir sobre algo dándole una nueva interpretación. https://dle.rae.es/reescribir

[12] De todos los elementos que conforman la plataforma Java, el juicio entre las partes se refiere únicamente a aspectos de la librería de software estándar de Java o librería de clases.

[13] Véase por ejemplo la versión 7 en https://docs.oracle.com/javase/7/docs/api/

[14] El concepto de *"pre-escritos"* cuando se refiere a programas de computación implica que la funcionalidad ya fue desarrollada o escrita, en este caso lo fue por Sun Microsystems, y que dicha funcionalidad se encuentra disponible para su utilización, como parte de una librería de software, en el caso Java. Normalmente una librería de software contiene una serie de rutinas y subrutinas pre-compiladas disponibles para que los programadores pueden reutilizar mediante técnicas, dependiendo el lenguaje, de vínculos dinámicos o estáticos, para así de esta forma incorporarlas o importarlas a su programa en construcción.

inglés como "*buit in packages"* [15]) los cuales pueden ser utilizados o no[16] por los programadores para escribir sus propios programas de computación.

---

[15] Existen dos tipos de paquetes en Java: Los llamados a) *built-in packages* que son los paquetes *pre-escritos* que vienen incluidos en la librería de clases de Java y, los conocidos como b) *user-defined packages* que son los paquetes que pueden ser creados o escritos por los propios programadores.

[16] Conforme a las constancias del juicio, Google argumentó durante el trámite del pleito que ciertos paquetes de Java eran necesarios para utilizar el lenguaje de programación Java, y que sin ellos el lenguaje Java fallaría. Esto fue usado por Google para reforzar su teoría de que los paquetes de Java eran parte del lenguaje de programación Java. Oracle lo negó, y conforme palabras del Juez William Alsup, Oracle reconoció que sólo ciertos paquetes de Java eran necesarios para el correcto funcionamiento del lenguaje Java. Los paquetes reconocidos en el juicio por Oracle como necesarios son tres (3), los cuales incluyen unas 62 clases aproximadamente. Estos son: **java.lang**, **java.io**, y **java.util**. Así se dijo por el juez de la Corte del Distrito de California, William Alsup que "*The original Java Standard Edition Platform ("Java SE") included "eight packages of pre-written programs." The district court found, and Oracle concedes to some extent, that three of those packages—java.lang, java.io, and java.util—were "core" packages, meaning that programmers using the Java language had to use them "in order to make any worthwhile use of the language."* Véase que en la versión inicial de Java SE (1996) existían 8 paquetes de Java, y en el año 2008 dos años antes de iniciarse el litigio había 166 paquetes, con más de 600 clases agrupadas, y más de 6000 métodos. Esto muestra de alguna forma la evolución de Java SE. De estos 166 paquetes de Java sólo 37 paquetes (de la edición de Java2SE 5.0) son sobre los que Oracle expresa que Google habría infringido sus derechos. Estos paquetes son: java.awt.font, java.beans, java.io, java.lang, java.lang.annotation, java.lang.ref, java.lang.reflect, java.net, java.nio, java.nio.channels, java.nio.channels.spi, java.nio.charset, java.nio.charset.spi, java.security, java.security.acl, java.security.cert, java.security.interfaces, java.security.spec, java.sql, java.text, java.util, java.util.jar, java.util.logging, java.util.prefs, java.util.regex, java.util.zip, javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, javax.net.ssl, javax.security.auth, javax. security.auth.callback, javax.security.auth.login, javax.security.auth.x500, javax.security.cert, and javax.sql. Como se verá oportunamente, en el "segundo juicio" que se refiere

A su vez, la librería de software estándar de Java posee una estructura de organización que es la siguiente:

i) paquetes (en inglés "*packages"*)[17];

---

al análisis de uso justo o legítimo que mando a llevar adelante el Tribunal de Apelaciones del Circuito Federal, en su sentencia del 27 de marzo de 2018 se dictaminó basado en las prueba producida que sólo 170 líneas de código de dichos tres paquetes de Java (**java.lang, java.io y java.util**) eran obligatorias de ser usadas en el uso del lenguaje de programación Java. Al mes de mayo del 2021 la última versión es la Java SE 16 véase en https://www.oracle.com/java/technologies/javase-downloads.html.

[17] Véase en https://docs.oracle.com/javase/8/docs/api/ que contiene la Especificación de la API para la Plataforma Java Standard Edition 8 (SE) donde se listan los paquetes de Java, todas las clases y sus métodos. Como ejemplo entre los *paquetes* se encuentran: java.applet; java.awt, java.awt.color; java.awt.datatransfer; java.awt.dnd; java.awt.event, java.awt.font, java.awt.geom, java.awt.im, java.awt.im.spi, java.awt.image, java.awt.image.renderable java.awt.print, java.beans, java.beans.beancontext, java.io, *java.lang*, java.lang.annotation, java.lang.instrument, java.lang.invoke, java.lang.management, java.lang.ref, java.lang.reflect, java.math, java.net, java.nio, java.util, java.util.concurrent, java.util.concurrent.atomic. A su vez, dentro del paquete *java.lang* se encuentran las siguientes *clases*: Boolean, Byte, Character, Character.Subset, Character.UnicodeBlock, Class, ClassLoader, ClassValue, Compiler, Double, Enum, Float, InheritableThreadLocal, Integer,Long, *Math*, Number, Object, Package,Process, ProcessBuilder, ProcessBuilder.Redirect, Runtime, RuntimePermission, SecurityManager, Short, StackTraceElement, StrictMath, String, StringBuffer, StringBuilder, System, Thread, ThreadGroup, ThreadLocal, Throwable, Void. A su vez, los *métodos* incorporados y declarados dentro de la clase *Math* son entre ellos: abs(double a), cbrt(double a), cos(double a) y *max*(int a, int b). Así el método *cos* devuelve el valor trigonométrico de un ángulo y el método *max* devuelve el mayor valor de dos números. Así los parámetros del método *max* son: *"Parameters: a - an argument. b - another argument. Returns: the larger of a and b"*. El paquete de Java *Java.lang*, la clase de Java *Math* y el método de Java *max* incluídos en la versión de la Plafaforma Java SE en el pleito ha sido citado como ejemplo para ilustrarse lo que constituiría la declaración e implementación del método de Java *java.lang.Math.max* y la invocación o llamada a este método declarado, por parte del Juez de la Corte de Distrito de

ii) clases[18] (en inglés "*classes"*); y
iii) métodos (en inglés "*methods"*).

Así, los paquetes están conformados por clases, y las clases se conforman por métodos.

Dicho de otra forma, los paquetes *agrupan* a las clases, y las clases *agrupan* a los métodos.

Este conjunto organizado de programas de computación *pre-escritos* (en inglés *"pre-written programs"* una forma habitual de denominar a los programas o subrutinas que constituyen o conforman las librerías de software en general) se lo ha denominado en el juicio[19], individualmente como *aplicación de programación de interface* o API, o colectivamente, como paquetes de API de Java.

California, por del Tribunal de Apelaciones del Circuito Federal, por el Abogado General de EE.UU. y por la Corte Suprema de Justicia de los EE.UU.

[18] Esta es la razón por la cual a la librería de software de Java también se la denomina librería de clases.

[19] Oracle America, Inc., (Plaintiff) v. Google Inc. (Defendant). May, 31, 2012. No. C 10-03561 WHA, US District Court for The Northern District of California. Order RE Copyrightability of Certain Replicated Elements of the Java Application Programming Interface. "*The Java language itself is composed of keywords and other symbols and a set of pre-written programs to carry out various commands such as printing something on the screen or retrieving the cosine of an angle. The set of pre-written programs is called the application programming interface or simply API (also known as class libraries)"* Véase en https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/1202/0.pdf?ts=1376380463

Ahora bien, durante el litigio el significado de la terminología API *no ha sido precisa*, y esto ha sido reconocido por el Juez de la Corte de Distrito.

Así, una de las partes (Oracle) ha mencionado que el término API es un camaleón, ya que puede significar varias cosas, algunas simples y otras muy complejas[20].

Más allá de ello, en el juicio, las partes litigantes, los *amicis curiae*, y los magistrados intervinientes en las distintas instancias del proceso se han referido a API como:

(i) al conjunto de los 37 paquetes de Java;

(ii) a un paquete de Java en particular;

(iii) a las "declaraciones de las API de Java" por cuanto las API sólo estarían conformadas por las declaraciones de código (y no así por el código implementado), y

(iv) interfaces de software (Google y sus *amicis curiae*).

[20] Véase en el escrito *Brief of Amici* presentado ante la Tribunal de Apelación del Circuito Federal (al respecto de la sentencia dictada por el Juez de Distrito del Norte de California) por Microsoft Corporation, EMC Corporation y NetAPP, INC a favor de la parte apelante (Oracle) con fecha 19 de febrero de 2013 "BRIEF FOR AMICI CURIAE MICROSOFT CORPORATION, EMC CORPORATION, AND NET APP, INC. IN SUPPORT O F APPELLANT, en la página 7 nota 3 se expresa textualmente lo siguiente: "*As Oracle´s opening brief explains (at 21), the terminology used in the case is confusing. The Java software packages at issue here are called "Application Programming Interfaces", or "API". The term API is used in the software industry to describe a wide range of things, some of which are very simple and some of which are very complex, and each of which has different purposes and context. To provide clarity, amici refer to the computer programs here as "software packages" or "platforms". Amici do not address APIs beyond the computer programs at issue here*". http://www.groklaw.net/pdf4/13-1021-70.pdf

Para Google, y para varios de los *Amici Curiae* que han presentado opinión en el juicio a su favor, las API constituyen sólo el código fuente declarado (o en inglés "*method headers")*[21], por ende, conforme su postura, sería redundante afirmar API y código declarado o declaraciones de las API de Java, ya que constituye lo mismo.

Véase en la referencia de más abajo el código fuente declarado y el código implementado.

En definitiva, la disputa judicial entre las partes se centra en:

(i) el "código fuente declarado" (en inglés "*declaring code"*) o "declaraciones" (en inglés "*declarations"*) o "métodos de encabezados" (en inglés "*method headers"*),

Y no en:

(ii) el "código fuente implementado" de las API de Java (por Sun Microsystems, el cual no fue copiado por Google, habiendo realizado Google su propia implementación para Android).

Continuando, los "métodos de Java" permiten a los desarrolladores construir otros programas de computación[22].

[21] Véase la presentación realizada "*Brief amici curiae of Eighty-Three Computer Scientists"* de enero de 2020 https://www.supremecourt.gov/DocketPDF/18/18-956/128391/20200113145027664_18-956%20Google%20v%20Oracle%20Computer%20Scientists%20Merits%20Amicus%20FOR%20FILING.pdf

[22] Este procedimiento de reutilización de código es muy común en la construcción de programas, y cada lenguaje de programación implementado suele poseer una librería estándar. Esta posibilidad, de poder utilizar subprogramas *pre-escritos* para poder desarrollar o construir otros programas

Los métodos de Java son una suerte de bloques de software que sólo pueden ejecutarse cuando son "llamados o invocados".

Estas llamadas o invocación a los métodos de Java se efectúa bajo un *"formato de comando"* determinado (lo que significa que si no se utiliza dicho formato dado la invocación al código declarado no se puede producir).

La utilización de los "métodos de Java" tiene por finalidad la "reutilización de código" *pre-escrito* y disponible para ser utilizado por los programadores en la construcción de sus programas o aplicaciones con el lenguaje de programación Java.

Es decir que una vez *declarado* el método de Java puede ser utilizado infinidad de veces.

Para ilustrar el código fuente (el declarado y el implementado) de un método, clases, paquete de Java *(esquema 1)* y su invocación -a través de un comando con su respectivo formato *(esquema 2)*- se hace referencia al mismo ejemplo citado por el Juez de la Corte de Distrito del Norte de California, como sigue:

ESQUEMA 1 **CÓDIGO FUENTE (ESCRITO EN LENGUAJE JAVA) DECLARADO E IMPLEMENTADO DE UN "MÉTODO DE JAVA"**

---

no es sólo una característica del lenguaje de programación Java, sino también de cualquier lenguaje de programación. Así en el lenguaje de programación C tenemos la librería estándar de C *.libc* que constituye la implementación de referencia de la especificación del lenguaje C conforme a la ISO C estándar https://en.wikipedia.org/wiki/C_standard_library, como también la implementación de software libre y de código abierto (FOSS) llamada *glibc* de GNU C Library (https://es.wikipedia.org/wiki/Glibc) En términos prácticos no haría sentido escribir código que esté disponible por la propia librería estándar del propio lenguaje de programación.

```
package java.lang;                    (declaración del paquete java.lang)
public class Math {                   (declaración de la clase Math)
public static int max (int x, int y) {    (declaración del método max)
if (x > y) return y;                  (implementación, devuelve x, o)
else return y;                        (implementación, devuelve y)
 }                                                (cierra el método)
}                                                 (cierra la clase)
```

En la declaración del método de Java ***max*** se puede ver que se comienza con la palabra ***public*** que significa que otros programas o métodos de Java fuera de la clase pueden llamar a ese método, ya que si en su lugar figurase ***private*** significaría que el método sólo estaría disponible para otros métodos agrupados dentro de la misma clase (pero no fuera de ella).

La palabra ***static*** significa que el método puede ser invocado sin la necesidad de tener que crearse una instancia de clase.

La palabra ***int*** significa que un número entero es devuelto por el método.

Por su parte ***max*** es el nombre del método.

Y finalmente **(*int x, int y*)** son los argumentos que necesariamente deben pasar por el método, estableciéndose que será en forma de número entero.

Por último, el marcador ***{*** es obligatorio.

La descripción del ejemplo obedece a las reglas propias del lenguaje de programación Java.

Por otra parte, para poder usarse dicho método de Java (***max***) por un programador en el desarrollo de su propio programa o aplicación, el programador debe *invocar* o *llamar* este método de Java (***max***) desde su programa en construcción.

Así, el programa del desarrollador deberá incluir mediante una llamada o invocación al método de Java bajo el formato de comando ya determinado.

ESQUEMA 2. **LLAMADA O INVOCACIÓN DEL MÉTODO JAVA BAJO EL FORMATO DE COMANDO DETERMINADO**.

int a = java.lang.Math.max (2,3)

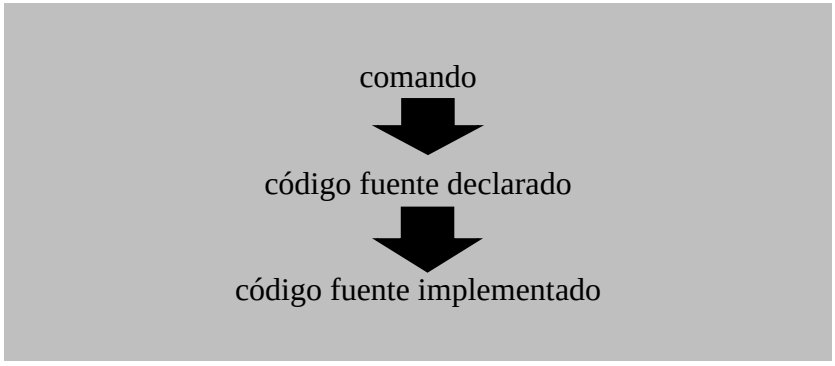
El programador al escribir el código fuente de la invocación o llamada al método de Java debe hacerlo mediante un formato de comando determinado, como sigue ***java.lang.Math.max ()*** el cual respeta además un determinado orden de posición, a saber: paquete ***java.lang***, clase ***Math***, método ***max***.

Con lo cual, para poder reutilizar el método de Java ***Max*** (el cual constituye la rutina o funcionalidad de la cual necesita y se quiere servir), el programador debe invocar o hacer un llamamiento al código fuente declarado del método de Java ***Max*** (ver *Esquema 1*) a través del formato comando determinado (ver *Esquema 2*).

Entonces estos símbolos y nombres, que conforman el formato de comando **java.lang.Math.max()**, *"por sí solos*" no hacen o pueden ejecutar *ninguna* acción.

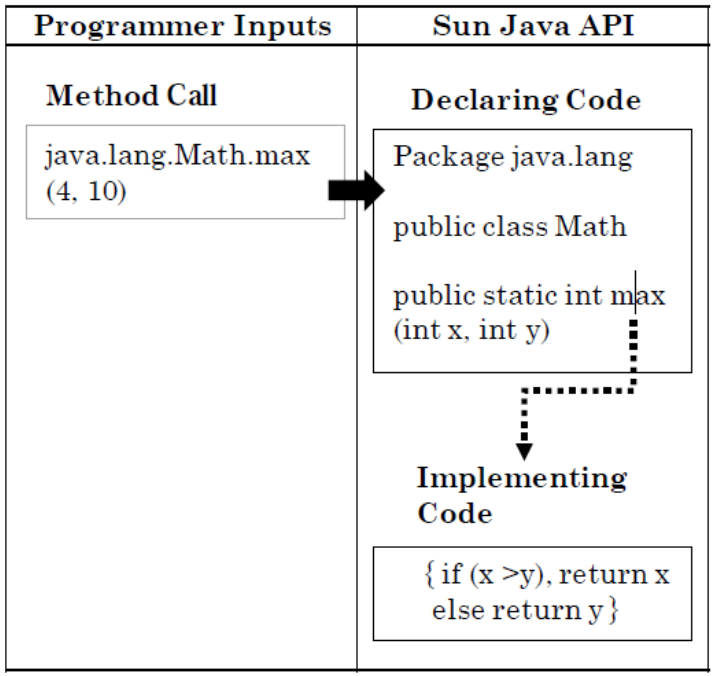
Para ello, es necesario que, mediante dicho formato de comando determinado se produzca la llamada o invocación al código fuente declarado del método de Java ***max***, para que éste, a su vez, luego invoque el código fuente implementado del método de Java ***Max*** (de esta forma el programador tendría la funcionalidad deseada -sin necesidad de tener que reescribir desde cero- incluida en su propio desarrollo o app).

Así, entonces mediante la llamada del código fuente declarado a través del formato de *comando* determinado se invoca al código fuente implementado del método de Java.



Sin el código fuente declarado del método de Java, la llamada al método de Java a través del correspondiente formato determinado, no podría producir la invocación al código fuente implementado, y por ende no podría llevarse a cabo ninguna actividad (es decir no se ejecutaría ninguna función).

DIAGRAMA DE UNA API DE JAVA DE SUN MICROSYSTEMS (SEGÚN LA CORTE SUPREMA DE EE.UU. EN SU SENTENCIA DE ABRIL 5 DE 2021)



Habiéndose realizado estas menciones veremos los dichos expresados por cada una de las partes.

Se comenzará con los expuestos por la parte demandante, Oracle America, INC, para seguir con los de la demandada Google, LLC.

CAPITULO III

LA POSICIÓN DE ORACLE[23]

Temario. *Los paquetes -programas de computación- pre-escritos de Java. Licenciamiento de la plataforma Java SE. La especificación de la plataforma Java SE. Google copió en forma idéntica el código fuente declarado de 37 paquetes de Java SE para introducirlo en la plataforma Android.*

Con fecha 12 de agosto de 2010 Oracle demandó a Google por infracción a sus derechos de autor y patentes de invención[24].

En efecto, Oracle argumentó que Android (incluyendo la máquina virtual *"Dalvik"* y el "*Android Development Kit"*) infringía los derechos de Oracle en relación a su tecnología Java.

En relación a los derechos de autor, Oracle sostuvo que la plataforma Java contiene una cantidad sustancial de material original (como por ejemplo y sin quedar limitado a código, especificaciones, documentación y otros materiales) protegible conforme al sistema de derecho de autor de los EE.UU. ("*Copyright Law Act*, 17 U.S.C. 101").

Que, sin consentimiento, autorización, aprobación o licencia alguna de parte de Oracle, Google violando la ley copió, preparó, publicó y distribuyó material protegido por las leyes de derechos

[23] Véase https://www.oracle.com/index.html

[24] Las patentes mencionadas por Oracle en su demanda fueron: Nos. 6,125,447; 6,192,476; 5,966,702; 7,426,720; RE38,104; 6,910,205; y 6,061,520.

de autor de EE.UU. propiedad de Oracle, o porciones de ellas u obras derivadas de ella.

Oracle sostuvo que los usuarios de Android, los cuales incluye a los fabricantes de dispositivos (por sus siglas OEM en inglés "*Original Equipment Manufacturer"*), deben obtener y utilizar porciones de la plataforma Java u obras derivadas de Java, para fabricar y hacer funcionar los dispositivos Android.

Oracle expresa que tal uso no se encuentra autorizado por Oracle.

Por lo tanto, Oracle agrega que Google ha inducido, causado y materialmente contribuido a los actos de infracción por parte de terceros, sea mediante su inducción, asistencia a otros, para usar, copiar y distribuir material protegido propiedad de Oracle, o material derivado de ellos.

Oracle expresa que Java[25] fue desarrollada originalmente por Sun Microsystems, la cual fue adquirida por Oracle Corporation en enero del año 2010, y posteriormente renombrada como Oracle America, Inc.

Java fue diseñada para lograr el objetivo de *"write once, run anywhere"* en un nuevo entorno informático de redes que incluía internet.

La idea básica detrás de Java es que un desarrollador de software pudiera escribir aplicaciones de software una vez, luego compilarlo en un formato intermedio conocido como "código de

[25] Al 17 de marzo del 2011 Oracle estimó que la plataforma Java había atraído a más de 6.5 millones de desarrolladores de software, más de 1.1 billón de computadoras de escritorio y 3 billones de teléfonos móviles corrían o ejecutaban Java.

bytes de java", y distribuirlo en forma de archivos de clases (en inglés *"class files"*) o archivos .jar *(*en inglés *"files jar").*

Entonces un usuario que quisiese correr la aplicación podría obtener una copia del código byte de la aplicación a través de varios mecanismos, incluyendo su descarga desde la red internet.

Entonces el usuario podría correr el código usando la máquina virtual de Java (en inglés "*Java virtual machine"* o por sus siglas "JVM") que fue escrita para la arquitectura particular de la computadora del usuario.

La aplicación podría correr en cualquier computadora que tuviese instalado la máquina virtual de Java.

La consistencia entre la implementación de la máquina virtual de Java de cada plataforma es lograda porque cada una de ellas está en cumplimiento con la *especificación* de la máquina virtual de Java[26], por lo tanto, el *"*java byte code" correrá o se ejecutará como se espera que lo haga.

LOS PAQUETES -PROGRAMAS DE COMPUTACIÓN- PRE-ESCRITOS DE JAVA

Durante años cuando los programadores querían escribir programas de computación tenían que escoger una determinada plataforma.

[26] Véase la especificación de la máquina virtual de Java de la versión 16 de la Java SE en https://docs.oracle.com/javase/specs/jvms/se16/html/index.html

Entonces, cada una de las grandes compañías de tecnología, como Apple y Microsoft, habían desarrollado sus propias versiones de lenguajes de programación.

Con lo cual, cuando se escribía un programa para las computadoras con Microsoft Windows, ese programa no podía correr o ejecutarse en las Apple MacIntosh.

Por ende, si alguien quería que un programa de computación pudiera ejecutarse y funcionar en Microsoft Windows y en Apple MacIntosh, debía escribir el mismo programa dos veces.

Con la plataforma Java, todo esto cambió.

Publicada en 1996, y con una de sus salientes características la "máquina virtual de Java" (JVM), se permitía escribir programas que pudieran ejecutarse o correr en diferentes computadoras con distintos sistemas operativos, sin que existiese la necesidad de reescribir los programas.

Lo único que se requería era tener instalado el programa de la máquina virtual de Java donde se iba a ejecutar ese programa desarrollado en cuestión.

Por lo tanto, una vez escrito un programa podría correr o ejecutarse ese programa en cualquier computadora sin importar su sistema operativo instalado (MS, Linux, Solaris, etc.), lo único que se exigía era tener instalado el programa de la máquina virtual de Java.

Esto es el *"write once, run anywhere"*, "escrito una vez, corre en cualquier lado" -queriéndose hacer referencia a cualquier sistema (hardware más sistema operativo).

Los desarrolladores de Sun Microsystems crearon una cantidad importante de programas de Java con la finalidad de llevarse a cabo funciones usualmente requeridas para desarrollar

o programar “otros programas de computación”, y organizó estos programas *pre-escritos* en “paquetes” (en inglés “*packages”*).

El conjunto de estos paquetes o programas *pre-escritos* es lo que se conoce como la librería de software estándar de Java o librería de clases de Java.

Estos paquetes están acomodados u ordenados en una estructura compleja.

A su vez, esta estructura consiste en numerosos módulos de programas probados, testeados y confiables (en inglés *“trie-and true”*), que comprenden una gran cantidad de funciones.

Estos “paquetes de Java” fueron de gran utilidad para que otros programadores pudieran escribir o crear sus propios programas de aplicaciones para diferente tipo de dispositivos.

Entonces, estos programadores ante la necesidad de una determinada funcionalidad en su desarrollo no tenían que “reinventar la rueda”, sino por el contrario, podían echar mano a cualquiera de los paquetes de Java que incluyera la funcionalidad necesitada por ellos.

También, los programadores podían escribir dicho programa conteniendo la funcionalidad deseada por ellos “desde cero”, es decir, no era obligatorio usar esos paquetes de Java en la construcción de sus nuevos programas.

En el juicio, tanto por el Juez de Distrito como por las partes, a estos paquetes de software se los denominó “API”.

Oracle menciona que el término API, que significa “aplicación de programación de interface” es confuso, porque es una suerte de “camaleón” (en inglés “*verbal chamaleon”*), ya que puede describir un protocolo de comunicación común o

trivial como sería pasar información entre programas de computación, o puede describir programas sofisticados, como los escritos por Sun Microsystems y que son objeto del presente juicio.

En el presente litigio, la Corte de Distrito y las partes han confusamente aplicado el mismo significado para describir "al conjunto de paquetes de Java" y al "código fuente de un sólo paquete de Java".

Entonces, a modo ilustrativo en un paquete de Java, los desarrolladores de Sun Microsystems escribieron un programa llamado ***"URLConnection"*** -que es un programa de computación, no un protocolo de comunicación trivial o simple entre programas) para establecer la conexión a internet (por ejemplo, con un banco o una tienda on-line).

Más allá de que parezca simple, es extremadamente complejo.

Los desarrolladores de Sun Microsystems eran expertos en protocoles de redes y algoritmos criptográficos.

El *código fuente declarado* (o declaración o| método de encabezado) para ***URLConnection*** es:

public URLConnection openConnection()   ***trhows java.io.IOException***

Entonces, digamos que un tercero desea desarrollar una aplicación o programa para que se pueda conectar con ***BankofAmerica.com***

Así, ese tercero (programador) puede (i) reinventar la rueda, escribiendo su propio algoritmo, o (ii) simplemente usar el código disponible *pre-escrito* por Sun Microsystems/Oracle, y para ello debe "declarar" en su propio programa, tipeando el código fuente de más abajo el cual va a invocar o llamar al código *pre-escrito* por Sun Microsystems*:*

"new URL("http://bankofamerica.com").openConnection()"

A medida que los distintos programadores conocen y hacen uso de estas funcionalidades en forma frecuente, la utilización de los paquetes de Java se convierte en *un hábito de la programación en el lenguaje Java,* y los programadores no necesitan saber o mirar dentro de él, ya que lo que les interesa, es la funcionalidad que cumple y que ellos necesitan para incorporar a sus propios programas.

Cada paquete de Java consiste en dos tipos relacionados de código fuente, *código declarado y código implementado.*

Entonces, cada componente en un paquete de Java comienza con una o más líneas, incluyéndose entre otras cosas, de una "descripción de la función", como puede ser utilizando el ejemplo anterior ***"public URLConnection openConnection() throws java.io IOException"*** .

También, este código refleja el "lugar" del componente dentro de la estructura del paquete de Java.

A su vez, un código similar, es el que los programadores (nuevamente los terceros que construyen otros programas, por ejemplo aplicaciones, utilizando los paquetes de Java) deben

declarar con el objeto de invocar el programa *pre-escrito* de Java, y en el caso, sería las línea de código expresado en el ejemplo anterior ***"URL(String spec).openConnection()".***

A este tipo de líneas de código se las conoce como *declaraciones, encabezados, firmas* o *nombres* dependiendo de algunos factores que no son relevantes para el caso en particular.

Entonces, a estas líneas de código se las llama "declaraciones" o "código fuente declarado".

El segundo tipo de código fuente, el *código implementado,* le indica a la computadora "como" llevar a cabo o ejecutar una función declarada.

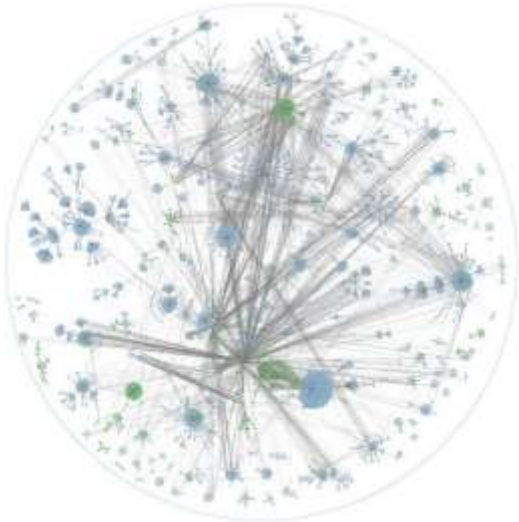
En el ejemplo mencionado, el código implementado sería las líneas de código indicando la apertura de la conexión de internet.

Estos programas *pre-escritos* están interconectados a través de una estructura, secuencia y organización.

A su vez, los programas están organizados dentro de "paquetes", y cada paquete se organiza en un conjunto de clases, y cada clase es organizada a su vez en un conjunto de métodos, los cuales cada uno de ellos cumple una *función específica*, como por ejemplo la de abrir una conexión a internet.

Estos paquetes de software -nuevamente se menciona- son los denominados como API en el juicio.

En la figura de más abajo se muestra la estructura de los paquetes y las clases, sin mencionarse en ella la complejidad de los 30.000 métodos.



Cada una de las líneas representa las opciones disponibles para los programadores de paquetes y clases.

Como se dijo anteriormente, los terceros programadores en la construcción de sus propios programas usando el lenguaje de programación Java, no tienen ninguna obligación de usar estos paquetes de Java "*pre-escritos*" por Sun Microsystems (a excepción de 170 líneas de código incluidas en tres paquetes de Java en: **Java.lang**, **Java.io** y **Java.util**).

En definitiva, quien desee usarlos para construir sus propios programas puede hacerlo, y quién no, puede desarrollar desde cero la misma funcionalidad -u otras no previstas en los paquetes de la librería de software Java- que sean necesarias según cada programador.

A su vez, Sun Microsystems especificó como trabaja el código de los paquetes en un documento que se denominó *"Especificación de las API de Java"*, que contiene más de

40.000 páginas y que *detalla el código declarado de Java* y su estructura organizacional.

La especificación de la API además describe la estructura y organización de cada uno de los paquetes de Java, -además de toda la estructura de todos los paquetes de Java-.

La especificación de Java, se *"espeja exactamente"* en el código declarado de los paquetes, incluyendo su estructura y organización.

La edición original de la plataforma Java Edición Estándar (Java SE) de 1996 tenía 8 paquetes de programas *pre-escritos*.

En 2005, cuando Google comenzó a copiar el código declarado, la versión de Java SE 5.0 tenía 166 paquetes, y así fue evolucionado, y por ejemplo en febrero de 2011 alcanzó a los 209 paquetes.

La creación de estos paquetes constituyó un proceso creativo e interactivo.

Parte de la creatividad se relaciona con establecer "que incluir" en cada uno de los paquetes y "como organizar" el código declarado de manera tal que el uso por parte de los programadores en la construcción de sus propios programas sea "intuitivo".

El código declarado puede ser muy extenso en su expresión creativa, por ejemplo:

public abstract void verify (PublicKey key, String sigProvider) throwsCertificateException,No-SuchAlgo-rithmException, InvalidKeyException, NoSuchProviderException, SignatureException

El proceso de creación comienza con la identificación de áreas de necesidad de nuevas o diferentes funciones por parte de los desarrolladores de Oracle en la comunidad del lenguaje de programación Java.

Las sugerencias, incluso, provienen del Comité Ejecutivo de Java (en inglés "*Java Executive Comittee"*) compuesto por Oracle, y empresas tales como IBM y Google[27].

Sun Microsystems y Oracle invirtieron varios millones de dólares en el trabajo de creación de los paquetes de Java.

Y ello incluyó la enseñanza en la comunidad de desarrolladores en cómo utilizar cada uno de esos paquetes de Java, de manera tal que cuando un programador de Java veía ***"URL.openConnection(),"*** sabía instantáneamente que eso significa "crear una conexión de internet", tan fácil y simple como eso.

LICENCIAMIENTO DE LA PLATAFORMA JAVA SE

[27] Al menos en el 2011 conformaban el Comité Ejecutivo de Java.

Oracle menciona que los varios componentes que conforman la plataforma Java SE (el lenguaje de programación Java, el Java *"runtime enviroment"* (que incluye la máquina virtual de Java, la librería estándar de Java y las API, y el "*Java development kit*" que incorpora una colección de programas de herramientas, incluyendo al compilador de Java) se licencia bajo diferentes "tipo de licencias", pero todas ellas poseen un objetivo común: *"proteger la compatibilidad de Java"*.

Entonces, más allá de los derechos de autor (conforme al régimen legal de los EE.UU.) sobre la plataforma Java SE propiedad de Oracle, Oracle promovió la utilización de los paquetes de Java por terceros, y para acomodar las necesidades de todos los interesados utilizó tres tipos de licencias, que son las siguientes:

a) *Licencia Pública General, versión 2 (*en inglés *"General Public License, version 2" o sólo "GPL v2")*[28]: Bajo este esquema de licencia de software libre y código abierto (FOSS) los licenciatarios podían ver, reproducir, modificar (y crear obras derivadas) y distribuir el código fuente de los paquetes de Java *(*en inglés *"Java packages")*, tanto del "código declarado" como del "código implementado".

b) *Licencia de Especificación*[29]: mediante esta licencia a diferencia de la licencia GPLv2, los licenciatarios sólo

[28] Véase en https://spdx.org/licenses/GPL-2.0-only.html

[29] Véase para la Especificación JSR- 389 Java SE versión 14, último *release* de marzo de 2020 https://download.oracle.com/otndocs/jcp/java_se-14-final-spec/license.html En su cláusula 2 se expresa: 2. "*License for the Distribution of Compliant Implementations*. *Specification Lead also grants you a perpetual, non-exclusive, non-transferable, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable*

podían acceder a la especificación de la plataforma Java SE (que recita el código fuente declarado). Es decir, esta licencia no permitía a los licenciatarios utilizar el "código fuente completo de Java". Por lo tanto, los licenciatarios utilizando sólo el código fuente declarado y la estructura de organización brindada por Oracle, debían desarrollar o construir su propia implementación independiente (es decir este tipo de código fuente) del conjunto de las API de Java o paquetes de Java. Entonces, esta licencia de especificación suponía un esquema de: *código fuente declarado de Sun Microsystems/Oracle + código fuente implementado del licenciatario de la especificación de Java (con el TCK aprobado lógicamente) = Código Binario Ejecutable del Licenciatario de la Especificación* (que constituiría la implementación independiente de Java por parte del tercero respetando el

---

copyrights or, subject to the provisions of subsection 4 below, patent rights it may have covering the Specification to create and/or distribute an Independent Implementation of the Specification that: ***(a)*** *fully implements the Specification including all its required interfaces and functionality;* ***(b)*** *does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; and* ***(c)*** *passes the Technology Compatibility Kit (including satisfying the requirements of the applicable TCK Users Guide) for such Specification ("Compliant Implementation"). In addition, the foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for any other purpose (including, for example, modifying the Specification, other than to the extent of your fair use rights, or distributing the Specification to third parties). Also, no right, title, or interest in or to any trademarks, service marks, or trade names of Specification Lead or Specification Lead's licensors is granted hereunder. Java, and Java-related logos, marks and names are trademarks or registered trademarks of Oracle America, Inc. in the U.S. and other countries"*

principio de "*write once, run anywhere*" instituído por Sun Microsystems/Oracle).

c) *Licencia Comercial*: mediante esta licencia los licenciatarios tenían acceso completo al código fuente (declarado e implementado) de Java con la finalidad de poder usarlo y adaptarlo para incorporarlo a sus propios productos comerciales y pudiendo mantener en secreto su código desarrollado. A cambio de ello, los licenciatarios debían pagar una regalía a Oracle.

En definitiva, ambas opciones a) y c) daban acceso por completo al código fuente (declarado e implementado) propiedad de Sun Microsystems/Oracle.

En el caso a) la opción bajo la GPLv2 implicaba que en caso de distribución de Java de "cualquier modificación" calificable como obra derivada bajo el sistema de derechos de autor de EE.UU. al producirse la distribución debía ser hecha disponible bajo los mismos términos y condiciones de la licencia GPLv2. (lo mismo aplicaba si la distribución de Java se producía sin modificaciones, ya que el disparador de la licencia conforme sus términos es la distribución)

La opción c) también daba acceso por completo al código fuente, pudiéndose modificar o adaptar Java, sin tener que publicar o hacer disponibles dichas modificaciones a terceros, con la única condición de tenerse que pagar una regalía a Sun Microsystems/Oracle.

Ahora bien, tanto la licencia comercial como la licencia de especificación debían pasar el denominado TCK o test de compatibilidad de Java. (no así el licenciamiento bajo la GPLv2)

LA ESPECIFICACIÓN DE LA PLATAFORMA JAVA SE

Sun Microsystems publicó la especificación de Java (protegida por derechos de autor) y ofreció licenciarla bajo determinadas condiciones.

Así para el caso de la especificación de la plataforma Java 2 Edición Estándar, Sun Microsystems permitía a los desarrolladores crear implementaciones independientes (en inglés *"clean room"*) de la especificación de la plataforma Java de Sun Microsystems con la condición de que se cumpliera con los requerimientos exigidos por su licencia de especificación.

Estos requerimientos, conforme Oracle eran los siguientes[30]:

(i) que se incluya una implementación completa de la versión actual de la especificación sin realizar *"subsetting"* o *"supersetting[31]"*,

[30] Véase el punto 10 en: *October 28, 2010- Filing 41 ANSWER TO COUNTERCLAIM #32 Answer to Complaint, Counterclaim Oracle America, Inc.'s Reply to Defendant Google Inc.'s Answer to Complaint for Patent and Copyright Infringement and Counterclaims by Oracle America, Inc. (Ballinger, Richard) (Filed on 10/28/2010)* https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/41/0.pdf?ts=1288527677 y en *November 29, 2010 Filing 60 ANSWER TO COUNTERCLAIM #51 Answer to Amended Complaint, Counterclaim by Oracle America, Inc.. (Peters, Marc) (Filed on 11/29/2010)* en *https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/60/0.pdf?ts=1291824224*

[31] Superponer la especificación ("*to superset"*) sería exceder las capacidades de la especificación estándar de Java de modo que los componentes del superconjunto agregados no funcionarían con la especificación estándar de Java. Y, por otro lado, subdividir la especificación (*"to subset"*) sería omitir ciertas capacidades de la especificación original, de manera que cualquier

(ii) que se implemente todas las interfaces y funcionalidades del paquete requerido por la plataforma Java 2, Edición Estándar definidas por Sun Microsystems sin realizar *"subsetting"* o *"supersetting"*,

(iii) No agregar ningún paquetes, clases o interfaces a los paquetes **java.* o javax.*** o a los subpaquetes de estas,

(iv) aprobar todos las pruebas o test relativos a la más reciente versión publicada de la especificación de la plataforma Java 2, Edición Estándar, las que eran disponibles por Sun Microsystems (en inglés *"Technology Compatibility Test"* o "TCK") seis (6) meses antes de cualquier versión beta publicada de la implementación independiente o de la actualización establecida en ella,

(v) que la implementación independiente no contenga código fuente u materiales binarios de propiedad de Sun Microsystems, y

(vi) que la implementación independiente no incluya código fuente o materiales binarios de Sun Microsystems sin una licencia apropiada y separada otorgada por Sun Microsystems.

---

programa diseñado bajo la especificación completa estándar de Java no funcionaría con la nueva especificación del subconjunto. De esta manera, Sun Microsystems/ Oracle se aseguraba de que todo el código Java desarrollado conforme a una licencia de especificación seguiría siendo "interoperable" al poder ejecutarse en cualquier sistema que tuviera instalado la máquina virtual estándar de Java, y, por lo tanto, mantener el principio de "escribir una vez, ejecutar en cualquier lugar" (WORA).

Oracle sostuvo que Google no cumplía con las condiciones enumeradas anteriormente conforme a la licencia de especificación de la plataforma Java 2, Edición Estándar.

También se sostiene por Oracle que los desarrolladores estaban en total conocimiento que la licencia de especificación de Sun Microsystems requería las pruebas de compatibilidad usando el TCK de Sun Microsystems, el cual menciona Oracle estaban disponibles y sin requerirse pago alguno para universidades, colegios, organizaciones sin fines de lucro, y personas físicas en http://java.sun.com/scholarship

GOOGLE COPIÓ EN FORMA IDÉNTICA EL CÓDIGO FUENTE DECLARADO DE 37 PAQUETES DE JAVA SE PARA INTRODUCIRLO EN LA PLATAFORMA ANDROID. GOOGLE QUEBRÓ EL PRINCIPIO WORA

Google pudo haber escrito sus propios paquetes para su plataforma Android en el lenguaje de programación Java, y no hay nada objetable en este punto.

Por el contrario, lo cuestionable, y lo que Google hizo fue copiar 37 paquetes de Java en forma idéntica.

Google copió los 37 paquetes de Java que creía que los programadores de Java querrían encontrar en Android, así *"invocables"* por los programadores para ser utilizados en la construcción de sus propios programas en la plataforma Android -no ya en la plataforma Java que son incompatibles- bajo los "mismos nombres" que se usaban en la plataforma Java.

Esta copia idéntica (reconocida por Google en el pleito) del código fuente declarado de los 37 paquetes de Java implicó la copia idéntica de la estructura de cada uno de los 37 paquetes, y luego Google "parafraseo" el código fuente implementado, es decir, con la especificación de las API de Java que espeja el código fuente declarado Google sólo siguió la "guía" de la especificación y completo los "espacios en blancos".

En definitiva, Google hizo lo que otras empresas que tomaron la licencia de especificación hicieron (ya que estas empresas podían hacerlo porque poseían justamente una licencia de especificación otorgada por Oracle), sin cumplir claramente Google con los requisitos de compatibilidad del TCK (diferencia de los que poseían una licencia de especificación).

Al ser incompatible Android con Java, Oracle menciona que Android es la única implementación de la especificación de Java incompatible con Java, es decir, Google quebró la especificación de Sun Microsystems, el lema *"write once, run anywhere",* siendo el lema de Android *"write once, run only in Android".*

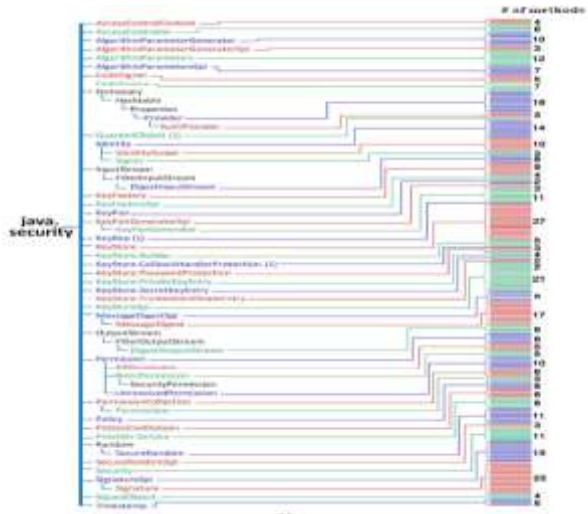
Google hizo la parte más fácil, la menos compleja al parafrasear el código fuente implementado.

La figura de más abajo muestra el paquete de Java denominado ***java.io***. Este paquete maneja *inputs* y *outputs*.

Sobre la izquierda se encuentran las numerosas clases, por ejemplo, ***InputStreamReader*** (que traduce un flujo de datos en texto legible para humanos). Las clases a su vez, pueden contener subclases, y a su vez, las subclases pueden contener sub-subclases, y así sucesivamente.

Google copió el código declarado del paquete ***java.io*** incluyendo todas sus clases. La figura no nuestra todos los métodos, que son un total de 569, distribuidos dentro de aproximadamente 50 clases. A su vez, la clase ***InputStreamReader*** tiene 5 métodos. Otras clases dentro de ***java.io*** poseen entre 3 y 39 métodos. Google, copió también en forma idéntica el código declarado de todos los métodos.

Por su parte, la figura de más abajo muestra lo copiado por Google a nivel método respecto del paquete de Java denominado ***java.security.***



Las líneas que parecen imperceptibles son los 362 métodos.

Google copió en forma idéntica los 362 métodos.

El código declarado que define el método o clase *incluye más información* que sólo el nombre y su función.

De hecho, la mayoría incluye "parámetros", que define su operación (por ejemplo: *"5"* ***directs java.io.StringReader.skip*** para saltear los próximos 5 caracteres).

La estructura de los paquetes de Java no es simplemente lineal o puramente estructural.

Un esquema real sería uno tridimensional con interconexiones complejas.

Entonces, habría que mirar la figura de arriba y pensar en los grupos marcados en color como si fuese el plano de un piso en un edificio de 50 pisos, con una red de toboganes y escaleras que conectan cada uno de los pisos, e incluso que conectan con otros edificios, representando a los demás paquetes de Java.

Estas conexiones son "interfaces" (que no se deben confundirse con la terminología de "interfaces" asociadas al juicio como API) con grupo de clases compartiendo características similares.

Así la figura ***java.io*** muestra la interface sobre la derecha en *itálica* conectando una clase con otra clase, dentro del mismo paquete y en otros paquetes de Java.

Entonces lo copiado por Google habría que multiplicarlo por los 37 paquetes de Java, con unas 677 clases, y unos 6508 métodos aproximadamente, totalizando al menos unas 7000 líneas de código declarado[32].

La copia de Google causó la fragmentación que Sun Microsystems y Oracle querían evitar.

A su vez, a pesar de que Google copió en forma idéntica el código fuente declarado de los 37 paquetes de las API de Java con la finalidad de atraer a los programadores de Java a su plataforma Android, *Google diseñó Android para ser incompatible con la plataforma Java, es decir que las*

[32] Al respecto de las 7000 líneas de código fuente declarado: La Corte de Distrito del Norte de California las estimó en 7000, pero Oracle menciona, si bien no lo discutió en la apelación de la sentencia, que dicha cifra es *muy baja* ya que "sólo las clases y métodos por sí solos" suman 7000, y el código fuente declarado de "un sólo elemento" son varias líneas de código.

aplicaciones escritas para Android no funcionan en Java, y las escritas para la plataforma Java no funcionan en Android.

A su vez, ningún programa escrito para la plataforma Java antes de la existencia de Android, funciona en la plataforma Android.

CAPITULO IV

LA POSICIÓN DE GOOGLE

Temario. *Android la plataforma de código abierto. La Open Handset Alliance (OHA). OpenJDK: La implementación de código abierto de la plataforma Java SE. La especificación de la plataforma Java SE. El TCK y la licencia de la especificación de Java SE. Apache Harmony: La implementación independiente de la especificación de Java SE por la Fundación de Software Apache. Apache Harmony y las API de los 37 paquetes de la librería estándar de Java. Android y la elección de Apache Harmony. La distribución de Apache Harmony en ausencia del TCK. GNU ClassPath. Google no viola ningún acuerdo de licencia relativo a la plataforma Android. El caso Sun Microsystems, Inc. v. Microsoft Corp., 188 F.3d 1115, 1118 (9th Cir. 1999) no es referenciable.*

ANDROID LA PLATAFORMA DE CÓDIGO ABIERTO (OPEN SOURCE) [33]. LA OPEN HANDSET ALLIANCE (OHA)

[33] Véase en la contestación de demanda de Google, en la sección *"Factual Background"* Punto C. Subpunto 20. Google afirma: "La plataforma Android ha sido un éxito en la industria de telefonía móvil. A pesar de que Android ha llegado después que otros al mercado de smartphones o teléfonos inteligentes, habiéndose hecho disponible el primero de ellos en el año 2008, hoy en día (octubre de 2012) existen aproximadamente 90 diferentes dispositivos basados en Android fabricados por 20 fabricantes distintos y disponibles en decenas de países. Aproximadamente unos 200.000 teléfonos basados en Android son activados diariamente en más de 50 operadores inalámbricos diferentes. El Mercado de Android ("Android Market") la tienda donde los desarrolladores pueden vender aplicaciones ("apps") que ellos mismos crean basados en dispositivos Android, posee más de 80.000 apps disponibles para ser descargadas. https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/32/0.pdf?ts=1376347301

En la contestación de demanda y contrademanda de fecha 4 de octubre de 2010[34], y en su ampliación de fecha 10 de noviembre de 2010, Google niega la infracción de derechos alegada por Oracle y comienza con una descripción de los hechos que hacen al sustento de su defensa.

Al respecto de la plataforma Android, expresa Google que la plataforma Android es un "*stack*" de software de código abierto *(*en inglés *"open source software")* para dispositivos móviles que incluye un (i) sistema operativo, (ii) middleware y (iii) aplicaciones.

Destaca Google, que una característica de la plataforma Android es la máquina virtual denominada Dalvik ("VM").

La máquina virtual Dalvik VM se basa en una versión del núcleo (en inglés "*kernel"*) de Linux para los servicios centrales del sistema, como seguridad, gestión de memoria, gestión de procesos, stack de red y modelo de controlador, y como una capa de abstracción entre el hardware y el resto del stack de software.

La máquina virtual Dalvik ejecuta archivos en formato ejecutable Dalvik (.dex).

Los desarrolladores de la plataforma Android pueden construir o crear aplicaciones para teléfonos móviles basados en

[34] *October 4, 2010-Filing 32*- GOOGLE INC.'S ANSWER to Complaint with Jury Demand, COUNTERCLAIM against Oracle America, Inc. by Google Inc. (Zimmer, Donald) (Filed on 10/4/2010) https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/32/0.pdf?ts=1376347301 y *November 10, 2010 Filing 51* Google Inc.'s ANSWER to Amended Complaint for Patent and Copyright Infringement, Amended COUNTERCLAIM against Oracle America, Inc. by Google Inc. (Zimmer, Donald) (Filed on 11/10/2010) https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/51/0.pdf?ts=1376938370

Android en diferentes lenguajes de programación, incluyendo al lenguaje de programación Java.

Si bien las aplicaciones de software para la plataforma Android pueden ser escritas en el lenguaje de programación Java, el código byte *(bytecode)* de Dalvik es distinto al código byte de Java.

Google expresa que la *Open Handset Alliance (OHA)*[35], un grupo de 78 empresas de tecnología y teléfonos móviles, que incluye a Google, decidió que la información y el código fuente para Android, incluidos Dalvik VM y el kit de desarrollo de software de Android ("SDK" o "*Software Development Kit*"), estén disponibles como software libre y de código abierto (en inglés *"free open source software -FOSS")*, y además de forma gratuita[36].

Agrega que la primera versión de Android fue publicada por la *Open Handset Alliance* en el mes de noviembre del año 2007[37].

---

[35] http://www.openhandsetalliance.com/oha_members.html entre sus miembros se encuentran las siguientes empresas: Accenture, WIPRO, Wind River (an Intel Company), Google, eBay, ARM, Intel Corporation, Texas Instruments Inc., Telefónica, ST-Ericsson, Qualcomm, NVIDIA Corporation, Marvel Semiconductor Inc., Broadcom Corporation, ACER Inc., Alcatel Mobile Phones, DELL, Fujitsu, Garmin International Inc., HTC Corporation, HUAWEI Technologies, Kyocera, Lenovo Mobile Communication Technology Ltd, LG Electronics Inc., Motorola Inc., NEC Corporation, Samsung Electronics, SHARP Corporation, SONY ERICSSON, Toshiba Corporation, ZTE Corporation, etc.

[36] http://source.android.com y http://developer.android.com

[37] Google hace referencia a las palabras del CEO de Sun Microsystems, Jonathan Schwartz felicitando a Android de esta manera: *"Needless to say, Google and the Open Handset Alliance just strapped another set of rockets*

Expresa que la mayoría del software de Android se encuentra disponible bajo los términos de la licencia de software de código abierto denominada Licencia de Software Apache, versión 2 (en inglés *"Apache Software License 2.0*[38] o simplemente *"Apache 2.0"*)[39].

---

to the community´s momentum-and the vision defining opportunity across out (and other) planets"

[38] https://www.apache.org/licenses/LICENSE-2.0.html

[39] Véase el siguiente link https://source.android.com/setup/start/licenses?hl=en en donde Google explica cuáles son las razones por las cuales decidió utilizar la Licencia Apache -y no la Licencia GNU LGPL-. Se menciona: *Why Apache Software License? For userspace (non-kernel) software, we prefer Apache 2.0 (and similar licenses such as BSD and MIT) over other licenses such as the Lesser General Public License (LGPL). Here's why. Android is about freedom and choice. The purpose of Android is to promote openness in the mobile world, and we can't predict or dictate all the uses for our software. So, while we encourage everyone to make open and modifiable devices, we don't think it's our place to force them to do so. Using LGPL libraries could be restrictive. Here are some of our specific concerns: In simplified terms, LGPL requires shipping of source to the application; a written offer for source; or linking the LGPL-ed library dynamically and allowing users to manually upgrade or replace the library. Android software is typically shipped as a static system image, so complying with these requirements restricts device manufacturer designs. For instance, it's difficult for a user to replace a library on read-only flash storage. LGPL requires the allowance of customer modification and reverse engineering for debugging those modifications. Most device makers don't want to be bound by these terms. Historically, LGPL libraries have been the source of many compliance problems for downstream device makers and application developers. Educating engineers on these issues is difficult and time consuming. It's critical to Android's success that device makers can easily comply with the licenses. The issues discussed above are our reasons for preferring Apache 2.0 for our code. They aren't criticisms of LGPL or other licenses. We appreciate all free and open-source licenses, and respect others' license preferences. We've simply decided that Apache 2.0 is right for our goals"*

Por otro lado, se menciona que ciertas partes de Android, como ser los parches del kernel de Linux, están disponibles bajo los términos de la licencia de código abierto denominada Licencia Pública General (GPL), versión 2 (en inglés "*General Public License"* o simplemente *"GPLv2"*)[40].

Al respecto de las principales librerías de clases *(*en inglés *"library classes")* de Dalvik VM éstas incorporan un subconjunto *(*en inglés *"a subset")* del proyecto Apache Harmony[41], que es la implementación independiente de código abierto de la librería estándar de Java realizada por la Fundación de Software Apache[42].

Más allá de la librería de software Harmony, la plataforma Android fue desarrollada de forma independiente [43] por la Alianza de Teléfonos Abiertos (Open Handset Alliance o OHA).

---

[40] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/plain/COPYING y https://spdx.org/licenses/GPL-2.0-only.html

[41] https://en.m.wikipedia.org/wiki/Apache_Harmony Apache Harmony is a retired open source, free Java implementation, developed by the Apache Software Foundation. It was announced in early May 2005 and on October 25, 2006, the Board of Directors voted to make Apache Harmony a top-level project. The Harmony project achieved (as of February 2011) 99% completeness for J2SE 5.0, and 97% for Java SE 6. The Android operating system has historically been a major user of Harmony, although since Android Nougat it increasingly relies on OpenJDK libraries.

[42] Véase información de la Fundación de Software Apache en https://apache.org/

[43] Menciona Google, que "*Android includes certain APIs in order to be interoperable with the Java Platform, i.e., so that programs written in the Java programming language using common programming techniques will run properly on Android devices. However, the Android source code – e.g., the code for the Dalvik virtual machine, etc. – does not use Oracle's Java source code"*. Véase *March 31, 2011 Filing 102* CLAIM

En efecto, Google afirma que la plataforma Android, incluyendo el sistema operativo de Android, el kit de desarrollo de Android ("SDK") y la máquina virtual Dalvik, fueron creados en forma independiente por Google sin ninguna referencia a ninguna obra protegida de propiedad de Oracle *(Punto 23. Defensa 15. Creación Independiente)*[44].

OPENJDK: LA IMPLEMENTACIÓN DE CÓDIGO ABIERTO DE LA PLATAFORMA JAVA SE. LA ESPECIFICACIÓN DE LA PLATAFORMA JAVA SE. EL TCK Y LA LICENCIA DE LA ESPECIFICACIÓN DE JAVA SE

Así las cosas, Google expresa que aproximadamente entre los años 2006[45] y 2007 Sun Microsystems anunció que parte del código fuente de la plataforma Java estaría disponible como

---

CONSTRUCTION STATEMENT Google's Responsive Claim Construction Brief filed by Google Inc. (Baber, Bruce) (Filed on 3/31/2011) https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/102/0.pdf?ts=1377022695

[44] En el punto E, subpunto 23 de su contestación de demanda Google agrega que la plataforma Android y ninguno de sus componentes infringe ninguna de las patentes mencionados por Oracle, a saber: U.S. Reissue Patent No. RE38,104 ("the '104 reissue patent"), and U.S. Patent Nos. 5,966,702 ("the '702 patent"), 6,061,520 ("the '520 patent"), 6,125,447 ("the '447 patent"), 6,192,476 ("the '476 patent"), 6,910,205 ("the '205 patent"), and 7,426,720 ("the '720 patent").

[45] El 13 de noviembre del año 2006 Sun Microsystems anunció que en el primer trimestre del año 2007 la plataforma Java SE sería software de código abierto bajo la licencia GPL v2. Véase el anuncio de Rich Green Vice Presidente Ejecutivo de Software de Sun Microsystems en https://www.youtube.com/watch?v=uknTHW0UIUA

software libre y de código abierto (en inglés "*free open source software* -FOSS").

En efecto parte del código fuente de la plataforma Java Standard Edition (SE) se hizo disponible como software de código abierto conforme a los términos de la licencia de software de código abierto denominada Licencia Publica General, versión 2 (en inglés *"General Public License v2"* o "*GPLv2"*)[46].

Es decir, antes de que Android fuera publicado, Sun Microsystems había hecho disponible los paquetes de las API de Java bajo la licencia GPLv2 más la excepción *ClassPath* bajo el nombre de OpenJDK[47], con lo cual Google argumenta que

[46] Oracle reconoció que Sun Microsystems había hecho disponible parte del código fuente de la Plataforma Java Edición Estándar y también de otras Ediciones durante el año 2006 y 2007 sujeto a los términos y condiciones de la Licencia Pública General, Versión 2 (General Public License, version 2 o "GPLv2"). *Véase October 28, 2010- Filing 41 ANSWER TO COUNTERCLAIM #32 Answer to Complaint, Counterclaim Oracle America, Inc.'s Reply to Defendant Google Inc.'s Answer to Complaint for Patent and Copyright Infringement and Counterclaims by Oracle America, Inc. (Ballinger, Richard) (Filed on 10/28/2010)* https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/41/0.pdf?ts=1288527677 y *November 29, 2010 Filing 60 ANSWER TO COUNTERCLAIM #51 Answer to Amended Complaint, Counterclaim by Oracle America, Inc.. (Peters, Marc) (Filed on 11/29/2010)* en *https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/60/0.pdf?ts=1291824224*

[47] Véase en el sitio oficial de OpenJDK que se expresa por Oracle lo siguiente: "CLASSPATH" EXCEPTION TO THE GPL Certain source files distributed by Oracle America and/or its affiliates are subject to the following clarification and special exception to the GPL, but only where Oracle has expressly included in the particular source file's header the words "*Oracle designates this particular file as subject to the "Classpath" exception as provided by Oracle in the LICENSE file that accompanied this code."* Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination. As a special

exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version. "ADDITIONAL INFORMATION ABOUT LICENSING Certain files distributed by Oracle America, Inc. and/or its affiliates are subject to the following clarification and special exception to the GPLv2, based on the GNU Project exception for its Classpath libraries, known as the GNU Classpath Exception. Note that Oracle includes multiple, independent programs in this software package. Some of those programs are provided under licenses deemed incompatible with the GPLv2 by the Free Software Foundation and others. For example, the package includes programs licensed under the Apache License, Version 2.0 and may include *FreeType*. Such programs are licensed to you under their original licenses. Oracle facilitates your further distribution of this package by adding the Classpath Exception to the necessary parts of its GPLv2 code, which permits you to use that code in combination with other independent modules not licensed under the GPLv2. However, note that this would not permit you to commingle code under an incompatible license with Oracle's GPLv2 licensed code by, for example, cutting and pasting such code into a file also containing Oracle's GPLv2 licensed code and then distributing the result. Additionally, if you were to remove the Classpath Exception from any of the files to which it applies and distribute the result, you would likely be required to license some or all of the other code in that distribution under the GPLv2 as well, and since the GPLv2 is incompatible with the license terms of some items included in the distribution by Oracle, removing the Classpath Exception could therefore effectively compromise your ability to further distribute the package. Failing to distribute notices associated with some files may also create unexpected legal consequences. *Proceed with caution and we recommend that you obtain the advice of a lawyer skilled in open-source matters before removing the Classpath Exception or making modifications to this package which may subsequently be redistributed and/or involve the use of third-party software".* https://openjdk.java.net/legal/gplv2+ce.html

cualquiera podría haber duplicado las API de Java de los mismos 37 paquetes (tomados por Google y puestos en Android con la misma estructura, secuencia y organización) sin obligación alguna de pagar por ello, y con la sola condición de dar cumplimiento a los términos de la licencia GPLv2 más la excepción *ClassPath*.

Como se verá más adelante Google no copió o duplicó el código fuente declarado y su estructura, secuencia y organización del proyecto *OpenJDK* de Sun Microsystems/Oracle, sino que lo hizo del proyecto Apache Harmony, de la Fundación de Software Apache.

Google expresa que Sun Microsystems hizo disponible "la especificación de la plataforma Java SE"[48] bajo una licencia gratuita sin costos, mediante la cual se permitía a los desarrolladores construir implementaciones independientes *(*en inglés *"clean room implementation")* de la "especificación de la plataforma Java de Sun Microsystems".

Esta licencia se la denominó "licencia de especificación".

Entonces la única manera de demostrar compatibilidad con la "especificación de Java de Sun Microsystems" era cumplir con todos y cada uno de los requisitos que imponía Sun Microsystems en el denominado "Test de Compatibilidad de Java" (en inglés "*Sun´s Technology Compatibility Kit"* o "TCK" o también conocido como "*Java Compatibility Kit"* o "JCK") para cada edición particular de la plataforma Java.

[48] Los links mencionados por Google en su demanda que son: http://java.sun.com/docs/books/jls/third_edition/html/jcopyright.html y http://java.sun.com/docs/books/jvms/second_edition/html/Copyright.doc.html, hoy redirigen al siguiente link de Oracle https://docs.oracle.com/javase/specs/

Conforme la posición de Google si una determinada implementación independiente demostraba compatibilidad con la especificación de Java de Sun Microsystems, entonces Sun Microsystems concedía una licencia que cubría todos los derechos de propiedad intelectual (incluyéndose todos los derechos de patentes de invención y derechos de autor) que eran necesarios para practicar la especificación.

Como ejemplo de implementación independiente de la especificación de Java, Google menciona la realizada por la Fundación Apache denominada "Apache Harmony"[49].

[49] Al respecto del proyecto Apache Harmony véase en el sitio oficial de la Fundación de Software Apache. https://apache.org/jcp/sunopenletterfaq.html What is the Apache Harmony project? Apache Harmony is a project of the Apache Software Foundation *focused on creating an independent, compatible implementation of Java SE. That means we're writing the whole implementation from scratch, or incorporating software from other open-source projects*. Why was Harmony created? Harmony was created for many reasons. The fundamental reason is the same as other projects in the open source/free software space working Java (such as GNU Classpath, Kaffe, GCJ, etc) - *we wanted to do an implementation of a complete, compatible Java SE runtime environment, including virtual machine, class library and tools under a FLOSS license*. Was it always possible to create and distribute implementations of JSRs under free and open-source licenses? No, but the ASF was instrumental in making this possible. In 2002, the Apache Software Foundation, working with other members of the Java community, led the effort to change the JCP governance document - the "Java Specification Participation Agreement" or JSPA. These changes finally made it possible to create independent implementations of Java specification under free and open-source licenses. Before these changes, it was impossible to do so". Why doesn't Apache simply ignore this and ships Harmony without passing the JCK? We can ship Harmony without passing the JCK - *it's our source code to do with what we wish* - and we will with milestone releases as we progress towards completion. However, we could never claim to be Java compatible, which is something very important to Java users, and is the stated goal of the project. Also, users wouldn't be assured that they had all necessary IP rights from the spec's contributors. Compatibility is important to us as is not putting

APACHE HARMONY: LA IMPLEMENTACIÓN INDEPENDIENTE DE LA ESPECIFICACIÓN DE JAVA SE REALIZADA POR LA FUNDACIÓN DE SOFTWARE APACHE

Ahora bien, al respecto del TCK y su licencia de especificación, antes de convertirse Java en software de código abierto, incluía ciertas restricciones, tales como términos de licenciamiento adicionales y pagos por licencias.

Conforme los dichos de Google, Sun Microsystems fue criticado por miembros de la comunidad de código abierto, incluyendo a Oracle (antes de que adquiriera a Sun Microsystems), por la negativa de Sun Microsystems de no hacer disponible la totalidad el código fuente de la plataforma Java, es decir por no hacer Java completamente software libre y de código abierto (en inglés *"free open source software -FOSS"*).

Google hace referencia al caso de la Fundación de Software Apache, y menciona que por el año 2006, la Fundación Apache intentó obtener un test de compatibilidad "TCK" por parte de Sun Microsystems para verificar la compatibilidad de la implementación independiente (Apache Harmony) realizada por la Fundación Apache, con Java.

El objetivo del TCK es demostrar la compatibilidad entre la implementación independiente (Apache Harmony) y la especificación de Java.

Debido a las críticas, Sun Microsystems ofreció liberar el TCK bajo una licencia de software libre y de código abierto (FOSS), pero imponía a la vez ciertas "restricciones de uso"[50]

users in IP jeopardy, as it has been for every JSR the ASF has ever implemented. We have no interest in forking the technology.

[50] Véase en el sitio web oficial de la Fundación de Software Apache https://apache.org/jcp/sunopenletterfaq.html What is a "field of use"

(por sus siglas en inglés “FOU” “Field of Use”) que limitaban las circunstancias por la cuales los usuarios de Apache Harmony podían usar el software creado en forma independiente por la Fundación Apache: por ejemplo, una de esas restricciones del TCK incluía la telefonía móvil.

Ante tales circunstancias que no eran aceptables para la Fundación Apache el 10 de abril del 2007 la Fundación Apache publicó un carta abierta [51] dirigida a Sun Microsystems

---

restriction? A "field of use" restriction is a restriction that limits how a user can use a given piece of software, either directly or indirectly. To give a concrete example from the Sun/Apache dispute, if Apache accepted Sun's terms, then users of a standard, tested build of Apache Harmony for Linux on a standard general purpose x86-based computer (for example, a Dell desktop) would be prevented from freely using that software and that hardware in any application where the computer was placed in an enclosed cabinet, like an information kiosk at a shopping mall, or an X-ray machine at an airport.

[51] Véase en https://www.apache.org/jcp/sunopenletter.html “*On April 10, 2007, the Apache Software Foundation sent the following letter to Sun Microsystems regarding our inability to acquire an acceptable license for the Java SE 5 technology compatibility kit, a test kit needed by the Apache Harmony project to demonstrate compatibility with the Java SE 5 specification, as required by the Sun specification license for Java SE 5. …” Since August 2006, the ASF has been attempting to secure an acceptable license from Sun for the test kit for Java SE. This test kit, called the "Java Compatibility Kit" or "JCK", is needed by the Apache Harmony project to demonstrate its compatibility with the Java SE specification, as required by Sun's specification license. The JCK license Sun is offering imposes IP rights restrictions through limits on the "field of use" available to users of our software*. La Fundación Apache manifestó que las restricciones impuestas por Sun Microsystems eran violatorias del Contrato de Participación en las Especificaciones de Java (Sección 5) (“*Java Specification Participation Agreement*”) que es el acuerdo que establece las reglas del Proceso de la Comunidad Java (*Java Community Process* o JCP) del cual el mismo Sun Microsystems estaba obligado a cumplir teniendo en cuenta que era parte firmante del mismo. Véaselo en https://jcp.org/aboutJava/communityprocess/JSPA2.pdf

solicitándole, dentro del plazo de 30 días[52] para que otorgue (i) una licencia de especificación TCK o JCK sin restricciones de uso, o (ii) una explicación de parte de Sun Microsystems de porque estaba protegiendo porciones del negocio comercial de Java a expensas del software de código abierto de la Fundación Apache y violando la promesa pública realizada por Sun Microsystems acerca de que toda especificación liderara por Sun Microsystems (como lo era Java) sería completamente implementable y distribuible como software libre y de código abierto (en inglés *"free open source software*- FOSS*"*).[53]

Más allá de ello, Sun Microsystems continúo rechazando la postura y argumentos de la Fundación Apache[54].

---

[52] https://www.apache.org/jcp/sunopenletter.html *"We expect you to offer an acceptable, JSPA-compliant license to us within 30 days, or provide a public explanation of why you cannot do so"*.

[53] Básicamente lo que quería la Fundación Apache era que Sun Microsystems removiera las "restricciones de uso" (FOU), y a su vez que no estuviera condicionada solamente esa remoción a la Licencia Apache V.2.0 sino a cualquier licencia de software libre y de código abierto (FOSS), es decir independientemente de la Licencia FOSS, la "FOU" debía ser removida, y no sólo si la implementación independiente (Apache Harmony) de la Fundación Apache se licenciaba bajo los términos de la Licencia Apache V. 2.0. La Fundación Apache promulga las especificaciones abiertas *("Open Specifications")* por ende su deseo era que se pudieran realizar implementaciones abiertas bajo licencias de código abierto (FOSS). Un punto importante que destaca la Fundación Apache es que cualquier *obra derivada de la implementación de Apache Harmony* debía pasar su propio TCK, y lo dice así "*Note that it's still true that if someone makes a derivative work of Apache Harmony, they are still obligated to secure their own JCK license and test their derivative if they wish to call their work compatible"*

[54] Oracle reconoció que en agosto de 2006 la Fundación de Software Apache requirió un TCK (el test de compatibilidad con la plataforma Java) para verificar que la implementación independiente denominada Apache Harmony realizada por la Fundación de Software Apache era compatible con Java, y

Menciona Google, que como miembro del Comité Ejecutivo del Proceso Comunitario de Java (en inglés *"Java Community Process"* o por sus siglas "JCP")[55] Oracle había manifestado su preocupación acerca de la posición de Sun Microsystems, en relación al rechazo de hacer la totalidad de la plataforma Java software libre y de código abierto.

Así en el mes de diciembre del año 2007 durante una reunión en el JCP, Oracle propuso que el JCP dispusiera una nueva política simplificada de derechos de propiedad intelectual que permitiera el mayor número posibles de implementaciones[56].

---

también reconoció Oracle, que Sun Microsystems incluyó la restricción de uso (FOU) en la Licencia TCK. Oracle también reconoció que la Fundación Apache escribió en el mes de abril de 2007 una carta abierta requiriendo una licencia TCK (o de especificación) sin la restricción de uso (FOU), y que Sun Microsystems rechazó el retirar la restricción de uso. Véase el documento "*November 29, 2010 Filing 60 ANSWER TO COUNTERCLAIM #51 Answer to Amended Complaint, Counterclaim by Oracle America, Inc. (Peters, Marc) (Filed on 11/29/2010)* en https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/60/0.pdf?ts=1291824224

[55] The Java Community Process (JCP): *Program is the process by which the international Java community standardizes and ratifies the specifications for Java technologies. The JCP ensures high-quality specifications are developed using an inclusive, consensus-based approach. Specifications ratified by the JCP must be accompanied by a Reference Implementation (to prove the Specification can be implemented) and a Technology Compatibility Kit (a suite of tests, tools, and documentations that is used to test implementations for compliance with the Specification)* Conforme surge de https://jcp.org/en/participation/committee a Mayo de 2021 el Comité Ejecutivo del JCP está conformado por: Alibaba, ARM, Bellsoft, Marcus Biel, BNY Mellon, Eclipse Foundation, Ken Fogel, Fujitsu Limited, JetBrains, IBM, Intel, London Java Community, MicroDoc, Oracle, SAP SE, SouJava, Tomitribe y Twitter.

[56] Oracle reconoció en las reuniones del Comité Ejecutivo de los días 4 y 5 de Diciembre de 2007 (http://jcp.org/aboutJava/communityprocess/summaries/2007/December07-

A su vez, la empresa BEA Systems, que en ese momento se encontraba en negociaciones de ser adquirida por Oracle (lo cual sucedió posteriormente), *propuso una resolución acerca de que las licencias de TCK fueran ofrecidas sin ninguna restricción de uso para ningún área*, por lo tanto, permitiéndole el uso sin restricciones a organizaciones como la Fundación Apache.

Google afirma, que Oracle votó a favor de dicha resolución.

Expresa Google, que un año después, en el mes de febrero del 2009, Oracle reiteró su posición [57] cuando respaldó una

summary.html) del Java Community Process había propuesto la *Resolución 1* que decía lo siguiente: "It is the sense of the Executive Committee that the JCP become an open independent vendor-neutral Standards Organization where all members participate on a level playing field with the following characteristics: 1) members fund development and management expenses, 2) a legal entity with by-laws, governing body, membership, etc., 3) a new, simplified IPR Policy that permits the broadest number of implementations, 4) stringent compatibility requirements, 5) dedicated to promoting the Java programming model. Furthermore, the EC shall put a plan in place to make such transition as soon as practical with minimal disruption to the Java Community." Oracle también reconoció que dicha resolución fue respaldada por Bea Systems. Para ello, véase *November 29, 2010 Filing 60 ANSWER TO COUNTERCLAIM #51 Answer to Amended Complaint, Counterclaim by Oracle America, Inc. (Peters, Marc) (Filed on 11/29/2010)* en *https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/60/0.pdf?ts=1291824224*

[57] Oracle reconoció que en la reunión del mes de Abril del 2009 del Comité Ejecutivo del Java Community Process se dijo lo siguiente, y que Oracle votó afirmativamente diciendo "sí": "Following the February 2009 EC meeting, the following motion for electronic (email) voting was proposed by Apache Software Foundation and seconded by Intel Corp. *"TCK licenses must not be used to discriminate against or restrict compatible implementations of Java specifications by including field of use restrictions on the tested implementations or otherwise. Licenses containing such limitations do not meet the requirements of the JSPA, the agreement under which the JCP operates, and violate the expectations of the Java community that JCP specs can be openly implemented*." Para ello véase *November 29, 2010 Filing 60*

declaración que decía *"Las licencias TCK no deben ser usadas para discriminar en contra de o restringir implementaciones compatibles de la especificación de Java mediante la inclusión de restricciones de uso en el testeo de dichas implementaciones"*

Sólo unos meses después, en abril de 2009 Oracle anunció que estaría adquiriendo Sun Microsystems.

Desde ese mismo momento, expresa Google, y contrariamente a las declaraciones realizadas por Oracle anteriormente, Oracle ha ignorado el requerimiento público de la comunidad de código abierto de hacer totalmente la plataforma Java software libre y de código abierto (FOSS).

APACHE HARMONY Y LAS API DE LOS 37 PAQUETES DE LA LIBRERÍA DE SOFTWARE ESTÁNDAR DE JAVA

Conforme constancias del juicio[58], Apache Harmony, como se mencionó anteriormente es la implementación de código

ANSWER TO COUNTERCLAIM #51 Answer to Amended Complaint, Counterclaim by Oracle America, Inc. (Peters, Marc) (Filed on 11/29/2010) en *https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/60/0.pdf?ts=1291824224*

[58] Esta decisión del Juez William Alsup del mes de abril del año 2016 (en esta época ya se estaba discutiendo la cuestión del uso justo o "*Fair Use*") *consideró que Google tenía derecho a contar de donde había obtenido el software bajo disputa,* y que, por lo tanto, la historia relacionada a la implementación Harmony de la Fundación Apache era relevante, y se dijo así: *THE APACHE STORY IS RELEVANT TO WILLFULLNESS AND FAIR USE. Given that Oracle is accusing Google of willful copyright* infringement, *Google is entitled to present to the jury the story of how it acquired the software code in question, including its downloading of the 37 APIs in question from Apache's website. At a minimum, therefore, the Apache story is relevant to Google's defense against Oracle's charge of willfulness. April*

abierto independiente de la especificación de la plataforma Java SE realizada por la Fundación de Software Apache, contiene las API de los 37 paquetes de la librería estándar de Java que están en discusión en el presente juicio.

También surge de estas mismas constancias que a comienzos del año 2005 la Fundación Apache hizo disponible públicamente el código fuente de Harmony pudiéndose descargar el mismo desde el sitio web de la Fundación Apache.

En algún momento después del año 2005 (sin que sea haya precisado una fecha exacta en estas constancias mencionadas) Google descargo el código fuente de Harmony para usarlo en la plataforma Android[59].

28, 2016. Opinion or Order Filing 1748 ORDER IN LIMINE RE ORACLE'S MOTION IN LIMINE NO. 2 TO EXCLUDE EVIDENCE OF APACHE HARMONY AND GNU CLASSPATH by Judge William Alsup [denying #1552 Motion in Limine]. The Court reserves its ruling on evidence regarding GNU Classpath for the reasons stated in this order. (whasec, COURT STAFF) (Filed on 4/28/2016). https://www.plainsite.org/dockets/download.html?id=235606491&z=2d4c55f9

[59] En relación a la "restricción de uso" (FOU) se dijo "*Google proffers that at the time in question, it was an accepted practice that the header lines in the APIs could be copied freely without a license so long as the copier used its own implementing code and, in the case of Java, did not call itself Java. If this proves true, this would certainly mitigate Oracle's charge of bad faith and willful acquisition and use and explain why Google might have believed that it needed no license whatsoever from Sun to use the Apache Harmony APIs. So, at least to some extent, the Apache story bears upon whether or not there was a custom and practice in favor of the kind of use made by Google. One caveat: it is problematic (from Google's point of view) that Google knew of Sun's field-of-use restriction, so it is not entirely clear how the Apache story will establish custom and practice within the meaning of Wall Data. Accordingly, the Court will listen carefully to the evidence as it is presented and at some point may restrict further evidence regarding custom under Rule 403, but the Court is not yet prepared to say that balance favors exclusion*

ANDROID Y LA ELECCIÓN DE APACHE HARMONY

Así las cosas, Google escogió a Apache Harmony la implementación independiente y de código abierto de la especificación de la plataforma Java SE de Sun Microsystems realizada por la Fundación Apache teniendo en cuenta que Harmony era licenciada por la Fundación Apache bajo los términos de la licencia de software Apache versión 2.0 (Apache v2.0) la cual permitía a los desarrolladores de Android modificar el código fuente de Harmony sin tener que enviar dichas modificaciones nuevamente a la Fundación Apache.

LA DISTRIBUCIÓN DE APACHE HARMONY EN AUSENCIA DEL TCK[60]

altogether. April 28, 2016. Opinion or Order Filing 1748 ORDER IN LIMINE RE ORACLE'S MOTION IN LIMINE NO. 2 TO EXCLUDE EVIDENCE OF APACHE HARMONY AND GNU CLASSPATH by Judge William Alsup [denying #1552 Motion in Limine]. The Court reserves its ruling on evidence regarding GNU ClassPath for the reasons stated in this order. (whasec, COURT STAFF) (Filed on 4/28/2016) https://www.plainsite.org/dockets/download.html?id=235606491&z=2d4c55f9

[60] Véase resolución del Juez William Alsup. *April 28, 2016. Opinion or Order Filing 1748 ORDER IN LIMINE RE ORACLE'S MOTION IN LIMINE NO. 2 TO EXCLUDE EVIDENCE OF APACHE HARMONY AND GNU CLASSPATH by Judge William Alsup [denying #1552 Motion in Limine]. The Court reserves its ruling on evidence regarding GNU Classpath for the reasons stated in this order. (whasec, COURT STAFF) (Filed on 4/28/2016)* https://www.plainsite.org/dockets/download.html?id=235606491&z=2d4c55f9

Durante el pleito se planteó la situación de si la implementación independiente de la especificación de Java SE realizada por la Fundación Apache podía haber sido distribuida teniendo en cuenta que no había obtenido la aprobación del TCK, es decir, del test de compatibilidad entre la implementación independiente y la especificación de Java SE.

Google y Oracle no estuvieron de acuerdo en esto.

Así Oracle argumentó que la Fundación Apache no podía haber distribuido Harmony ya que como no había logrado obtener la aprobación de compatibilidad entre su implementación y la especificación de Java, pues nunca había entonces obtenido una licencia de especificación por parte de Sun Microsystems u Oracle.

Es decir, la condición de requisito esencial que marca Oracle es que la licencia de especificación estaba sujeta a obtener la aprobación del test de compatibilidad, no obtenida esta, no hay derecho a licencia alguna.

Por otra parte, Google alega que la Fundación Apache podía haber distribuido -como de hecho lo hizo cuando dejo disponible para descarga a Apache Harmony desde su sitio web- Apache Harmony teniendo en cuenta que es *"su propio código fuente [implementado*]"[61], es decir que las líneas de código fuente eran de la Fundación Apache, y no de Sun Microsystems u Oracle.

[61] Nuevamente, What is the Apache Harmony project? Apache Harmony is a project of the Apache Software Foundation focused on creating an independent, compatible implementation of Java SE. That means we're writing the whole implementation from scratch, or incorporating software from other open-source projects. https://apache.org/jcp/sunopenletterfaq.html

Lo único que no podía hacer la Fundación Apache es llamar o denominar a Apache Harmony como "compatible con Java"[62].

GNU CLASSPATH[63]

Por otra parte, conforme constancias del juicio[64], Google no quiso [65] tomar como referencia (código fuente) la

[62] La compatibilidad con Java fue el objetivo primario de la Fundación Apache- Véase cuando se dice: *Why was Harmony created?* Harmony was created for many reasons. The fundamental reason is the same as other projects in the open source / free software space working Java (such as GNU Classpath, Kaffe, GCJ, etc.) *we wanted to do an implementation of a complete, compatible Java SE runtime environment, including virtual machine, class library and tools under a FLOSS license*. https://apache.org/jcp/sunopenletterfaq.html

[63] https://www.gnu.org/software/classpath/

[64] *April 28, 2016. Opinion or Order Filing 1748 ORDER IN LIMINE RE ORACLE'S MOTION IN LIMINE NO. 2 TO EXCLUDE EVIDENCE OF APACHE HARMONY AND GNU CLASSPATH by Judge William Alsup [denying #1552 Motion in Limine]. The Court reserves its ruling on evidence regarding GNU ClassPath for the reasons stated in this order. (Filed on 4/28/2016)* https://www.plainsite.org/dockets/download.html?id=235606491&z=2d4c55f9

[65] El testimonio de Dan Bornstein, un programador experto de Android, propuesto por Google. Se menciona que Bornestein, dijo así: "Bornstein went on to describe how Android was built, using a combination of open-source software off the Internet and code written at Google. Apache Harmony was a major source for the "core libraries" that Android needed. His team used some Java APIs and built many Android APIs of their own. Annette Hurst, abogada de Oracle preguntó a Bornestein "You instructed Noser [a Google contractor] that they absolutely could not use Classpath code, because it was covered by an incompatible license," said Hurst, referring to the GPL license used by GNU Classpath. "Isn't that true?" "We did not want to use code, as much as possible, that used that license," Bornstein agreed. Hurst asked about more

implementación independiente de código abierto de la especificación de Java llamada GNU *ClassPath* realizada por la Fundación de Software Libre (Free Software Foundation -FSF).

GNU *ClassPath* implementa la librería estándar de Java o librerías de clases (o en inglés "Java Class"), y se licencia bajo los términos de la Licencia Pública General, versión 2 (GPL v2) más la excepción ClassPath[66].

examples of code that Bornstein wanted to avoid because of licensing issues. It was a return to a theme Oracle lawyers have brought up in other cross-examinations: the idea that Google was blowing off licensing rules. "And you didn't use the JDK source code because the license was incompatible with Android?" asked Hurst. "It was more about the timing," said Bornstein. "The [Apache] Harmony train had left the station. We were well on our way to getting it done. Technically, we could have dropped it, but I don't think there would have been a point to it." https://arstechnica.com/tech-policy/2016/05/top-programmer-describes-androids-nuts-and-bolts-in-oracle-v-google/

[66] Véase "*ClassPath* is distributed under the terms of the GNU General Public License with the following clarification and special exception: "Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination. *As a special exception*, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version". https://www.gnu.org/software/classpath/license.html

GOOGLE NO VIOLA NINGÚN ACUERDO DE LICENCIA RELATIVO A LA PLATAFORMA ANDROID [67]. EL CASO SUN MICROSYSTEMS, INC. V. MICROSOFT CORP., 188 F.3D 1115, 1118 (9TH CIR. 1999) NO ES REFERENCIABLE

Google menciona [68] que el caso citado por Oracle, *Sun Microsystems, Inc. v. Microsoft Corp., 188 F.3d 1115, 1118 9th Cir. 1999,* en donde Sun Microsystems demandó a Microsoft no es relevante para el juicio en cuestión por cuanto los hechos son muy diferentes.

Google menciona que Microsoft había firmado un contrato con Sun Microsystems por el cual se le concedía a Microsoft una licencia o autorización para crear obras derivadas de la plataforma Java SE y a usar el logo de "Java compatible" en la medida y siempre y cuando esas obras derivadas creadas por

[67] Véase: *Order denying Rule 50 Motions, June 8, 2016.* En relación a una instrucción solicitada por Google relativa a lo que se debía entenderse por costumbre (o en inglés *"custom"*) conforme el caso *Wall Data Inc. V. Los Angeles County Sheriff´s Dept (9th Cir. 2006)* relacionado al uso justo. Y en su parte relevante en relación a la inexistencia de licencia se cita el párrafo 27 y se dice *"Similarly, you have heard evidence about various license from Apache Foundation, Apache Harmony involving Java, and the General Public License. These are relevant in some ways, but Google concedes it had no license from Sun or Oracle, and it is important to remember that Google makes no claim that its use was pursuant to a license from Sun or Oracle, directly or indirectly. Instead, Google claims that its use was a fair use and therefore required no license at all"*

[68] Véase *March 31, 2011 Filing 102 CLAIM CONSTRUCTION STATEMENT Google's Responsive Claim Construction Brief filed by Google Inc. (Baber, Bruce) (Filed on 3/31/2011)* https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/102/0.pdf?ts=1377022695

Microsoft hayan pasado los requerimientos de compatibilidad establecidos por Sun Microsystems.

Argumenta Google que Oracle no puede acusar a Google de violar acuerdo de licencia alguno relativa a la plataforma Android, ya que Google no utiliza el logo de "Java compatible" en relación a la plataforma Android como tampoco está sujeto Android a ningún requisito de compatibilidad o cualquier otro requerimiento.

Sostiene Google que la circunstancia de que Oracle haya afirmado *"que Android no ha implementado la totalidad de la especificación de Java y por ende no da cumplimiento a ella"* confirma que Android no es una plataforma Java.

CAPITULO V

VEREDICTO DEL JURADO Y SENTENCIA DE LA CORTE DE DISTRITO[69] NORTE DE CALIFORNIA[70]

[69] Véase https://www.cand.uscourts.gov/ La Corte de Distrito de California abarca los siguientes 15 condados: Alameda, Contra Costa, Del Norte, Humboldt, Lake, Marin, Mendocino, Monterey, Napa, San Benito, San Francisco, San Mateo, Santa Clara, Santa Cruz, y Sonoma. Por otro lado, en el sistema judicial de los EE.UU. existen 94 Cortes de Distritos o Juzgado de Primera Instancia véase en https://www.uscourts.gov/about-federal-courts/court-role-and-structure con su respectivas Corte de Apelaciones https://www.uscourts.gov/sites/default/files/u.s._federal_courts_circuit_map_1.pdf

[70] *Oracle America, Inc., (Plaintiff) v. Google Inc. (Defendant). May, 31, 2012. No. C 10-03561 WHA, US District Court for The Northern District of California. Order RE Copyrightability of Certain Replicated Elements of the Java Application Programming Interface*. En su sentencia el Juez William Alsup menciona que debido a la complejidad del caso la Corte del Distrito decidió que el Jurado entendería mejor las problemáticas planteadas por las partes si el juicio ("*trial*") se llevaba a cabo *en fases*. Razón por la cual se establecieron tres (3) fases. La *primera fase* sería la que revisaría (i) si la tecnología bajo disputa era protegible bajo la ley de los derechos de autor de EE.UU. ("*copyrighability*") (ii) y si fuera protegible, si analizaría la existencia o no de la infracción a los derechos de autor ("*copyright infringement*"). Por último, en esta fase se revisarían las llamadas defensas equitativas ("*equitable defenses*"). La *segunda fase* cubriría las infracciones sobre patentes de invención. Y, por último, en una *tercera fase* se tratarían los daños ("*damages*"). Entonces, así para la primera fase se acordó que el Juez Alsup decidiría la cuestión de "*copyrightability*" (es decir, si es o no protegible por derechos de autor de los EE.UU. la tecnología en cuestión) y las defensas equitativas de Google, y el Jurado ("*jury*") decidiría los asuntos de infracción a los derechos de autor y el uso justo ("*fair use*"). Además, el Jurado, y si lo hubiera, también debería revisar acerca de cualquier otra copia literal sin autorización catalogada como de "*minimis*" o de menor cuantía o de menor cantidad de material protegido. A su vez, en relación a la determinación de si había existido infracción de derechos y en su caso sobre la existencia de "uso justo", el Jurado fue instruído por el Juez Alsup a que estos *asumieran* como

Con fecha 7 de mayo de 2012, el Jurado emitió un veredicto por el cual encontró que Google había infringido los derechos de autor de Oracle en relación a los 37 paquetes de Java[71].

---

cierto que "la estructura, la secuencia y la organización de los 37 paquetes de la librería de software estándar de Java "como un todo" ("*as a whole*") era protegible bajo la ley de los derechos de autor de EE.UU. (*Copyright Law Act 1976*). Por supuesto que esto no era una decisión final, ni mucho menos, sino que el Jurado fue instruído de dicha forma meramente por una cuestión de costos del proceso ante una supuesta sentencia en contra (de la decisión del Jurado) de primera instancia (donde no se reconocería como protegible a la estructura secuencia y organización) y una sentencia de la Corte de Apelaciones en contra de dicha sentencia de 1era instancia por la cual se mandaría a reinstalar la decisión del Jurado. (haciéndoselo de esta forma se evitaría un nuevo juicio (*re-trial*) y los mayores costos que esto ocasionaría. En la fase 1 teniendo en cuenta las pruebas aportadas por ambas partes en relación al código compilable para los 37 paquetes de la librería estándar *de Java (*o *Class Library*) el Jurado entendió que Google había infringido los derechos de autor de Oracle, pero no emitió resolución acerca del uso justo o "*fair use*". En relación a la documentación de la *Java Standard Library* el Jurado dictamino que no había encontrado infracción de derechos. Finalmente, como último punto de esta fase 1 en relación a ciertas porciones o fragmentos de código ("*snippets code*") el Jurado sólo encontró que nueve (9) líneas de código llamado "rangeCheck" estaban en infracción. En relación a la fase 2 el Jurado no encontró que haya existido infracción de patentes de invención.

[71] El Jurado tenía instrucciones del Juez del Distrito que el veredicto se focalizara en la SSO de los paquetes de las API de Java, ya que no hacía falta que el Jurado se expidiera acerca del "código fuente declarado" por cuanto Google había aceptado que lo había copiado literal, textual. También el Jurado encontró que Google había infringido los derechos de Oracle en nueve (9) líneas de código de rangeCheck, y por otro lado entendió que no había existido infracción en relación a ocho (8) archivos de seguridad compilados. El Jurado no se expidió sobre el asunto de uso justo o "*fair use"*. Oracle interpuso una moción por infracción en relación a los 8 archivos descompilados -una suerte de revisión de la decisión del Jurado que resuelve el Juez del Distrito- (JMOL-*Judgement as a matter of Law*) Véase en https://www.law.cornell.edu/rules/frcp/rule_50 En relación a los 8 archivos descompilados objeto del JMOL de Oracle., el Juez de Distrito sostuvo que

A su vez, el mismo Jurado con fecha 23 de mayo de 2012 entendió que Google no infringía los derechos de patentes de Oracle[72] en relación a las reivindicaciones 11, 27, 29, 39, 40 y 41 de la patente RE 38104, y de las reivindicaciones 1 y 20 de la patente 6.061.520.

Por otra parte, el 31 de mayo y 20 de junio de 2012, el Juez de la Corte del Distrito Norte de California, Willian Alsup[73] dicto sentencia, reversando el veredicto del Jurado en cuanto expresó que los paquetes de las API de Java y su estructura, secuencia y organización no eran protegibles bajo el sistema de derechos de autor de EE.UU.

William Alsup recalcó que el presente caso se encuentra regido por los siguientes ***principios*** aplicables:

(i) Bajo la doctrina de fusión (en inglés *"merge doctrine"*) cuando hay una sola manera (o muy pocas) de expresar algo, entonces nadie puede reclamar titularidad sobre esa expresión bajo la ley de derechos de autor de EE.UU.;

(ii) Bajo la doctrina de nombres, los nombres y las frases cortas no son protegibles bajo la ley de derechos de autor de EE.UU.;

Google copió los archivos en su totalidad, que del testimonio rendido durante el juicio revelaba que la utilización de esos archivos era significativa y por último que un ningún Jurado razonable podía encontrar la copia como "de minimis". En definitiva, el Juez reversó la decisión del Jurado y condenó a Google por la copia de los ocho (8) archivos descompilados.

[72] Véase *JURY VERDICT (Phase II). (dt, COURT STAFF) (Filed on 5/23/2012) (Additional attachment(s) added on 2/20/2019: #1 Special Verdict Form) (amgS, COURT STAFF)* en https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/1190/0.pdf?ts=1376386946

[73] Véase en https://www.cand.uscourts.gov/judges/alsup-william-wha/

(iii) Bajo la sección 102, inciso b) la protección otorgada por los derechos de autor de EE.UU. *no se extiende a las ideas, procedimientos, procesos, sistemas, métodos de operación o conceptos sin importar su forma*. Los elementos funcionales esenciales para la interoperabilidad no son protegibles bajo la ley de derechos de autor de EE.UU., y, por último,

(iv) Bajo la teoría del caso *Feist*, indicándose que no se debería reconocer protección por el sistema de derechos de autor de EE.UU. simplemente para recompensar una inversión realizada.

FUNDAMENTOS DE LA SENTENCIA DE LA CORTE DE DISTRITO DEL NORTE DE CALIFORNIA, JUEZ WILLIAM ALSUP[74] .

Menciona el Juez de la Corte de Distrito que, en el mes de agosto del año 2010, Oracle demandó a Google por infracción a sus derechos de autor y a sus derechos de patentes de invención.

Para ello, Oracle argumentó que la plataforma Android infringía los derechos de autor y de patentes de invención que Oracle poseía sobre la tecnología Java.

Específicamente, en relación a los derechos de autor, Oracle argumentó que Google infringía sus derechos en relación a 37 paquetes de la librería estándar de Java o Java API.

Los argumentos brindados por el Juez de la Corte del Distrito para *rechazar la demanda de Oracle* fueron los siguientes.

[74] Véase https://www.cand.uscourts.gov/judges/alsup-william-wha/

IMPLEMENTACIÓN INDEPENDIENTE POR PARTE DE GOOGLE

Un punto central que remarca la sentencia de primera instancia es que ambas partes estuvieron de acuerdo en que Google no había copiado en forma "literal" el software bajo disputa, sino que, por el contrario, que Google había realizado su propia implementación de los 37 paquetes de la librería estándar de Java ("Sun Java Standard Library" o "Class Library").

ESTRUCTURA, SECUENCIA Y ORGANIZACIÓN

El Juez de la Corte de Distrito menciona que el reclamo neurálgico de Oracle "debería ser" que Google copió la estructura, secuencia y organización del total de los 37 paquetes de la librería estándar de Java.

Ello por cuanto se entendió que mientras los nombres no están protegidos por el sistema de derechos de autor de EE.UU., el sistema de nombres organizados de Java -que abarca a los 37 paquetes con sus 600 clases y sus más de 6000 métodos- conforma una estructura organizacional o taxonomía, protegible bajo la ley de derechos de autor de EE.UU.

Ahora bien, por otro lado, el hecho de que un método de operación posea miles de *comandos* seleccionados en una *taxonomía creativa* no le cambia el carácter de método de operación.

El Juez de Distrito expresó: *"Si es creativo. Si es original. Sí se asemeja a una taxonomía. Pero constituye una estructura de comandos, un sistema o método de operación. Por esta razón no*

puede ser protegido por el derecho de autor -tal vez por el sistema de patentes".

Es decir, la Corte de Distrito reconoció que la estructura, secuencia y organización de las API de Java era original y creativa, pero menciona que "la estructura, secuencia y organización de comandos" no es más que un método para operar, un método de operación, que se encuentra excluido del sistema de protección de derechos de autor de EE.UU. (*US Copyright Law*, Titulo17, sección 102, inciso b).

También al respecto de la estructura de las API se expresa que la *misma funcionalidad* podría haber sido ofrecida en Android sin haberse tenido que copiar la misma estructura de comandos usados en Java.

Es más, podría haber sido hecho mediante una suerte de reorganización de los métodos bajo diferentes grupos entre las distintas clases y paquetes de Java -aún, usando los mismos nombres que en Java-.

También en el momento de la creación de Java por parte de Sun Microsystems había muchas maneras de agrupar a los métodos, y, así y todo, se podría haber duplicado el mismo rango de funcionalidad.

De todas formas, teniendo en cuenta que la estructura de comandos es un sistema o un método para operar conforme a lo establecido en la sección 102 b) de la ley de derechos de autor de EE.UU. no puede ser protegida.

La sección 102, inciso b) menciona lo siguiente: "...en ningún caso la protección del derecho de autor para las obras originales se extiende a las ideas, procedimientos, procesos, sistemas, métodos de operación...sin importar la forma..."

Y por último al respecto de su exclusión de protección, se dijo que no es menos cierto que los nombres *serían más que nombres*, *así serían una suerte de símbolos* en una estructura de comandos en donde los comandos toman la forma de expresión de ***java.package.Class.method()***.

Entonces cada comando posee una función pre-asignada.

En definitiva, el "árbol de nombres", posee un elemento creativo, pero es también una estructura de comandos, una suerte de conjunto utilitario y funcional de símbolos, cada uno de ellos con una función pre-asignada.

Y por ello su exclusión de la protección de la ley de derechos de autor.

NO IMPORTA QUE TAN CREATIVO O IMAGINATIVO PUEDA SER UN MÉTODO DE JAVA, CUALQUIERA TIENE DERECHO A UTILIZAR LA MISMA ESPECIFICACIÓN DEL MÉTODO (INPUTS, OUTPUTS, PARÁMETROS) SIEMPRE Y CUANDO Y LÍNEA POR LÍNEA, EL CÓDIGO FUENTE IMPLEMENTADO SEA DIFERENTE

Siempre que el código que se utilice para implementar un método de Java sea diferente, cualquier persona puede conforme al derecho de autor de EE.UU. escribir su propio código fuente para llevar a cabo exactamente la misma función o especificación de cualquier método de Java utilizado en la librería estándar de Java**.**

Expresa la Corte del Distrito que contrariamente a lo que sostiene Oracle, el sistema de derechos de autor EE.UU. no confiere derechos sobre todas y cada una de las formas de llevar a cabo la implementación de una función o especificación, ello

es así, sin importar que tan creativo pueda ser la implementación o especificación.

Agrega que la ley sólo confiere derechos sobre la manera específica en que un autor ha creado su propia versión.

Por ende, cualquier otro individuo puede desarrollar o escribir su propia implementación para llevar a cabo o cumplir con idéntica funcionalidad, por lo tanto, las ideas, conceptos y funciones no pueden ser monopolizadas por el sistema de derechos de autor de los EE.UU.

Al respecto del ejemplo del método de Java que se ha mencionado anteriormente, en donde una función de comparar dos números y devolver como resultado el mayor de ellos, el Juez menciona que Google, -y cualquier otro- tiene derecho a escribir su propio código para llevar a cabo la misma función siempre que el código implementado en el cuerpo del método sea diferente de la implementación protegida (la implementación protegida a la que se refiere es a la realizada por Sun Microsystems, Inc., es decir la *Java Class*).

IMPOSICIÓN DE LOS "HEADERS" POR REGLAS DEL LENGUAJE DE PROGRAMACIÓN JAVA

No importa que las declaraciones (en inglés "*declarations"*[75]) o las líneas de encabezados (en inglés *"headers"*) de un método

[75] El término declaración *(declarations)* fue utilizado durante el trámite del juicio (*trial*) para describir a los *headers* o encabezados (es decir código no implementado) para los métodos y las clases de Java. El Juez Alsup reconoce que desde el punto de vista terminológico *headers* sería el vocablo técnico apropiado, pero que en su sentencia de primera instancia y a los efectos de ser consistente con el vocabulario utilizado durante el juicio se utiliza encabezados (*headers)* y declaraciones (*declarations)* en forma indistinta.

sean idénticas, ya que, conforme a las reglas del lenguaje de programación Java deben ser idénticas para declarar el mismo método especificando la misma funcionalidad, aún, cuando su implementación sea diferente.

LOS MÉTODOS DE JAVA COMO "MÉTODOS DE OPERACIÓN" PODRÍAN SER PROTEGIDOS BAJO EL SISTEMA DE PATENTES DE INVENCIÓN

Agrega la Corte de Distrito que gran parte de la prueba rendida en el juicio por Oracle se enfocó en mostrar que los métodos en la *Java Class* tenían un esfuerzo creativo: Alsup, menciona que es cierto, que es así, y agrega que inventar un nuevo método de Java *(*en inglés *"Java method")* para entregar un nuevo resultado (en inglés "*output"*) puede ser creativo, incluso puede tener altura inventiva, incluyendo las opciones del *input* necesario y sus *outputs*.

Lo mismo aplica para las clases de Java *("Java Classes")*.

Ahora, tales invenciones -en cuanto al concepto y al nivel funcional- son protegidas únicamente por el derecho de patentes de invención.

DOCTRINA DE LA FUSIÓN ("MERGE DOCTRINE")

Se menciona que cuando existe una sola forma de expresar una idea o una función, cualquier persona es libre para expresar esa idea, y nadie puede monopolizarla, justamente porque existe una sola forma de expresarla.

Y si bien los métodos de Android y los nombres de clases *(*en inglés *"class name")* podrían haber sido diferentes de aquellos nombres utilizados por Oracle en Java -y aún, así habrían funcionado de igual forma- el derecho de autor de EE.UU. no se extiende, y por ende no protege, a los nombres o las frases cortas.

INTEROPERABILIDAD

Al respecto de este punto, opina el Juez de grado que la duplicación de la estructura de comandos es necesario para la "*interoperabilidad*" diciendo que seguramente millones de líneas de código han sido escritas en lenguaje Java antes que Android existiera.

Estos programas necesariamente utilizaron el formato de comando ***java.package.Class.method()***.

Así estos programas llamados o invocados en todos o algunos de los 37 paquetes en cuestión, necesariamente usaron los comandos y su estructura de nombres.

Ese código era de propiedad de los propios programadores (terceros), y no de Oracle.

En el caso en cuestión, para que al menos parte de este código corriera en Android, Google era requerido de proveer el mismo idéntico sistema de comando ***java.package.Class.method()*** utilizando los mismos nombres asignados con la misma taxonomía y con la misma funcionalidad.

Por lo tanto, concluye el Juez de la Corte del Distrito que Google replicó aquello que era necesario para tener un grado de interoperabilidad -pero no más allá de ello, creando y desarrollando así su propia implementación independiente.

FRAGMENTACIÓN (O "BALKANIZATION"). LOS CASOS SONY Y SEGA

En relación a la denominada "fragmentación" o *"balkanization"*, y de porque los casos judiciales de *SONY y SEGA* son similares a este caso, se expresa que la interoperabilidad se encuentra en el corazón de la estructura de comandos, ilustrada por la misma preocupación de Oracle en lo que denomina "fragmentación", queriendo referirse al problema de tener una *"interoperabilidad imperfecta"* entre plataformas.

Cuando esto ocurre, las aplicaciones basadas en Java no pueden correr o ejecutarse en plataformas que no son compatibles.

Sun Microsystems y Oracle han tratado de evitar esta *balkanization* a través de sus programas de licenciamiento (recuérdese que existían tres (3) modelos: a) licencia GPL v2 más *ClassPath Exception* ("*open source"*), b) licencia de especificación, y c) licencia comercial.

Expresa el Juez de Distrito, que Oracle ha dejado la impresión que sí Google hubiese copiado *"todos"* los 166 paquetes de la librería estándar de Java, Oracle no habría demandado a Google.

Recordemos que Oracle demandó a Google soló por la infracción de 37 paquetes, por lo tanto, no hay reclamo de infracción de derechos de autor por los restantes 129 paquetes que conformaban la librería estándar de Java en la edición 2 de la plataforma Java al momento de la demanda.

Mientras la "fragmentación" es una consideración de negocio legítima de ser considerada hace nacer la pregunta de si una

licencia era o no requerida para *replicar "parcial o totalmente" la estructura de comando* (claro y sólo si y en la medida de que Android no llevara la marca registrada Java, y que Google no sostuviera que Android es totalmente compatible con Java).

Por otro lado, las decisiones judiciales del Circuito Noveno de SEGA y SONY, tienen cierta similitud con el caso en cuestión, ya que en estas dos decisiones judiciales se dijo que los *"procesos de interfaces requeridos o necesarios"* para la interoperabilidad fueron considerados como *"requerimientos funcionales para la compatibilidad"* y se los consideró como no protegibles por el derecho de autor de EE.UU. conforme lo establecido en la Sección 102, inciso b).

Agrega el Juez que en ambas decisiones judiciales se sostuvo que los *procesos de interface* que eran necesarios de ser duplicados con el objetivo de garantizar la compatibilidad fueron considerados "aspectos funcionales" y, por lo tanto, no protegibles a la luz de la sección 102, inciso b) de la ley de derechos de autor de los EE.UU.

Por tanto, sostiene la Corte de Distrito que en el caso en cuestión, y en el mismo sentido reflejado anteriormente, la estructura de comandos para los 37 paquetes de Java es análoga a los casos de SEGA y SONY diciéndose que si alguien pudo replicar las interfaces de la BIOS de SONY con el objetivo de hacer correr los juegos de la Playstation en computadoras de escritorios (escribiendo su propia implementación), entonces Google podía duplicar la estructura de comandos de los 37 paquetes en Android con el objeto de acomodar el código fuente de terceros confiando en los 37 paquetes (de nuevo con su propia implementación).

Por ello dice la Corte de Distrito, que *"la compatibilidad total"* no es relevante a los efectos del análisis de la Sección 102,

inciso b), y agrega que en el caso SONY, el producto acusado de infracción únicamente implementó 137 funciones de las 242 funciones de la BIOS de la PlayStation porque eran las únicas funciones invocadas por los juegos.

Por ello la Corte de Apelaciones en ese caso sostuvo que el producto acusado de infracción no infringía o violaba la ley de derechos de autor de EE.UU.

Al respecto la Corte de Distrito expresa que ello traza un paralelo con la decisión que tomó Google de implementar "*sólo algunos, pero no todos*" de los paquetes de la librería estándar de Java en Android.

RAZONES POR LAS CUALES EL CASO AMERICAN DENTAL ASSOCIATION (ADA) V. DELTA DENTAL PLANS ASSOCIATION (DECISIÓN JUDICIAL DEL CIRCUITO SÉPTIMO) NO CONTROLA EL PRESENTE CASO JUDICIAL

Asumiéndose que en el Circuito Noveno la taxonomía es materia protegible del derecho de autor (y cita como referencia el caso *Practice Mgmt. Info v. Am. Med. Ass´n,* del Circuito Noveno del año 1997), se expresa que la taxonomía en ADA no tenía nada que ver con programas de computación, y que no se trataba de un sistema de comandos, y mucho menos de un sistema de comandos para un lenguaje de computación.

Así, la taxonomía en ADA subdividía todos los procesos dentales dentro de un esquema de categorías numeradas con una descripción en idioma inglés creada por ADA.

Esto era usado posteriormente por las compañías de seguro y dentistas para facilitar la facturación.

Por el contrario, la taxonomía en el caso *Oracle v. Google*, se compone en su totalidad por un sistema de comandos para llevar a cabo funciones de computadoras.

Agrega que, en el caso ADA, se sugirió que un "sistema", bajo la Sección 102 inciso b), debía venir con "instrucciones para uso".

Ahora bien, como la taxonomía en ADA no tenía instrucciones para ser usada, se consideró que no era un sistema -además y entre otras razones-.

Por el contrario, los paquetes de Java en cuestión vienen con instrucciones de uso, como ser la documentación y los comentarios embebidos.

Estos describen cada uno de los paquetes, clases y métodos, que *inputs* se necesitan, y que outputs se devolverá-la clásica forma de instrucciones de uso-.

PORQUE NO ES CORRECTA LA VISIÓN DE ORACLE RESPECTO DEL CASO JOHNSON CONTROLS V. PHOENIX CONTROL SYS[76] (CIRCUITO NOVENO, DEL AÑO 1989)

[76] En su sentencia el juez Alsup explica que el caso de Oracle v. Google que él lleva se rige por la ley del Distrito Noveno y por la Corte Suprema de los Estados Unidos. Ello es así conforme al sistema judicial de los Estados Unidos de América. Al respecto, la Corte Suprema menciona el caso *Feist Publications, Inc. V Rural Telephone Services Co., Inc., 499 U.S 340 (1991).* Este caso se refirió a la protección bajo derechos de autor de compilaciones de elementos puramente fácticos. En este caso la Corte Suprema de Justicia consideró que un directorio telefónico comprendido por nombres, direcciones, números telefónicos organizados en orden alfabético era protegible bajo la ley de derechos de autor. En este caso la Corte de Justicia rechazó la teoría de que el sistema de derecho de autor de EE.UU. estaba

destinado a recompensar a los autores por el "sudor de su frente" *("sweat of the brow")*. La Corte con esto dijo que no se debería conceder protección bajo la ley de derechos de autor de EE.UU. simplemente porque haya habido demasiado "sudor de la frente" dentro de la obra. Así se concluyó que la protección sólo se extendía a los componentes originales de una obra de un autor, y dijo: *This inevitably means that the copyright in a factual compilation is thin. Notwithstanding a valid copyright, a subsequent compiler remains free to use the facts contained in another´s publication to aid in preparing a competing work, so long as the competing work does not feature the same selection and arrangement"*. En relación al Circuito Noveno, la Corte de Apelaciones reconoció que los elementos no literales de un programa, incluyendo su estructura, secuencia y organización y la interface de usuario pueden ser protegidos bajo la ley del derecho de autor de EE.UU. dependiendo de si la estructura, secuencia y organización de que se trate califica como "*la expresión de una idea en lugar de una idea en sí misma". (Johnson Controls, Inc. V. Phoenix Control Sys, Inc. 886, F.2d 1173, 1175 (9th Cir. 1989)*. Alsup menciona que la decisión de Johnson es uno de los pilares en los cuales Oracle apoya su postura. Johnson Controls había desarrollado un programa de computación que controlaba un sistema para el tratamiento de aguas residuales. En la primera instancia la Corte del Distrito reconoció que la estructura, secuencia y organización del programa constituía una "expresión" y por ende protegible, y por ello concedió una medida cautelar (*preliminary injuction*), a pesar de que el producto de software acusado de infracción no poseía código fuente o código objeto similar. La Corte de Apelaciones confirmó la medida cautelar dictada en primera instancia por la Corte del Distrito expresando que el programa de software de la parte demandante era totalmente sofisticado y cada aplicación individual estaba adaptada a las necesidades del comprador**.** Entonces, teniendo en cuenta que había existido discreción y oportunidad para la creatividad en la estructura, se consideró que la estructura del programa constituía expresión y no sólo una idea**.** La empresa Johnson Controls, no mencionó sobre que particular estructura merecía protección. Alsup también recorre los casos *Brown Bag Software v. Symantec Corp., 960 F.2d 1465 (9th Cir. 1992), Atari Games Corp v. Nintendo of America Inc., 975 F.2d 832 (Fed. Cir. 1992)* en donde el Circuito Federal interpreta los precedentes del Circuito Noveno, *Sega Enterprises Ltd v. Accolade, Inc., 977 F.2d 1510 (9th Cir, 1992)* y *Sony Computer Entertainment, Inc. V. Connectix Corporation, 203 F.3d 596 (9th Cir. 2000):* en este caso citándose la sección 102, inciso b) y el caso SEGA, la Corte de Apelaciones sostuvo que la BIOS de la PlayStation contenía elementos funcionales no protegibles por el derecho de autor de EE.UU., y concluyó que el paso intermedio acusado de infracción de copiar código

Se sostiene que no es correcta la visión de Oracle de lo resuelto en el caso *Johnson Controls V. Phoenix Control Sys*.

Así se dice que en el Circuito Noveno (que es el Circuito al cual pertenece la Corte del Distrito de California a cargo de William Alsup) la estructura, secuencia y organización de un programa de computación puede o no calificarse como un elemento protegible dependiendo "de los hechos particulares de cada caso" y siempre sujeto a la exclusión de elementos no protegibles.

Explica que contrariamente a lo que sostiene Oracle, el caso *Johnson Controls* no sostuvo que toda estructura, secuencia y organización en un programa de computación estaba dentro del ámbito de protección del derecho de autor de EE.UU.

Por el contrario, en ese caso se encontró que la estructura, secuencia y organización -del programa que en sí mismo se entendía protegido- teniendo en cuenta los hechos particulares del caso, merecía protección bajo la ley de derechos de autor de EE.UU.

El Juez Alsup expresa que las circunstancias del caso *Johnson Controls v. Phoenix Control* no son las mismas circunstancias del caso *Oracle v. Google.*

PORCENTAJES

objeto constituía "uso justo" ya que había sido realizado con el propósito de obtener acceso a elementos no protegibles -por el derecho de autor- del software de SONY.

Del total de los 166 paquetes en Android, existen 129 paquetes por los cuales no se ha demandado por infracción derechos.

De los 37 paquetes demandados, el 97% de las líneas de código de Android fueron escritas por Google, y respecto del 3 % restante expresa la Corte del Distrito que eran replicables bajo la teoría de la fusión y la teoría de nombres.

ELEMENTOS PARTICULARES DEL CASO

La sentencia no expresa que los paquetes de la librería estándar de Java pueden ser usados sin licencia alguna, como tampoco sostiene que la estructura, secuencia y organización de todos los programas de computación pueden ser "robados".

Muy por el contrario, sostiene que *"ante los hechos específicos del caso, los elementos particulares copiados por Google podían ser utilizados bajo la ley de derechos de autor de EE.UU."*.

Durante el mes de febrero del año 2013 se presentaron diferentes opiniones de los *Amicus Curiae* (*amigos del tribunal*) soportando o apoyando la posición de cada una de las partes litigantes.

Asimismo, fueron presentadas las apelaciones correspondientes.

CAPITULO VI

RECURSO DE APELACIÓN INTERPUESTO POR ORACLE CONTRA LA SENTENCIA DE LA CORTE DE DISTRITO RELACIONADOS A LA PROTECCIÓN DEL CODIGO FUENTE DECLARADO Y DE SU ESTRUCTURA, SECUENCIA Y ORGANIZACIÓN BAJO EL SISTEMA DE DERECHOS DE AUTOR DE EE.UU.

El Juez de la Corte de Distrito dijo en su sentencia en relación al código fuente declarado que no importaba lo creativo o imaginativo que éste pudiera ser si sólo existe "una manera de escribirlo", y, por lo tanto, la doctrina de la fusión (en inglés "*merge doctrine"*) impediría a cualquiera reclamar derechos exclusivos sobre ello.

Por lo tanto, no puede existir infracción de derechos de autor en la utilización de idénticas declaraciones o código fuente declarado.

Por otro lado, también mencionó la sentencia de primera instancia que el código fuente declarado consiste en nombres y frases cortas que no son protegibles bajo el derecho de autor.

Así mismo, la Corte de Distrito reconoció que la estructura, secuencia y organización de cada una de los paquetes de Java era "creativa" y "original", pero sin importar ello, sostuvo que dicha estructura, secuencia y organización no era protegible por derechos de autor conforme al artículo 102 inciso (b) del Título 17 USC, ya que constituía una *"estructura de comandos para un sistema o un método de operación*".

Conforme la posición de Oracle existe dos axiomas que definen el presente caso:

Axioma 1: El umbral de la ley de derechos de autor de EE.UU. para otorgar protección es "bajo" (*Feist Publ´ns, Inc. v. Rural tel. ser. Co -1991*). En efecto la ley de derechos de autor protege la "expresión original", y una obra es "original" si posee al menos un mínimo grado de creatividad. A su vez, Oracle obtuvo la presunción como de "obra original y protegida" mediante la registración de la plataforma Java en la Oficina de Derechos de Autor de EE.UU. Más allá de ello, los paquetes de Java son fácilmente protegibles teniéndose presente que exceden el grado de creatividad exigido por la ley.

Axioma 2: La protección bajo la ley de derechos de autor de EE.UU. se extiende a los programas de computación, de la misma manera que se lo hace para cualquier otra obra literaria. Así entonces, se protege a las obras literarias expresadas en palabras, números u otros símbolos verbales o numéricos. Los programas de computación cumplen con esa definición. La conclusión del Juez de la Corte de Distrito en relación a que existe "un tipo de software" (código fuente declarado) que esta por fuera de estas reglas constituye una violación a ambos principios.

Entonces, teniendo en cuenta estos dos axiomas, el código que Google copió está *protegido bajo dos principios independientes:*

En primer lugar, el código declarado es protegible porque es "expresivo". Google admitió en el juicio que copió en forma literal 7000 líneas de código. El código declarado copiado por Google es protegible como cualquier otra obra literaria, ya que Oracle ejerció "creatividad en la selección y organización de sus líneas de instrucción".

En caso similares, donde se reconoció que había existido distintas alternativas o maneras de obtener el mismo resultado se sostuvo que la estructura de código copiado era protegible.

Por otro lado, cada vez que Oracle desarrolló código -incluso código declarado- se comienza con "una pizarra limpia". No hay "convenciones de programación ni reglas dictadas por el lenguaje de Java" acerca de lo que el código declarado debería ser.

La decisión de Google de no copiar el código implementado por Oracle, y de si copiar el código declarado por Oracle, no transforma a este último en no protegible. Ningún copista puede excusar su conducta inapropiada al mostrar cuanto de su obra no ha pirateado. Google intenta hacer parecer trivial las 7000 líneas de código que copió.

Ahora lo copiado por Google es lo más importante para los programadores de lo que luego Google *parafraseó,* que fue su código implementado.

De hecho, los programadores no conocen al código implementado, tampoco les es de interés, todo lo que los programadores saben y necesitan saber para programar o construir sus propias aplicaciones es el código fuente declarado de Oracle.

En segundo lugar, la estructura, secuencia y organización de los paquetes de Java fue reconocida en el juicio como original y creativa, por lo tanto, es protegible bajo la ley de derechos de autor, y tiene protección en dos sentidos:

a) porque el código declarado incorporado en ella, es protegible, y a su vez, y

b) en forma independiente, porque la misma estructura, secuencia y organización es protegible independientemente que el código declarado incorporado a ella lo sea.

Con lo cual, aún si Google no hubiera copiado el código declarado de Oracle, y si la estructura, secuencia y organización, como lo hizo, de todas formas, también hubiese existido infracción de derechos.

En la construcción de la estructura y organización de los paquetes de Java existieron infinitas opciones para Oracle, y, además de ello, se la construyó de manera tal que los programadores la encontraran atractiva y muy fácil de recordar.

A su vez, Google reconoció que la estructura, secuencia y organización de los 37 paquetes de las API incluidas en Android es sustancialmente la misma que la estructura, secuencia y organización de los 37 paquetes de las API de Java.

Por otro lado, la estructura, secuencia y organización sería igualmente protegible por los derechos de autor, aunque Google no hubiese copiado una sola línea de código. (y más lo sería como en el caso teniendo en cuenta que ha copiado los elementos que la conforman con sus 7000 líneas de código).

En definitiva, los elementos no literales del programa de computación, que incluye la estructura, secuencia, y organización pueden ser protegidos por derechos de autor donde existe una expresión creativa.

Por ello, en el caso, conforme constancias del juicio, Oracle tuvo infinitas opciones para organizar y seleccionar los distintos programas dentro de los paquetes de Java, con lo cual *"la idea de cómo organizar cada paquete no se fusiona como su expresión",* ya que como se dijo existieron múltiples formas de

poder llevarse a cabo, por ello el artículo 102, inciso b) no es aplicable.

Por lo demás, el razonamiento del Juez de Distrito cuando dice que ni una sola línea de código declarado puede recibir protección bajo derecho de autor, ya que constituyen comandos para llevar a cabo funciones preasignadas, si ello fuese correcto, ningún programa de computación estaría protegido, ya que esa es la definición de programa de computación conforme a la ley, cuando dice *"conjunto de declaraciones o instrucciones para ser usadas directa o indirectamente en una computadora con el propósito de llevar a cabo un resultado"*.

Entonces, si un programa de computación no es protegible por derechos de autor por la simple razón de que posee el propósito de llevar a cabo funciones preasignadas, ningún programa de computación sería protegible.

De hecho, esto se encuentra en contradicción con lo dispuesto en casos judiciales anteriores, donde se ha mencionado que se reconoce la protección del código fuente aún en los casos que consistan en comandos para llevar a cabo funciones preasignadas.

El Juez de Distrito menciona como aplicable el caso de *Lotus Development v. Borland International, Inc*.

Ahora ese caso, no tiene relación con el caso en cuestión, ya que el demandado sólo copio un sistema simple de menú de comandos, y no código.

Entonces, en el caso *Lotus*, se estableció una definición de método de operación muy expansiva que dice *"cualquier medio por el cual una persona opera algo"*, pero esa decisión no puede ser aplicada más allá de los hechos de ese caso, ya que de lo contrario le quitaría la protección bajo derechos de autor a todos

los programas de computación, y ello, iría en contra de lo sancionado por el Congreso de los EE.UU.

En la actualidad ningún Circuito de los EE.UU. adopta la fórmula del caso *Lotus*.

Por otro lado, en el caso *Atari*, se encontró infracción aún, cuando no hubo copia de *elementos literales* del programa de computación.

Atari había copiado la estructura organizacional y secuencial de la función de seguridad de *Nintendo*. Y se determinó que como los programadores de *Nintendo* tuvieron varias maneras disponibles para llevar a cabo la función de seguridad, *Nintendo* ejerció creatividad en su selección.

En relación a la doctrina de la fusión *(*en inglés *"merge doctrine")*, la Corte de Distrito se equivoca al aplicar esta doctrina en relación al código declarado cuando menciona que *"no importa lo creativo o imaginativo que sea si existe una sóla manera de escribirlo"*, y que, por lo tanto, la doctrina de la fusión impide a cualquiera reclamar derechos exclusivos sobre ello.

Ahora, dicha doctrina no es aplicable a ninguna línea individual del código declarado a menos que sus autores originarios (Oracle) hayan tenido disponible *"una sola forma o manera"* -que no fue el caso- de escribir esas líneas de código, en el caso, de las 7000 líneas de código fuente declarado.

Entonces, como como surge de las constancias del juicio, Oracle tuvo infinidad de opciones para cada una de las líneas individuales y también tuvo una infinidad de opciones en la selección y organización de las 7000 líneas de código que Google copió.

Por ejemplo, el Juez menciona *"**java.lang.Math.max**"*.

Los desarrolladores de Sun Microsystems podrían haberlo llamado ***"Math.maximun", "Equations.compare", "Arith.bigger, "MeasuringStick",*** etc.

A su vez, quedó demostrado que "la idea y la expresión no se fusionaron" cuando fue el mismo Juez del Distrito que expresó que los *métodos y las clases de Android* podrían haber tenido un nombre diferente que *los métodos y las clases de Java*, y aun así hubieran funcionado igual.

En definitiva, Oracle tuvo infinidad de posibilidades acerca de que métodos incluir y de cómo organizarlos.

De hecho, también ha sido reconocido por el Juez que conceder protección por derechos de autor no hubiese impedido que Android provea las mismas funciones, pero organizadas de una manera diferente, y esto podría haber sido hecho mediante la reorganización de varios métodos dentro de distintos grupos dentro de varias clases y paquetes (aunque incluso se hubiesen usado los mismos nombres).

En este sentido, había muchas formas de agrupar a los métodos y así duplicar el mismo rango de funcionalidades.

De hecho, en unos 100 paquetes de Android Google escribió su propio código declarado, con su propia estructura y organización.

Pero no hizo lo mismo con los 37 paquetes de Java en cuestión.

Entonces, como varias expresiones alternativas existen disponibles, la teoría de la fusión no es aplicable.

Adicionalmente, tanto respecto de las *líneas individuales* de código declarado o de *todo el cuerpo* del código declarado, las opciones disponibles son las que poseía el autor de la obra

original (Oracle) en el momento de su creación, y no, las opciones que podría haber tenido el copista al momento de copiar.

El Juez del Distrito menciona que para llevar a cabo una determinada función la especificación del método establecida en la declaración o código declarado debe ser *idéntica*.

Ahora bien, una vez que el autor de una obra original (en el caso Oracle) eligió crear y denominar a un método como *"**java.lang.Math.max***", los programadores que quieren utilizar el programa pre-escrito de Oracle lógicamente tienen que llamarlo por su nombre.

Ahora, esto no significa que la obra original no pueda ser protegida por el derecho de autor.

Qué es lo que el copista cree que necesita copiar de la obra original para su beneficio, es irrelevante, porque el derecho de autor subsiste desde el momento mismo de creación y perdura conforme a los plazos establecidos en la ley.

Por otro lado, en relación a las *"frases cortas"* el Juez del Distrito menciona que cada una de las líneas del código declarado en forma aislada no son protegibles.

El Juez para ello invoca una regulación de la Oficina de Derechos de Autor de EE.UU., la cual se refiere a "obras" que estén un tanto vacío de texto. En definitiva, ello quedaría cubierto por el derecho de marcas.

Ahora bien, por el contrario, lo anterior sobre "frases cortas" no elimina de protección al ensamble de 7000 líneas de código.

El error se encuentra en querer *diseccionar la obra* en forma demasiado minuciosa, poniéndose foco en la línea individual del

código, en lugar de ponerlo en el *complejo arreglo del cual forma parte*.

De hecho, aplicar la teoría de las frases cortas como lo propone el Juez del Distrito, significaría que virtualmente ningún programa de computación estaría protegido.

Por ejemplo, el código fuente declarado en el método de Java en ***java.security.cert.package*** es el siguiente:

public abstract void verify (PublicKey key, String sigProvider)
throws CertificateException, NoSuchAlgorithmException,
InvalidKeyException, NoSuchProviderException
SignatureException

O del código fuente declarado en la clase ***java.nio.channels****:*

public abstract class DatagramChannel
extends AbstractSelectableChannel
implements ByteChannel, ScatteringByteChannel, GatheringByte
Channel {

Aún en el caso de un “código fuente declarado” más corto que los ejemplos mencionados más arriba, no necesariamente quedarían desprotegidos.

En definitiva, es equivoca la posición del Juez del Distrito en relación a la aplicación de la “teoría de frases cortas”, y, por lo tanto, explica Oracle no es aplicable al presente caso.

CAPITULO VII

SENTENCIA I DEL TRIBUNAL DE APELACIONES DEL CIRCUITO FEDERAL DE EE.UU.[77]

PROTECCIÓN DE LAS API DE JAVA Y DE SU ESTRUCTURA, SECUENCIA Y ORGANIZACIÓN

Con fecha 9 de mayo de 2014 el Tribunal de Apelaciones del Circuito Federal[78] integrada por los jueces O´Malley, Plager y Taranto determinaron que el código fuente declarado y la estructura, secuencia y organización de los 37 paquetes de las API de Java estaban sujetas a protección bajo las leyes del derecho de autor de EE.UU.

Es decir, la sentencia del Tribunal de Apelaciones del Circuito Federal reversó la sentencia dictada en la primera instancia por la Corte de Distrito del Norte de California en materia de protección bajo el derecho de autor del código fuente declarativo de los 37 paquetes de las API de Java y de su estructura, secuencia y organización, ordenándose *reestablecer la decisión del Jurado* que había determinado oportunamente la protección de la estructura, secuencia y organización de las API de Java bajo el sistema de derecho de autor de EE.UU.

Por otra parte, el Tribunal de Apelaciones del Circuito Federal mandó a que el Jurado tratara el asunto de *"fair use"*.

Tal como se había mencionado durante el trámite del pleito, el Tribunal de Apelaciones del Circuito Federal mencionó que

---

[77] Véase en http://www.cafc.uscourts.gov/sites/default/files/opinions-orders/13-1021.Opinion.5-7-2014.1.PDF

[78] Véase en http://cafc.uscourts.gov/

cada paquete de Java posee dos tipos de código fuente, y que las partes lo han llamado código declarado (en inglés "*declaring code"*) y código implementado (en inglés "*implementing code"*).

Código declarado es la expresión que identifica a las funciones pre-escritas y que se lo ha denominado como "declaración" o *header* (lo que es lo mismo que "encabezado").

El *header* introduce el cuerpo del método *("method body"*) y especifica con precisión el *input*, el nombre y otras funcionalidades.

A su vez, las expresiones utilizadas por un programador desde el código declarado instruyen a la computadora para ejecutar el código implementado asociado, el que le da a la computadora las instrucciones del paso a paso para llevar adelante la función declarada.

El Tribunal de Apelaciones del Circuito Federal menciona que Google adquirió la empresa Android, Inc. en el año 2005 con la idea de desarrollar una plataforma para teléfonos móviles inteligentes.

En el mismo año, Google y Sun Microsystems comenzaron a discutir la posibilidad de que Google tomara una licencia para usar y adaptar la plataforma Java para dispositivos móviles.

De hecho, las partes trataron la posibilidad de tener un acuerdo de colaboración bajo el cual la tecnología Java sería "una parte del código abierto" de la plataforma Android.

El punto de desencuentro entre las partes era la negativa de Google de hacer la implementación de sus programas compatibles con la máquina virtual de Java (JVM) o hacerlos interoperables con otros programas de Java.

Oracle sostuvo que la posición de Google era contraria al principio de *"write once, run anywhere"*, y, por ende, decidió no otorgar una licencia sobre los paquetes de las API de Java a Google.

Posteriormente a ello, ya suspendidas las negociaciones entre las partes, Google decidió utilizar el lenguaje de programación Java para construir su propia máquina virtual llamada Dalvik ("*Dalvik Virtual Machine o DVM"*).

También, Google desarrolló su propia implementación para las funciones de los paquetes de la API de Java que eran claves o necesarias para los dispositivos móviles.

Así Google desarrolló su plataforma Android que incluyen 168 paquetes de API, 37 de los cuales corresponden a los paquetes de las API de Java en debate en el pleito.

En relación a los 37 paquetes de las API de Java en cuestión, dice el Tribunal de Apelaciones del Circuito Federal que Google creía que los desarrolladores esperarían encontrar el mismo *set* de las 37 funcionalidades en el mismo sistema de llamadas de Android bajo los mismos nombres utilizados por Java.

Por lo que, para obtener dicho resultado, Google copió en forma idéntica el código fuente declarado (en inglés "*declaring source code"*) de las API de los 37 paquetes de Java, insertando el código declarado en Android.

Al efectuarse ello, Google copió la estructura organizacional de todos los nombres de los métodos, clases, interfaces, y paquetes, cubriendo los 37 paquetes con sus 600 clases y con más de sus 6000 métodos de Java.

Por otra parte, se agrega que no hay dudas de que Google escribió su propia implementación[79] y, por otro lado, a pesar de que Android fue escrita en lenguaje Java, Google diseño Android para que no fuera compatible con la plataforma Java, es decir que una aplicación escrita para la plataforma Android no funcionara en la plataforma Java.

Menciona el Tribunal de Apelaciones del Circuito Federal que está claro que Google podía haber escrito o desarrollado sus propios paquetes de las API de Java usando el lenguaje Java, pero Google escogió no hacerlo.

En su lugar, Google copió 7000 líneas de código fuente declarado replicando el total de la estructura, secuencia y organización de los 37 paquetes de las API de Java.

Conforme Google:

(i) había una sola manera de escribir las declaraciones de los métodos de Java, y

(ii) la organización y la estructura de los 37 paquetes de las API de Java conforma una *"estructura de comandos"* que se encuentra excluida de protección conforme al artículo 102 inciso (b) de la ley de derechos de autor de EE.UU.

[79] Excepto para la función de rangeCheck que consistió en nueve líneas de código, y ocho archivos de seguridad descompilados. Respecto de rangeCheck la Corte de Distrito había dicho que un ingeniero de Sun Microsystems que había construido rangeCheck posteriormente trabajó en Google y contribuyó con dos archivos que contenían la función de rangeCheck –"Timsort.java" y "Comparable Timsort" en la Plataforma Android. En definitiva, las nueve líneas de la función de rangeCheck se copiaron directamente en Android. Con relación a los ocho archivos descompilados, la Corte de Distrito encontró que habían sido copiados y utilizados como archivos de testeo.

El Tribunal de Apelaciones del Circuito Federal afirma que ello no es correcto.

Por su parte el Tribunal de Apelaciones del Circuito Federal expresa que el Juez del Distrito se ha equivocado y no ha distinguido correctamente la cuestión principal acerca de cuál es la diferencia entre:

(i) aquello que es susceptible de ser protegido bajo la ley de derecho de autor bajo el sistema de EE.UU. (en inglés "*copyrightable"*), y
(ii) la conducta que constituye infracción de derechos de autor (en inglés "*infringing activity"*)

PROTECCIÓN BAJO EL SISTEMA DE DERECHOS DE AUTOR DE EE.UU.

La ley de derechos de autor de EE.UU. proporciona protección "a las obras originales de autores que hayan sido fijadas en un medio tangible de expresión", lo cual incluye a las obras literarias (USC por *"United States Code"* o Código de los Estados Unidos de América, Titulo 17, Capitulo 1, sección 102, inciso a[80]).

[80] Véase en https://www.copyright.gov/title17/ En su texto original el articulo 102 expresa: "*102. Subject matter of copyright: In general (a) Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories: (1) literary works; (2) musical works, including any accompanying words; (3) dramatic works, including any accompanying music; (4) pantomimes and choreographic works; (5) pictorial, graphic, and sculptural works; (6) motion pictures and other*

Por otra parte, los programas de computación se encuentran definidos como *"el set de declaraciones o instrucciones para ser usados directa o indirectamente por una computadora con el objeto de producir un resultado cierto"*[81].

Así, los programas de computación pueden estar sujetos a la protección bajo las leyes de derechos de autor de EE.UU. como *"obras literarias" (Atari Games Corp v. Nintendo of Am., Inc. 975 F.2d 832, 838 (Fed. Cir. 1992) ("como obras literarias, la protección bajo las leyes de derechos de autor de EE.UU. se extiende a los programas de computación)*.

Por su parte, para que una obra pueda ser protegida bajo el sistema de derechos de autor debe ser o estar calificada como *"original"* (USC Titulo 17, sección 102, inciso a).

El requisito de la originalidad no es particularmente exigente.

---

audiovisual works; (7) sound recordings; and (8) architectural works. (b) In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work. Por su parte el artículo 101 en sus definiciones, expresa cuando se considera que una obra se encuentra fijada en un medio tangible, y dice así: "A work is "fixed" in a tangible medium of expression when its embodiment in a copy or phonorecord, by or under the authority of the author, is sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration. A work consisting of sounds, images, or both, that are being transmitted, is "fixed" for purposes of this title if a fixation of the work is being made simultaneously with its transmission."

[81] Véase el artículo 101 que textualmente expresa "A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result" https://www.copyright.gov/title17/92chap1.html

El vocablo o término "originalidad" tiene un significado propio para la ley de derechos de autor que implica que la obra en cuestión haya sido creada en forma independiente, en el sentido que no sea copia de otra, y claro que esa originalidad posea un cierto grado mínimo de creatividad.

Por otro lado, la protección brindada por el derecho de autor de EE.UU. sólo se extiende a la *"expresión de las ideas"* y no a las ideas en sí mismas *(Mazer v. Stein, 347 U.S. 201 217 (194).*

Esta distinción se la conoce como la dicotomía o división de la idea y su expresión que se codifica en el artículo 102 inciso (b), que dice textualmente:

"en ningún caso el derecho de autor para la autoría de una obra original se extiende a las ideas, procedimientos, procesos, sistemas, métodos de operación, conceptos, principios o descubrimientos, sin importar la forma en las cuales estos sean descriptos, explicados, ilustrados, o incorporados en la obra en cuestión." (Golan v. Holder, 132 S. Ct. 873 890 año 2012) y (Baker v. Seldon, 101 U.S. 99, 101 año 1879).

También, la protección de derechos de autor se extiende a los elementos literales y a los no-literales de un programa de computación.

Los elementos literales de un programa de computación son el código fuente y el código objeto[82].

Los elementos no-literales de un programa de computación, entre otros, es la estructura, secuencia y organización del programa.

[82] Johnson Controls Inc, v. Phoenix Control sys., Inc., 886 F.2 1173, 1175 (9th Cir. 1989)

Si los elementos no-literales de un programa de computación están o no protegidos depende de las circunstancias particulares de cada caso.

El reclamo de Oracle en materia de derechos de autor bajo la ley de EE.UU. se refiere a dos aspectos sobre los que entiende que debe existir protección:

(i) los elementos literales de los paquetes de las API de Java (las 7000 líneas de "código fuente declarado") y,

(ii) los elementos no literales de cada uno de los 37 paquetes de las API de Java, es decir su estructura, secuencia y organización (su sigla en inglés "SSO")

El Tribunal de Apelaciones del Circuito Federal marca un importante asunto que es la diferencia entre:

(i) *aspectos literales y no-literales* de un programa de computación, y la

(ii) *copia literal y no-literal* (*Altai, 982 f.2d en 701-02*). La copia literal es la "copia exacta" de la expresión original. La copia no-literal es una suerte de "*parafraseo*" en lugar de copiar palabra por palabra. En el caso de autos, Google aceptó que había copiado el código fuente declarado *"verbatim"* en forma exacta, textual, literal, palabra por palabra.

Oracle ha mencionado que el código fuente declarado se incorpora a la estructura de cada paquete de las API de Java.

Por ello cuando Google copió el código declarado en esos paquetes de Java también copio la secuencia y organización de cada uno de los paquetes utilizada por Oracle.

También Oracle menciona que los elementos no-literales de los paquetes de las API de Java, es decir su estructura secuencia y organización que llevaron al código implementado por Google, está sujeta a la protección otorgada por la ley de los derechos de autor de EE.UU.

Por ello, Oracle no argumenta que exista *copia litera*l de la totalidad de la estructura, secuencia y organización, sino que afirma que Google copio literalmente el código declarado y luego parafraseó el resto de la estructura secuencia y organización cuando escribió su propio código fuente implementado.

Por ello, Oracle afirma que existe *copia no literal* de parte de Google respecto de la totalidad de la estructura, secuencia y organización.

Por otra parte, teniendo en cuenta las declaraciones testimoniales en el juicio quedo probado que el código fuente declarado y su estructura, secuencia y organización de los paquetes de las API de Java eran originales.

Así, surge de estas constancias que el diseño de los paquetes de las API de Java era un proceso creativo [83], y que los

[83] Véase "As noted above, the Java Packages provide a set of pre-existing programs that allow developers to quickly and efficiently create their own applications in Java. *But merely offering a programming platform in which developers could build cross-system applications would not have been enough to get developers to use it*. Indeed, that functionality could have been made accessible in any number of ways (as discussed infra). *To attract developers to invest the time to learn the platform, the selection and then naming and organization of the packages had to be easy to understand, memorize, and master. A package's declaring code therefore must communicate what a program does, how it relates to other programs, and what is needed for it to work, in a way that is intuitive and resonates with programmers. For this reason, Sun invested an enormous amount of creativity in developing an original hierarchy of packages organized into a*

desarrolladores de Sun Microsystems /Oracle tuvieron varias opciones para estructurar y organizar las API.

También, esto fue reconocido por el propio Juez de Distrito, y por Google, que no disputó lo dicho por el Juez de Distrito en cuanto a que las API de Java cumplían con el requisito de originalidad.

Dicho lo anterior, puede verse entonces que las partes estuvieron de acuerdo en que los paquetes de las API de Java cumplían con el requisito de originalidad requerida de acuerdo con el artículo 102, inciso (a), pero no estaban de acuerdo en la interpretación y aplicación del artículo 102, inciso (b).

Así, Google sostuvo al respecto que hay dos pasos en el análisis de la protección bajo derechos de autor, en donde el artículo 102, inciso (a) concede protección a las obras originales, mientras que el inciso (b) del mismo artículo 102, quita la protección a la obra en cuestión si ésta posee elementos funcionales.

---

distinctive library and unique classes. This code and organization is integral to the program, the way a table of contents is intertwined with a book. *Designing a package entailed numerous creative choices*. Java's architects had to decide what packages to create, and within those, how many classes and methods to include. Moreover, for every package, class, and method that made the cut, Java's creators had to name each element and determine how to arrange them. Over time, Java's developers evolved existing packages and added new ones. And the art of package design stretches beyond the level of individual packages. Because Java programmers benefit from familiarity with the Java Package library, and because many packages are interrelated, significant attention was paid to the totality of the library: the selection and arrangement of those packages must be just as appealing and elegant as each individual package. Making these thousands of choices was a massive creative task. *Oracle I, C.A. 20,797–98. None was required for the programs to perform their function or dictated by the Java language. Pet. App. 165a–166a*. https://www.supremecourt.gov/DocketPDF/18/18-956/133505/20200219153012820_Brief.pdf

A los ojos del Tribunal de Apelaciones del Circuito Federal esta interpretación de Google es incorrecta, ya que el Congreso de los EE.UU. ha enfatizado que el artículo 102 inciso (b) "...*de ninguna manera extiende o contrae el ámbito de protección conferido por el derecho de autor...*" "*...y que su propósito es reafirmar...*" y que "*el principio básico entre la división de idea y expresión se mantiene inalterado...*".

Por lo tanto, el artículo 102, inciso (b) no extingue la protección acordada a la expresión de una idea por la sola circunstancia que dicha expresión se haya incorporado en un método de operación.

Menciona el Tribunal de Apelaciones del Circuito Federal que las Cortes de Distritos no están en un todo de acuerdo en relación a que "examen o test" se debería utilizar para marcar una línea entre lo que constituye una expresión protegible por el derecho de autor de EE.UU. y lo que no.

Por ejemplo, en el caso *Whelan Assocs. Inc v. Jaslow Deantal Lab* se dijo que todo aquello que no fuera necesario para el propósito o funcionamiento de una obra constituye expresión protegible, y en el caso *Lotus* se sostuvo que un método para operar es el medio mediante el cual un usuario puede operar algo.

Por el contrario, conforme el Circuito Noveno (que como se mencionó anteriormente por parte del Juez de la Corte de Distrito constituye ley aplicable al caso en cuestión) a los fines de determinarse si los *elementos no-literales de un programa de computación* constituyen una expresión protegible bajo la ley de derechos de autor se utiliza el examen o test desarrollado por el Tribunal de Apelaciones del Circuito Federal del Circuito Segundo, conocido como test de "abstracción, filtración y comparación".

Este examen descarta la idea de que por “el sólo hecho de que cualquier cosa ejecute una función no pueda ser protegido”, es decir, aunque un elemento de una obra pueda ser caracterizado como un “método de operación”, ese elemento, puede, sin embargo, contener un grado de expresión que lo haga elegible para ser protegido por el derecho de autor.

De alguna forma este examen descarta la posición de los casos *Lotus* (todo aquello que ejecuta una función ya no es protegible) y *Whelan* (una vez identificadas las ideas de un programa de computación, todo el resto es expresión protegible, en este caso sobre la base que más de una idea pueda estar incorporado en un programa de computación)

Ahora bien, este examen en su totalidad, es decir en sus tres etapas (abstracción, filtración y comparación), sólo aplicaría ante los casos donde el titular de derechos alega infracción de *elementos no-literales*.

En el caso en cuestión, donde se ha admitido por parte de Google la *copia literal* del código fuente declarado *no es requerido*, ya que no hace falta realizar un examen completo (de las tres etapas abstracción, filtración y comparación), sino que alcanza con realizar la etapa de “filtración”, que es la etapa en donde se focaliza en la protección de la obra conforme a los principios del derecho de autor (en inglés “*copyrightability”*). Es decir, si es protegible o no.

De las constancias de autos, si bien es cierto que se mencionó por el Juez de la Corte de Distrito el examen o test en cuestión, pareciera según se afirma que no se lo aplicó.

En efecto, en su lugar cuando se analizó el código fuente declarado, el Juez de Distrito concluyó e interpretó directamente el artículo 102 inciso (b) como “una prohibición de protección

de cualquier elemento funcional que *"sea esencial para la interoperabilidad" sin importar su forma.*

El Tribunal de Apelaciones del Circuito Federal sostiene que invocar la "interoperabilidad" en el análisis de protección por derechos de autor es incorrecto, y que el Juez del Distrito yerra en este aspecto.

Sostiene que la interoperabilidad es un elemento que debe tenerse presente en el análisis de uso justo, pero no en el análisis de si una obra merece o no protección conforme a la ley de derechos de autor de EE.UU. (en inglés "*copyrightability")*.

DOCTRINA DE LA FUSIÓN ("MERGE DOCTRINE")*[84]*

La doctrina de la fusión funciona como una excepción a la dicotomía o división entre la idea y su expresión.

Esta doctrina reza que cuando existe un número limitado de maneras de expresar una idea, se dice que la idea "se ha fusionado" con su expresión, y entonces la expresión se torna no protegible por el derecho de autor del sistema de EE.UU. (en inglés "*Copyright Law"*).

En definitiva, una supuesta obra protegida bajo derechos de autor no podría tener protección judicial por infracción de

[84] La teoría de la fusión sólo fue argumentada por la Corte de Distrito en relación al código declarado, y no respecto de su estructura secuencia y organización (SSO). De hecho, El Juez de Distrito reconoció que había cientos de formas de haber organizado los paquetes de la API de Java. Por ello reconoció que la SSO era original y que no había nada en las reglas del lenguaje de programación de Java que requiriese que Google hubiese replicado como lo hizo los mismos paquetes, clases y métodos.

derechos, si la idea contenida en dicha obra sólo podría expresarse de una sola manera.

Menciona el Tribunal de Apelaciones del Circuito Federal que el Circuito Noveno trata este concepto de la fusión de "idea/expresión" como una "defensa afirmativa cuando se alega una infracción", entonces el análisis no es relevante a la hora de analizar "la protección bajo la ley de derechos de autor de los paquetes de Java" (en inglés "*copyrightability"*).

Recuérdese que según la Corte de Distrito bajo las reglas de Java un programador debe usar idénticas declaraciones o líneas de encabezados (en inglés "*headers"*) para declarar un método especificando la misma función.

Por lo tanto, expresaba el Juez de Distrito que como hay una sola manera de escribir el código declarado para cada uno de los paquetes de Java, la idea (método declarado) se fusiona con la expresión (código implementado), y concluye que nadie podría reclamar derechos sobre ello, ya que no es protegible bajo el sistema de derechos de autor.

A este respecto dice el Tribunal de Apelaciones del Circuito Federal que nuevamente la Corte de Distrito se equivoca, ya que aplica erróneamente la doctrina de la fusión.

En efecto dicha teoría aplicaría sólo sí al momento, o en el tiempo, en que hubiese sido creado por Oracle el código declarado hubiese podido haberse hecho de una sola manera.

Es decir, que en aquel tiempo "Oracle sólo hubiese tenido una sola opción" (porque sólo existe una sola opción).

Ahora, está probado en el juicio que Oracle tuvo infinidad de opciones para organizar y seleccionar las 7000 líneas de código que Google copió.

Así el método de Java utilizado como ejemplo (***java.lang.Math.max***) podría haber sido llamado y organizado como **"Math.maximun"** o **"Arith.larger"**.

En definitiva, la doctrina de la fusión (acerca de las opciones de como expresar una idea) se tiene en cuenta al momento de la creación de la obra, y no al momento de la infracción.

Vale decir, no interesa que opciones de expresar esa idea tenía el copista (al momento de copiar), sino su creador (al momento de crear).

Por lo tanto, como existieron múltiples alternativas de expresión al momento de creación del código declarado para Oracle, el Tribunal de Apelaciones del Circuito Federal rechazó la aplicación de la doctrina de la fusión.

FRASES CORTAS Y NOMBRES

Se sostuvo que es cierto que las palabras, las frases cortas como nombres, títulos y eslogan no están sujetos a derechos de autor bajo el sistema de EE.UU. -si bien como obras literarias usualmente lo son-.

Ahora bien, a los efectos de responder la pregunta sobre si existe protección bajo derechos de autor, la cuestión no era analizar si la obra en cuestión que contenían frases, eran cortas o no, sino más bien analizar si ellas "eran creativas" o no.

Al diseccionarse en forma individual las líneas de código declarado en frases cortas, la Corte de Distrito erró en reconocer una combinación original de elementos que puede ser protegido.

Oracle ejerció creatividad en la selección y disposición de sus métodos declarados cuando creó los paquetes de la API de Java, escribiendo su propio código declarado, lo cual posee expresiones protegibles que son susceptibles de ser protegidas por el derecho de autor.

Por estas razones se concluye que no es correcta la visión de la Corte de Distrito y se rechaza sus argumentos.

DOCTRINA DE "SCENES A FAIRE"

La doctrina de la *"scenes a faire"* limita a ciertas obras creativas de protección bajo la ley de derechos de autor.

Bajo esta doctrina se establece que las obras no pueden ser protegidas en contra de infracciones si son estándares o comunes dentro de un género o tópico, o si son necesarias de ser seguidas conforme a determinado tema.

En el juicio se rechazó la postura de Google en relación a la doctrina del *scenes a faire*.

En primer lugar, se encontró que Google no había presentado suficiente evidencia que mostrase que agrupar a los métodos dentro de las clases, o las clases dentro de los paquetes constituyera una práctica común y esperada.

En la instancia anterior se sostuvo que es imposible decir "que todas las clases y sus contenidos es típico de esas clases".

A su vez, al igual que la doctrina de la fusión, en la doctrina del *scenes a faire* el foco de las circunstancias debe ser evaluada al momento de la creación, y no de la copia o infracción.

Es decir que las circunstancias de si agrupar clases y todos sus contenidos dentro de una clase es o no típico o esperado, se debe evaluar al momento en que Oracle creó y dispuso esa agrupación, y no al momento en que Google lo copió.

Por estas razones el Tribunal de Apelaciones del Circuito Federal rechazó la apelación de Google en relación a este punto, y sostuvo que la doctrina del "*scenes a faire"* no afecta la protección del código fuente declarado como tampoco de la estructura, secuencia y organización de los paquetes de las API de Java.

LA ESTRUCTURA, SECUENCIA Y ORGANIZACIÓN DE LOS PAQUETES DE LAS API DE JAVA

Para determinar que la estructura, secuencia y organización de las API de Java no eran protegibles bajo la ley de derechos de EE.UU., la Corte de Distrito sostuvo que más allá de reconocerle creatividad a la estructura, secuencia y organización de los 37 paquetes que forman parte de la librería de software estándar de Java, la misma constituye un *método o sistema de operación*, y por lo tanto no es protegible bajo el derecho de autor.

En arribar a esta conclusión, la Corte de Distrito basó su decisión en el caso *Lotus Development Corp v. Borland International*[85]resuelto por el Circuito Primero.

En el caso *Lotus*, el demandado había copiado la jerarquía de comandos del menú (*print, quit,* etc.) y sus interfaces del

[85] En este caso la Corte Suprema de Estados Unidos había concedido el *certiorari*, pero después de la etapa de los argumentos orales, SCOTUS anunció que estaba dividida en partes iguales. Por lo tanto, SCOTUS dejo vigente la decisión del Circuito Primero.

programa de computación Lotus 1-2-3, que era un programa de hojas de cálculo que permitía a los usuarios ejecutar funciones contables. Estos comandos estaban organizados en más de 50 menúes y sub-menúes.

Borland no copió código de *Lotus*, pero si copió la jerarquía de comandos del menú de *Lotus* para incluirlo en su programa.

La pregunta que se había generado en ese caso era si esa jerarquía de comandos del menú de un programa de computación era protegible bajo el sistema de derechos de autor.

A pesar de que la Corte de Distrito que llevo el caso había encontrado que en la disposición y selección de los comandos había cierta expresión creativa de parte de *Lotus Development*, el Tribunal de Apelaciones del Circuito Federal del Circuito Primero resolvió que la "jerarquía de comandos" no era protegible por el sistema de derechos de autor, ya que constituía un método para operar conforme a lo dispuesto en el artículo 102, inciso (b).

Así "método para operar" se lo definió como "los medios por los cuales una persona opera algo, lo cual podría ser un auto, un procesador de alimentos, o una computadora"

El Tribunal de Apelaciones del Circuito Federal expresa que la Corte de Distrito del Norte de California se equivoca al fundar su decisión en el caso *Lotus*, porque *Lotus* es distinto en cuanto a los hechos del caso en cuestión, y además es inconsistente con la ley aplicable para el Tribunal de Apelaciones del Circuito Federal del Circuito Noveno.

Para empezar, en el caso *Lotus* el demandado no copió código de *Lotus*, y en el caso en cuestión Google reconoció que había copiado el código fuente declarado de Oracle en forma idéntica.

En segundo lugar, en el caso *Lotus* los comandos (*copy*, *print*, *etc*.) que se discutieron no eran creativos, ahora no hay dudas de que el código fuente declarado y la estructura, secuencia y organización de las API de Java son creativas y originales.

Por último, mientras que en *Lotus* se estableció que los comandos eran "esenciales para operar" el sistema del programa Lotus 1-2-3, es indiscutible que Google no necesitaba copiar la estructura, secuencia y organización de los paquetes de las API de Java para escribir programas en lenguaje Java.

A su vez, el Circuito Noveno, no adoptó el razonamiento del caso *Lotus* en cuanto a "método de operación", porque se afirma que es inconsistente con lo resuelto por el Tribunal de Apelaciones del Circuito Federal del Circuito Noveno donde se ha reconocido que la estructura, secuencia y organización de un programa de computación es elegible para la protección brindada por el derecho de *autor en la medida que constituya la expresión de una idea, y no la idea en sí misma. (Johnson Controls, 886 F.2d en 1175-76).*

Y se agrega, que mientras en el caso *Lotus* se sostuvo que "la expresión que es parte en un método de operación no es protegible por el derecho de autor", el Tribunal de Apelaciones del Circuito Federal, tomando la doctrina aplicable del Circuito Noveno- concluye totalmente lo opuesto, ya que *"el derecho de autor protege la expresión de un método de operación"*.

Por lo tanto, se rechaza la teoría de *Lotus* dada por la Corte de Distrito del Norte de California, ya que no es aplicable, y además contradice el caso *Johnson Controls* que es el que rige para el Circuito Noveno.

Agrega el Tribunal de Apelaciones del Circuito Federal, que según la Corte de Distrito la estructura, secuencia y organización es creativa, pero sin importar ello constituye una estructura de

comando para operar, es un “método de operación”, y por ende no es protegible bajo el sistema de derecho de autor.

Es decir, es creativa, expresa una idea, pero no es protegible porque es funcional.

El problema con esta visión, es que los programas de computación son por definición “funcionales”, es decir están designados para llevar a cabo una función.

Y esto es conforme a la “definición de programa” de computación, conforme artículo 101, Titulo 17 USC, donde se reconoce que la función es “producir un cierto resultado”.

Si hubiese que aceptar la posición de la Corte de Distrito que sugiere que un programa de computación no es protegible simplemente porque *“produciría ciertas funciones pre-asignadas”* ningún programa de computación seria protegible bajo el sistema de derechos de autor.

Esto sería contrario a lo querido por el Congreso al proveer protección a los programas de computación, como también sería contradictorio con la doctrina obligatoria emanada del Circuito Noveno.

Entonces, un conjunto de comandos que instruyen a una computadora para producir un resultado esperado puede contener expresión que sea elegible para la protección bajo el sistema de derechos de autor.

Es decir, que una obra original -incluso aquella que su propósito sea una función-se encuentra sujeta a protección bajo el derecho de autor siempre que el autor posee múltiples formas de expresar la idea.

El artículo 102, inciso (b) no niega protección "en forma automática" a los elementos de un programa de computación que son funcionales.

Oracle no reclamó protección en abstracto al respecto de la idea de la estructura organizacional de las funciones de su programa de computación o de sus paquetes/clases/métodos.

Por el contrario, Oracle reclamo por la protección de su "forma particular" de organizar y nombrar a los 37 paquetes de las API de Java.

En la etapa de los argumentos orales Oracle expresó que nunca sostendría que alguien que use el formato de clasificación en "paquete/clases/métodos" violaría los derechos de Oracle.

En absoluto.

Oracle no posee la propiedad de todas las formas posibles de concebir una manera de "organizar".

Oracle sólo sostuvo que posee la propiedad, y por ello reclamó la protección, de la "forma específica de su expresión", de nombrar cada uno de los 362 métodos, poniéndolos en 36 clases distintas, y en 20 subclases.

Teniendo en cuenta todo lo expuesto, el Tribunal de Apelaciones del Circuito Federal entendió que la estructura, secuencia y organización era original y creativa, y que el código declarado podía haber sido escrito y organizado en varias formas y aun así lograr obtenerse las mismas funciones[86], por ello

[86] Amici McNealy y Stuphin (Ex Ejecutivos de Sun Microsystems) mencionan "*...en otros ambientes de programación se puede ver que los creadores de otras plataformas de desarrollo proveen las mismas funciones con distintas expresiones creativas. Por ejemplo, en Java un desarrollador para establecer la zona horaria (setting the time zone) llamaría al método de Java setTime-Zone dentro de la clase DateFormat del paquete de java.text.*

concluyó que el artículo 102, inciso (b) no prohíbe a los paquetes de Java de la protección otorgada por el sistema de derechos de autor sólo porque estos a su vez ejecuten "funciones".

LA INTEROPERABILIDAD ALEGADA POR GOOGLE NO ES RELEVANTE PARA EL ANÁLISIS DE LA PROTECCIÓN BAJO EL SISTEMA DE LOS DERECHOS DE AUTOR DE EE.UU.

La interoperabilidad alegada por Google no es relevante para el análisis de si las API de Java son protegibles o no a la luz del derecho de autor de EE.UU.

La Corte de Distrito calificando a la estructura, secuencia y organización de los paquetes de las API de Java como un "método de operación" o "método para operar" o "sistema para operar", también dijo que la duplicación o la copia del "comando de estructura" era necesario para garantizar la *"interoperabilidad",* diciéndose de que en orden a que al menos parte del código pudiera correr en Android, Google tenía que proveer el mismo *sistema de comandos* de ***java.package.Class.method()*** utilizando los mismos nombres con la misma estructura o taxonomía y con las mismas especificaciones de funcionalidad.

Por ello, el Juez de Distrito concluye que Google replicó aquello que era necesario para lograr un grado de compatibilidad, no más que eso, pero realizando su propia implementación.

Microsoft provee una funcionalidad similar con una estructura y selección totalmente distinta para el esquema de nombres. En la plataforma de desarrollo para teléfonos Windows, Microsoft almacena los programas de zonas horarias dentro de la clase TimeZoneInfo en su Sistema de espacios de nombres o namespace (que sería la versión de "paquete" para Microsoft)".

Ahora bien, para llegar a esta conclusión el Juez de Distrito se basó en dos casos resueltos por el Tribunal de Apelaciones del Circuito Federal del Circuito Noveno: *Sega Enterprises v. Accolade, Inc, 977 F.2d 1510 (9th Cir.1992) y Sony Computer Entertainment, Inc. V. Connectix, Corp., 203 F.3d 596 (9th Cir.2000).*

Así interpretó que estas dos decisiones sostuvieron que los procesos de interfaces que son necesarios duplicar con el objeto de lograr compatibilidad son aspectos funcionales que no están protegidos bajo el sistema de derechos de autor de EE.UU. ello conforme a lo dispuesto en el artículo 102, inciso (b), USC Titulo 17.

Veamos que en el caso *Sega Enterprises v. Accolade, Inc*[87], *Sega* era un fabricante de consolas de videos juegos, como también de cartuchos de videos juegos que tenían estos últimos una suerte de programas ocultos cuyos elementos eran necesarios para lograr compatibilidad con su consola.

Accolade efectúo ingeniería inversa de los cartuchos de videos juegos de *Sega* para conocer los requerimientos de compatibilidad, y así poder crear sus propios cartuchos de videos juegos para ser usados en la consola de *Sega*.

Teniendo en cuenta que en el proceso de decompilación, *Accodale* efectúo copias intermedias del código objeto de la consola de *Sega*, y si bien ello fue reconocido como que podría causar infracción, se concluyó que desamblar o decompilar código objeto protegido por las ley de derechos de autor de EE.UU. constituye un uso justo (en inglés "*fair use")* en la medida de que dicha decompilación sea la única manera o medio para acceder a aquellos elementos de código que no son

[87] Sega Enterprises v. Accolade, Inc, 977 F.2d 1510 (9th Cir.1992)

protegibles por el derecho de autor y el copista tenga una razón legítima para obtener dicho acceso.

Como se entendió que la empresa *Accolade* tenía un interés legítimo en fabricar sus propios cartuchos para hacerlos compatibles con la consola de *Sega*, se reconoció que la copia de código efectuada constituía uso justo.

En el caso *Sony Computer Entertainment, Inc. V. Connectix, Corp.*[88], el Circuito Noveno se expresó en forma similar, y dijo que la ingeniería inversa y la copia intermedia realizada por el demandado en relación al programa de software de *Sony* -protegido por la ley de derechos de autor de EE.UU.- con el propósito de obtener acceso a elementos no protegidos del software de *Sony* constituye uso justo.

Se dijo que el proceso de decompilación era la única forma en que se podía lograr acceso a los elementos no protegidos

El Tribunal de Apelaciones del Circuito Federal menciona en primer lugar, que ambos casos se relacionan con uso justo, y no con la protección bajo derechos de autor de un determinado programa de computación (en inglés "*copyrightability"*).

En segundo lugar, expresa que utilizar como base los casos de *Sony* y *Sega* para revisar el asunto de la protección (en inglés "*copyrighability"*) en el presente juicio está fuera de contexto y constituye un error.

Si bien reconoce el Tribunal de Apelaciones del Circuito Federal que en ambos casos se reconoció que los programas de software contenían elementos funcionales que no estaban protegidos por la ley de derechos de autor de EE.UU., no podría

[88] Sony Computer Entertainment, Inc. V. Connectix, Corp., 203 F.3d 596 (9th Cir.2000)

afirmarse que por ese sólo hecho, es decir, *porque que existan elementos funcionales no protegibles en un programa, convierta a todo el programa de computación como no protegible por la ley de derechos de autor de EE.UU.*

Por ello el Tribunal de Apelaciones del Circuito Federal expresa su desacuerdo a la sugerencia efectuada por Google de que los casos *Sony* y *Sega* hayan creado una *"excepción de interoperabilidad"* al régimen de derechos de autor del sistema de EE.UU.

Y agrega que para determinar si ciertos aspectos de un supuesto software en infracción no están protegidos bajo la ley de derechos de autor de EE.UU., el foco debe estar puesto en los *"factores externos"* que influenciaron en las alternativas del creador del producto en supuesta infracción[89].

Por ejemplo, esto podría ser las especificaciones de una computadora sobre la cual un programa de software en particular deseara ejecutarse o los requerimientos de compatibilidad de otros programas con los cuales el programa está diseñado para operar en su conjunto.

Teniendo en cuenta que la protección bajo la ley de derechos de autor se focaliza en las opciones disponibles que tenía su autor al momento de crear el programa de computación, la pregunta relevante en materia de compatibilidad sería si las opciones del creador fueron dictadas o no por la necesidad de asegurar que su programa funcione con los programas existentes de terceros.

En definitiva, si el demandado busca posteriormente hacer interoperable su programa con el programa de la parte actora o

[89] (Dun & Bradstreet Software Servs v. Grace Consulting, Inc., 307 F.3d 197, 215 (3d Cir. 2002).

demandante, ello no tiene relación con respecto a si el software creado por el demandado tenía alguna limitación de diseño dictada por factores externos[90].

Por ello es la interoperabilidad de Oracle la que importa, y no la de Google, y es la que aplicable en el contexto de protección de derechos.

Y conforme a las evidencias del pleito no surge que cuando Oracle creó los paquetes de las API de Java lo hiciera para obtener requerimientos de compatibilidad de otros programas preexistentes.

De nuevo, el requisito de compatibilidad debe ser analizado al momento del creador no de quién copia (en el caso Google).

Por otro lado, Google sostuvo que los nombres y las declaraciones de las clases y los métodos de Java era la única manera o medio para lograr el grado de interoperabilidad con los programas existentes en el lenguaje Java.

Ahora, conforme las constancias del litigio Google diseño Android para que no sea compatible con la plataforma Java, o específicamente con la máquina virtual de Java (*Java virtual machine*), por lo tanto, el argumento de la interoperabilidad alegada por Google es al menos confuso.

Entonces, se ha dicho que Google copió los paquetes de Java con la finalidad de que una aplicación escrita en Java pudiera ejecutarse en la plataforma Android, pero no hay evidencia en el juicio que tal aplicación exista, como tampoco se menciona ninguna aplicación de Java, sea antes o después de Android, que pudiera ejecutarse en la plataforma Android.

[90] (Dun & Bradstreet Software Servs v. Grace Consulting, Inc., 307 F.3d 197, 215 (3d Cir. 2002).

En realidad, la compatibilidad buscada por Google no era con la plataforma Java o con la máquina virtual de Java (JVM), sino que Google quería capitalizar el hecho de que los desarrolladores ya estaban entrenados y con experiencia en utilizar los paquetes de las API de Java.

En definitiva, el interés de Google estaba en acelerar el proceso de desarrollo aprovechando a Java con su base existente de desarrolladores.

Si bien este objetivo de competitividad podría ser relevante a la hora de analizar el uso justo, concluye el Tribunal de Apelaciones del Circuito Federal que no es relevante para analizar si el código fuente declarado y la estructura, secuencia y organización de los paquetes de Java son protegibles o no por el sistema de derechos de autor de EE.UU.

Por otra parte, en relación al argumento sostenido por Google donde sugirió que Google tenía derecho a duplicar o copiar los API de los paquetes de Java porque ello se había vuelto el estándar efectivo en la industria, dicho argumento es rechazado.

Google no cita al respecto ningún caso que sugiera que la protección bajo los derechos de autor se pierda cuando se convierte en “popular”.

En realidad, Google tuvo la libertad de desarrollar sus propios paquetes de API y en su caso cabildear por su adopción, pero en su lugar, Google escogió copiar el código declarado por Oracle y la estructura, secuencia y organización, y capitalizar a la comunidad de programadores preexistentes quienes ya estaban acostumbrados y entrenados a usar los paquetes de Java.

Por esta razón es que se entendió que el argumento de “estándar de la industria” no tenía relación con el análisis de si

el código declarado y la estructura, secuencia y organización era protegible o no bajo la ley de derechos de autor de EE.UU.

USO JUSTO (O USO LEGÍTIMO)

Con relación al "uso justo" el Tribunal de Apelaciones del Circuito Federal en su sentencia del 9 de Mayo de 2014 entendió que conforme a las constancias del pleito no había prueba suficiente en relación a los cuatro factores del uso justo o legítimo ("*fair use"*) y por lo tanto, no podía tomar una decisión al respecto, razón por la cual, ordenó que se llevara a cabo un *segundo juicio* para que determinara la pregunta de si el uso realizado por Google en relación al código declarado y a la estructura, secuencia y organización podría ser considerado como "uso justo" (en inglés "*fair use"*).

En efecto, el Tribunal de Apelaciones del Circuito Federal entendió que existían entre las partes disputas en los hechos en cuanto a cómo Android había sido usado, y a su vez, si las API que Google había copiado tenían o servían para cumplir con la misma función en Android que en Java.

Por otra parte, recuérdese que el Jurado se abstuvo de tratar el asunto de "uso justo", y también de que, el Juez de Distrito había rechazado ordenar un juicio por este asunto, ya que había entendido que el código declarado y la estructura, secuencia y organización, copiado por Google no era protegible por el derecho de autor de EE.UU.

CAPITULO VIII

RECURSO DE APELACIÓN (PETICIÓN I DE *WRIT OF CERTIORARI*) ANTE LA CORTE SUPREMA DE JUSTICIA DE EE.UU. INTERPUESTO POR GOOGLE CONTRA LA SENTENCIA DEL TRIBUNAL DE APELACIONES DEL CIRCUITO FEDERAL

Contra esta sentencia del Tribunal de Apelaciones del Circuito Federal que determinó que el código fuente declarado y la estructura, secuencia y organización de los paquetes de las API de JAVA era protegible bajo el sistema de derecho de autor de EE.UU., Google con fecha 6 de octubre del 2014 presentó una Petición I de *Writ of Certiorari*[91] ante la Corte Suprema de Justicia de EE.UU. (SCOTUS).

ARGUMENTOS DE GOOGLE CONTRA LA SENTENCIA I DEL TRIBUNAL DE APELACIONES DEL CIRCUITO FEDERAL

Google sostuvo que la decisión del Tribunal de Apelaciones del Circuito Federal del 9 de mayo de 2014 era contraria a derecho, ello conforme a lo dispuesto en el artículo 102, inciso b), del Título 17 USC.

Afirma Google que el "código fuente declarado" o "encabezado del método" (en inglés "*method header"*) constituye un *"método de operación o un método para operar"*

[91] Véase en https://www.eff.org/files/2014/11/10/google_cert_petition_10-06-14_oracle.pdf

y, por ende, se encuentra excluido de toda protección bajo la ley de derechos de autor de EE.UU.

Al hacer Sun Microsystems disponible el lenguaje de programación Java sin restricciones de ningún tipo, Sun Microsystems buscaba atraer a la mayor cantidad de personas posibles para construir una de las redes más grande del mundo.

Mientras Sun Microsystems alentaba a los programadores a aprender y utilizar el lenguaje de programación Java, promocionaba sus programas pre-escritos (la librería de software estándar de Java).

Sun Microsystems tuvo éxito en llevar a una generación entera de programadores a la comunidad de Java.

Millones de programadores invirtieron tiempo y esfuerzo en aprender el lenguaje de programación Java, convirtiendo a Java en uno de los lenguajes de programación más populares del mundo.

Así entonces los programadores tenían acceso al conjunto de programas pre-escritos por Sun Microsystems a través de las API de Java.

Las aplicaciones de programación de interface proveían acceso a esos miles de métodos de Java, ejecutando cada uno de ellos una función determinada, como las de en caso de darse dos números y elegir el mayor de ellos.

Así los métodos se agrupaban en clases, y las clases en paquetes.

El "código fuente" de cada método consiste en:

(i) un "código fuente declarado" (en inglés "*declaring source code"*) o encabezado del método (en inglés

"*method header"*) o simplemente declaración (en inglés "*declaration"*), y

(ii) un código fuente implementado (en inglés "*implementing source code"*) o cuerpo del método (en inglés "*method body"*).

El *encabezado del método* o *declaración* introduce el cuerpo del método y especifica los nombres, parámetros y funcionalidad de cada uno de los métodos (en inglés *Java methods*).

El *cuerpo del método* (en inglés "*method body"*) es un bloque de código que implementa el método instruyendo a una computadora de como ejecutar una determinada función, por ello se lo conoce como "código fuente implementado".

Para poder hacer uso de los métodos, es decir, *de la función que proporciona*, los programadores no necesitan conocer el código fuente implementado, sólo deben conocer el *"comando abreviado"* (en inglés "*shorthand command"*) que causa que el código fuente implementado ejecute la función deseada, como la de elegir el mayor de dos números.

De esta manera, los programadores utilizan estos *comandos abreviados* para operar el método (programas pre-escritos que conforman la librería de software estándar de Java).

Estos *comandos abreviados* poseen un formato especifico:

"java.package.Class.method(input)"

Cada *"comando abreviado" deriva directamente* del *"código declarado"* o encabezado de método", que de igual forma que el

comando abreviado, especifica el nombre del método, clase, paquete, y el *input.*

Así por ejemplo en **"*java.lang.Math.max(1.2)*"** se refiere a un método particular ***"max"*** que devuelve como resultado el mayor de dos números (1,2), localizado dentro de la clase **"*Math*"**, y esta a su vez localizada dentro del paquete **"*java.lang*"**.

Android es una de las plataformas para dispositivos móviles más populares del mundo.

En el segundo trimestre del año 2014, los fabricantes como Samsung, HTC, LG, y Lenovo vendieron más de 255 millones de teléfonos móviles inteligentes que utilizan la plataforma Android[92].

La plataforma Android incluye unos 168 paquetes de métodos, y Google escribió o adquirió el código fuente implementado de esos paquetes (que conforman o integran la librería de software estándar de Android).

En el juicio en cuestión, no existe disputa entre las partes relacionadas a (i) el lenguaje de programación Java, y (ii) al código fuente implementado por Google.

Lo debatido y discutido entre las partes se centra sobre el uso que Google realiza sobre los mismos *"headers" o* encabezados o código fuente declarado de los métodos de cada uno de los 37 paquetes de la librería de software estándar de Android.

Métodos estos, que ejecutan funciones claves para los dispositivos de telefonía móvil.

[92] Véase en International Data Corporation (IDC), Worldwide Smartphone OS Market Share (2014) http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Entonces, los programadores independientes crean aplicaciones para utilizar en dispositivos Android.

Teniendo en cuenta que los programadores conocen y usualmente prefieren desarrollar en el lenguaje de programación Java, Google concluyó que tales programadores querrían encontrar en la plataforma Android las *mismas funcionalidades* provistas por los 37 paquetes de Java, e *invocables* mediante el mismo sistema de nombres usados en Java.

Por tanto, para que los *"comandos abreviados"* funcionaran en Android, Google tuvo que *replicar exactamente* los mismos "*headers"* o "encabezados de métodos" o "código fuente declarado" que el usado por Oracle en las API de Java, ya que cualquier cambio que Google hubiera introducido hubiera *impedido que esos comandos abreviados funcionaran correctamente*.

Es decir, como fuera sostenido por el Juez de la Corte de Distrito, en relación a estas líneas de código [fuente declarado] *"Android y Java deben ser idénticas".*

Como Google sólo replicó los encabezados de métodos o código fuente declarado, y, por otro lado, Google escribió su propio cuerpo del método o código implementado, sólo el 3% de las líneas de código son las mismas en los 37 paquetes en disputa.

Entonces dicho todo lo anterior, y en relación al código fuente declarado, en su oportunidad la Corte de Distrito del Norte de California mencionó en su sentencia de mayo de 2012 que los "encabezados de métodos o código fuente declarado o simplemente declaraciones", incluyendo sus nombres y su organización, conformaba un sistema o método de operación que se encuentra excluido de protección conforme a lo estipulado en el artículo 102 inciso b) del Título 17 del USC.

En efecto, dicho inciso b) del artículo 102 establece *"En ningún caso la protección del derecho de autor sobre una obra original se extiende a cualquier idea, procedimiento, proceso, método para operar, concepto, principio, o descubrimiento, sin importar la forma en la cual sea [tal idea, o método, etc.] descripta, explicada, ilustrada, o incluida en la obra original.*

Por tal razón, el Juez de la Corte de Distrito expresó que Google tenía derecho a escribir su propio código fuente implementado de las mismas funciones o métodos que se encuentran en las API de Java.

Así, se expresó que el derecho de autor de los EE.UU. no protege, y por ende no confiere derechos sobre una o todas las formas de implementar una función o especificación, sin importar que tan creativa pueda ser.

Teniendo en cuenta que el sistema del código fuente declarado o encabezados de métodos conforma una "estructura de comandos" para operar los programas pre-escritos, el Juez del Distrito concluyó que dicho sistema podría recibir tal vez protección a través del sistema de patentes de invención, pero no bajo el sistema de derechos de autor.

También el Juez de Distrito hace mención a la "compatibilidad".

Antes de la fecha de la publicación de Android, y hasta su disponibilidad en el mercado, ya existían millones de líneas de código en Java las cuales necesariamente usaban el *formato de comando* ***java.package.Class.method*()** para llamar o invocar alguna o todas de las funciones de los 37 paquetes de Java en cuestión, y que claro, también necesariamente usaban los mismos nombres de la estructura de comandos.

Entonces, para que al menos parte de ese código escrito por los programadores independientes teniendo en cuenta el aprendizaje realizado en el lenguaje Java, pudiera ejecutarse en Android, Google estaba obligado a utilizar el mismo *sistema de comandos* de ***java.package.Class.method()*** con los mismos nombres y su misma taxonomía con las mismas funciones especificadas.

Como resultado de ello, Google replicó sólo aquello que era necesario para lograr un grado de "compatibilidad", pero no más allá de ello, es decir, copió el código fuente declarado de las API de los 37 paquetes de Java y su estructura, secuencia y organización.

Ahora bien, por otro lado, el Tribunal de Apelaciones del Circuito Federal sostuvo en cambio que el artículo 102 inciso b) no excluye a los sistemas o métodos de operación de la protección de los derechos de autor, y que, por lo tanto, *todos los elementos originales de una obra están sujetos a la protección siempre y cuando,* y en la medida que *el autor de la obra original haya tenido múltiples formas de expresar esa idea*.

Con lo cual, la protección de los derechos de autor se extiende a todos los elementos de una obra original, en este caso un programa de computación, incluyendo a los métodos de operación, *en la medida de que el autor del programa de computación haya tenido al menos más de una forma de expresarlo*.

Así el Tribunal de Apelaciones del Circuito Federal sostiene que el inciso b) del artículo 102 no extingue de la protección acordada a una forma particular de expresión de una idea por el mero hecho de que dicha idea esté expresada o incorporada en un método de operación.

La visión del Tribunal de Apelaciones del Circuito Federal es que la sección 102 inciso b) sirve solamente para codificar el principio del derecho de autor de EE.UU. de la dicotomía o división entre "idea" y "expresión [de idea]", que expresa que la protección conferida por el derecho de autor se extiende sólo a la expresión de las ideas, pero no a las ideas en sí mismas.

Por lo tanto, afirma el Tribunal de Apelaciones del Circuito Federal que como Google podría haber escrito o construido sus propios paquetes si lo hubiese querido, y así haber escrito su correspondiente código fuente declarado o encabezados de métodos organizándolo y nombrándolo de diferentes formas, *y aún, así lograr tener la misma funcionalidad*, el artículo 102 inciso b) no impide la protección de derechos de autor a los paquetes de Java.

Es decir, que la interpretación del artículo 102, inciso a)[93] y b) debe hacerse de manera colectiva, conjunta y no de forma excluyente.

Google sostiene que este razonamiento del Tribunal de Apelaciones del Circuito Federal es absolutamente incorrecto, contrario al derecho de EE.UU., y a la historia legislativa de los incisos a) y b) del artículo 102 del Título 17 del USC.

Bajo el artículo 102 inciso a) la autoría de una obra original es *"generalmente"* protegible y, bajo el inciso b), que va sobre lo *"específico"*, se establece que en ningún caso la protección de la autoría sobre las obras originales se extiende a los sistemas,

[93] El inciso a) del artículo 102, es el que dice "...la protección del derecho de autor subsiste, de acuerdo con este título [17] en la autoría de obras originales fijadas en cualquier medio tangible de expresión, conocida o desarrollada posteriormente, desde donde puedan ser percibidas, reproducidas, o de cualquier otra forma comunicadas, sea directa o indirectamente o con la ayuda de una máquina o dispositivo..."

métodos para operar, etc., sin importar la forma en la cual estos [los métodos, sistemas, etc.] sean descriptos, explicados, ilustrados o incorporados en la obra.

No hay nada de ambiguo o de poco claro en dicha norma.

Lo que se dice es claro, afirma Google.

Entonces, aunque generalmente a la autoría de una obra original le sea otorgada protección bajo derechos de autor, la protección concedida a la obra no se extiende a los sistemas o métodos de operación incluidos o incorporados en dicha obra.

La exclusión legal es *explícita y absoluta* cuando la norma se refiere a *"sin importar la forma"* en la cual [el sistema o método de operación] sea descripta, explicada, ilustrada, o incorporada en la obra.

Afirma Google, que la Corte Suprema de Justicia ha tenido oportunidad de expresar mucho tiempo antes de la sanción de la ley de derechos de autor de EE.UU. en el año 1976, en el caso judicial *Baker v. Selden* del año 1880, que los sistemas y los métodos de operación (junto con los elementos específicos de la expresión que son *"incidentes necesarios"* para ellos) no son protegibles bajo derecho de autor.

Entonces, este caso define la importancia de aplicar el artículo 102 inciso b) como está redactado.

Los encabezados de los métodos de Java o el código fuente declarado de Java, los cuales permiten a los programadores utilizar los comandos abreviados *basados* en el mismo *header* o código fuente declarado, a los cuales estos están familiarizados, constituyen un método para operar "los programas pre-escritos de la plataforma Java".

Si Google no hubiera replicado exactamente los encabezados de los métodos de Java, por ejemplo:

Package.java.lang
public class Math
public static int max (int x,int y)

el código usado por esos *comandos abreviados* -basados en esos mismos encabezados o código declarado- no hubieran corrido o podido ejecutarse en Android.

java.package.Class.method()

Google se esforzó por replicar sólo los elementos necesarios para permitir a los programadores de aplicaciones en Android de poder usar los mismos *"comandos abreviados",* por ello sólo copió los "*headers"* o código declarado y no así el código fuente implementado que es el implementa o ejecuta los métodos.

Si Google no hubiera copiado los "*headers"* o código declarado de Java en forma exacta (para los 37 paquetes copiados) los *"comandos abreviados"* no hubieran funcionado, y de esa forma los programadores de aplicaciones en Android no hubieran podido reutilizar el conocimiento adquirido -acerca de cómo invocar las funciones pre-escritas de la librería de software estándar de Java en Android- en forma previa al momento de conocer el lenguaje Java y sus paquetes de API preconstruidos.

El sistema de derechos de autor, no puede bloquear este sistema o método de operación de la misma manera que no puede bloquear el sistema de teclado QWERTY.

Al presionarse en una tecla sobre un teclado QWERTY el teclado envía un *comando* que causa que la computadora ejecute una determinada función, como el de escribir la letra Q en una pantalla.

QUERTY constituye dos cosas: 1) un diseño de teclado y 2) una estructura de comandos para causar en las computadoras todo tipo de producción de letras y símbolos, de la misma manera que los *headers* o código fuente declarado son la estructura de comandos para utilizar los programas pre-escritos en Java y en Android.

Que los programadores de desarrollo de aplicaciones hayan invertido tiempo y recursos en el aprendizaje de esos comandos abreviados ***java.package.Class.method()***, confirma el hecho de que los "correspondientes encabezados de métodos o código declarado" del cual *derivan* los "comandos abreviados" constituye un sistema o método para operar los programas pre-escritos incluidos en la plataforma.

Google no disputa en absoluto que replicó los *headers* o código fuente declarado que eran más importantes para los dispositivos móviles, y lo hizo precisamente a razón del efecto de bloqueo o *lock-in*: como los usuarios de computadoras que están familiarizados con el teclado QWERTY, los programadores estaban acostumbrados a utilizar los *comandos abreviados de Java basados en el código declarado* o *headers*.

La decisión de Google fue no utilizar más de aquello que fuera necesario para una plataforma para dispositivos móviles.

De hecho, este caso es un claro ejemplo de la importancia de la compatibilidad y de los efectos de bloqueo o lock-in.

Los programadores han invertido tiempo y esfuerzo en aprender el lenguaje de programación Java, incluyendo el *formato de comandos abreviado.*

Y ahora, mucho tiempo después de que se ha atraído a los programadores a la comunidad Java, y después de cualquier protección en materia de patentes de invención que ya hubiera vencido, Oracle, el sucesor de Sun Microsystems intenta construir un muro alrededor de los *headers* o código fuente declarado de Java.

Por lo expuesto Google afirma que el código fuente declarado o *headers* no son protegibles por los derechos de autor de EE.UU.

A su vez, Google sostiene que los encabezados de los métodos o código fuente declarado constituye una *interface de software,* y, por lo tanto, no protegible por el sistema de derechos de autor de EE.UU.

Cuando IBM creó la computadora personal, desarrolló una interface llamada *"Basic Input/Output Systems"*.

Sus competidores como COMPAQ y Phoenix *reimplementaron* el sistema y crearon sus propias computadoras compatibles con IBM, y ello incrementó el número de opciones disponibles para los consumidores.

Apple, utilizó aplicaciones de programación de interface existentes en UNIX en el sistema operativo de sus computadoras, permitiendo a los programadores familiarizados en UNIX escribir programas de computación que pudieran ejecutarse o correr en las computadoras de Apple.

Oracle, construyó su sistema operativo Linux de la misma manera.

Y en otro orden, para competir en el mundo de los procesadores de texto, Microsoft *reimplementó* las interfaces de WordPerfect para utilizarlas en Microsoft Word.

La necesidad y la existencia de las interfaces de software, sin tenerse responsabilidad alguna por infringir derechos, son esenciales, y más hoy en un mundo interconectado.

Las leyes locales e internacionales reflejan la importancia de proteger el derecho al uso de las interfaces.

Por un lado, el Congreso de los EE.UU., cuando se refiere a la posibilidad de realizar ingeniería inversa para propósitos de identificar y analizar aquellos elementos de un programa de computación que son necesarios para lograr la interoperabilidad de un programa creado en forma independiente con otros programas.

Por otro lado, la Directiva de Software de la Unión Europea 91/250/EEC del 14 de mayo de 1992, dispone excepciones similares para casos de ingeniería inversa para *blackboxes* o "cajas negras".

Estas leyes tienen sentido, porque una vez que se identifica y analiza el código de computación de un programa que es necesario para lograr la interoperabilidad, los programadores poseen el derecho de usarlo, tal cual como lo hizo Google en este caso.

Así recientemente en el caso Europeo *SAS Institute Inc. v. World Programming Ltd* (Caso C-406/10, año 2012) se ha dicho que ninguna funcionalidad de un programa de computación ni de un lenguaje de programación y de los formatos de archivos de datos usados en un programa de computación con el objeto de ejecutar sus funciones constituyen una forma de expresión de

esos programas, y como tales, no están protegidos por los derechos de autor.

Lo expuesto revela que la comunidad de desarrolladores desde hace tiempo tenía el entendimiento de que las interfaces son libres de ser utilizadas por cualquiera.

Este entendimiento es el que ha permitido la innovación.

La sentencia de la Corte Federal de Apelaciones, impide la interoperabilidad, y priva a los consumidores de los beneficios de la compatibilidad.

En forma previa a decidir sobre el *Writ of Certiorari* interpuesto por Google, la Corte Suprema de los EE.UU. solicitó la opinión del Abogado General de los EE.UU.[94] quién el 26 de mayo del 2015 expresó estar de acuerdo con lo resuelto por el Tribunal de Apelaciones del circuito Federal, y por ello, recomendó a la Corte Suprema la denegación de revisión del *Writ of Certiorari* presentado por Google.

Posteriormente, la Corte Suprema de Justicia denegó el *Writ of Certiorari*, por lo tanto, los argumentos de Google no fueron revisados.

[94] Donald B. Verrili, Jr. Solicitor General (Abogado General de los EE.UU. durante la Administración Obama)

CAPITULO IX

RECURSO DE APELACIÓN INTERPUESTO POR ORACLE CONTRA LA DECISÓN DEL JURADO Y DE LA CORTE DE DISTRITO[95] RELACIONADOS A LA TEORÍA DE "USO JUSTO (o LEGITIMO)"

Teniendo en cuenta la decisión del Tribunal de Apelaciones del Circuito Federal del 9 de mayo de 2014 en relación al asunto del uso justo, una vez llevado a cabo el *segundo juicio*, el Jurado determino que el uso realizado por Google debía ser considerado uso justo.

Así, también lo entendió el Juez de la Corte del Distrito Norte de California.

Contra dicha decisión, el 10 de febrero de 2017 Oracle presentó su apelación ante el Tribunal de Apelaciones del Circuito Federal.

En la sentencia I del Tribunal de Apelaciones del Circuito Federal, se mencionó que no había suficiente evidencia en relación a los factores relevantes del uso justo como para que el Tribunal de Apelaciones del Circuito Federal se expidiera sobre ello, y por ello en cambio ordenó que se llevara a cabo un nuevo juicio en relación al uso justo.

En efecto, el Tribunal de Apelaciones del Circuito Federal mencionó que de las pruebas rendidas en el juicio -hasta la fecha de dictado de la sentencia I de apelación- no surgía con claridad, que cantidad de lo copiado por Google, es decir, de los paquetes

[95] District Court for the Northern District of California. Order Denying Rule 50 Motions. No. C 10-03561 WHA, June 8, 2016.

de Java, "*eran componentes esenciales"* de cualquier programa desarrollado en el lenguaje de programación Java.

Esta información era necesaria e importante para el "segundo juicio" en relación al *segundo factor* (naturaleza de la obra protegida por derechos de autor) y *tercer factor* (cantidad y sustancialidad de la parte utilizada de la obra protegida en su conjunto) del uso justo.

Todo esto quedó resuelto en el segundo juicio, en donde se estipuló que de las 11.500 líneas de código copiado por Google sólo 170 líneas eran necesarias para escribir y utilizar el lenguaje de programación Java.

Es decir, menos del 1,5% de lo que Google copió.

También, en el juicio Google reconoció que utilizaba las API de Java en Android con el mismo propósito que esas API poseían en Java.

La plataforma Java (Java SE) es un programa de computación para desarrollar y correr o ejecutar aplicaciones ("apps") escritas o desarrolladas en el lenguaje de programación Java.

La plataforma Java permite a los programadores escribir sus propios programas para que estos puedan ejecutarse en diferentes computadoras, más allá de su arquitectura o sistema operativo, sin tener que reescribir sus programas para cada tipo de sistema.

Oracle hace disponible la plataforma Java para los programadores en el desarrollo de sus aplicaciones sin que existan tarifas aplicables o pago de precio alguno.

Por el contrario, Oracle cobra un precio en concepto de licencia a:

(i) aquellos que quieren utilizar las API de Java en una plataforma en competencia con Java, o
(ii) cuando se encuentran incluida en un dispositivo electrónico.

Con la finalidad de preservar el *"write once, run anywhere"* Oracle requiere de los licenciatarios cumplir con los estrictos requisitos de compatibilidad (TCK).

Conforme constancias presentadas en el juicio en el año 2005 Java se encontraba licenciado en unos 700 millones de computadoras personales (PCs), como en unos 40 millones de dispositivos de televisión (TVs), en dispositivos *e-readers* y en varios dispositivos más de otros tipos (automóviles).

El mercado de la telefonía móvil fue particularmente lucrativo.

Así, Java se convirtió en la plataforma líder para desarrollar y correr aplicaciones en los teléfonos móviles.

En el año 2005, la plataforma Java estaba en más de 1 billón (o lo que es lo mismo 1.000 millones) de teléfonos móviles.

En el año 2006 Java alimentaba a casi al 100% de los teléfonos móviles inteligentes.

En el año 2008, Sun Microsystems trabajó en una nueva generación de plataforma Java para telefonía móvil basada en Java SE.

Sun Microsystems estaba listo para licenciar su plataforma Java para la nueva generación de dispositivos móviles.

En el año 2005 Google afrontó una amenaza existencial.

Google genera ingresos vendiendo publicidad en conexión con los resultados de búsquedas en internet.

El punto fue que los consumidores cada vez más realizaban dichas búsquedas en internet, pero no desde sus computadoras personales o notebooks, sino desde sus teléfonos móviles, y el buscador de Google no estaba optimizado para los teléfonos móviles.

Con lo cual, Google enfrentaba el inmenso riesgo de perder una porción más que importante en el mercado de búsquedas por internet (en inglés "i*nternet market share"*) si es que no actuaba rápidamente en el desarrollo de una nueva tecnología.

La solución de Google se llamó Android.

Así, Google sabía que para que una plataforma fuere exitosa necesitaba:

1) paquetes de API que los fabricantes de dispositivos móviles confiaran los suficiente como para aceptar instalarlos en ellos, y

2) una comunidad de programadores quién mejoraría o incrementaría el atractivo del consumidor de cualquier dispositivo, al escribirse innumerables aplicaciones para dichos dispositivos.

Google podría haber escrito desde cero sus paquetes de API, como lo hizo Microsoft, Apple u Oracle (antes de la adquisición de Sun Microsystems), pero no lo hizo.

Google no tenía tiempo.

Pero en realidad había algo peor.

Una vez creada, los programadores necesitarían aprender a utilizarla, y construir una comunidad de programadores alrededor de las API de Google para telefonía móvil.

Con la salida del *Iphone* de Apple en enero de 2007, Google estaba fuera de tiempo.

Con lo cual al copiar las API de Java Google redujo su tiempo de desarrollo, y cumplió con ambos objetivos a la vez.

En primer lugar, produciría una plataforma instantáneamente aceptable para los fabricantes de dispositivos, y, en segundo lugar, podría acceder a los 6 millones de desarrolladores de Java en el mundo para construir aplicaciones en Android.

Conforme constancias del juicio, Google fue avisado por uno de sus fundadores que las API de Sun Microsystems estaban

protegidas conforme a la ley de derechos de autor de EE.UU., y que Google debía buscar una licencia.

Con lo cual, Google quiso obtener una licencia de Sun Microsystems pero en términos que estaban en detrimento del modelo de licenciamiento de Sun Microsystems.

Google insistía que Sun Microsystems (Oracle) le permitiera a los fabricantes de dispositivos usar las API de Java en Android sin costo alguno, y sin límites en cuanto a la modificación de su código.

El punto era que Sun Microsystems estaba obteniendo ingresos de USD 100 millones anuales licenciando la plataforma Java para fabricantes de dispositivos móviles, y a diferencia de Google, Sun Microsystems no obtenía ingresos desde la publicidad.

Además, permitir la modificación del código de Java sin control (sin pasar el TCK) crearía posibles incompatibilidades en Java y pondría en riesgo el principio de "*write once, run anywhre*"

Como las opciones investigadas por Google no eran buenas[96] las únicas opciones para Google eran:

(i) abandonar el asunto, o

(ii) "usar las API de Java de cualquier manera y defender su decisión, haciendo enemigos en el camino"[97].

[96] Conforme prueba documental "...the alternatives all suck...we need to negotiate a license for Java" Appx54012

[97] Conforme prueba documental "do Java anyway and defend our decision, perhaps making enemies along the way" Appx54010-54011

Google tomó la opción (ii) y copió en forma exacta el código declarado de 37 paquetes de las API de Java, y la estructura, secuencia y organización (sus siglas en inglés "SSO") de las API de Java.

Posteriormente Google, *parafraseó* el resto implementado su propio código teniendo en cuenta la especificación de la API.

Google copió lo más neurálgico e importante de la plataforma Java, aquello que razonablemente cualquiera podría entender útil para la tecnología de dispositivos móviles de última generación.

En definitiva, Google copió lo que era esencial para Android, es decir, el *"core"* (núcleo, haciéndose referencia a lo central o principal de Android) de lo que dependía Android.

Entonces, habiendo copiado las API de Java y su estructura, secuencia y organización, Google obtuvo lo que no nunca hubiese obtenido con una licencia de Sun Microsystems.

Es decir, Google no sólo le otorgó Android sin costo alguno a los fabricantes de dispositivos, sino que hizo Android incompatible con Java, es decir que un programa escrito para la plataforma Android no corre o no funciona en la plataforma Java, y viceversa.

Así de la noche a la mañana, Android, conforme constancia del juicio[98], se transformó en un competidor para Java con focalización en la misma industria con similares productos.

La misma Corte de Distrito dijo "Android es una plataforma de software diseñada para dispositivos móviles y compite con Java en el mercado".

[98] Véase como referencia Appx50844.

Pero Google no era un competidor cualquiera, sino uno que atraía o captaba a los clientes de Oracle dándole gratis lo que Oracle licenciaba por el pago de un precio a esos clientes.

No había forma de competir contra eso.

En reuniones privadas con clientes de Oracle, como LG, AT&T, QUALCOMM, el grupo de ventas de Google hacia presentaciones promocionando Android como "La principal librería de Java" (en inglés "Core Java Libraries", o *"Java API"* o simplemente "*Java Application Framework")*. [99]

Android generó más de 42 billones (o lo que es lo mismo 42 mil millones) de dólares en ingresos, mientras que las consecuencias para Oracle fueron devastadoras.

Así el negocio de Oracle comenzó a desaparecer.

Si bien Oracle podía continuar con la obtención de ciertos ingresos en relación al licenciamiento de Java SE en computadoras personales, el licenciamiento relacionado a la telefonía móvil cayo drásticamente.

Clientes de Oracle cambiaron rápidamente hacia Android, así lo hicieron Samsung, Motorola, HTC y ZTE.

El "*market shared*" de Oracle cayó de 80% a 0%, y el de Google creció de 0% a 80%[100].

PRIMER FACTOR. PROPÓSITO Y CARÁCTER DEL USO

[99] Véase conforme referencia Appx 54501,54503,54505,54509, entre otros mencionados en la presentación de Oracle.
[100] Véase conforme referencia Appx1104 mencionado en la presentación de Oracle.

El uso de Google es puramente comercial.

Ello no sólo surge de las constancias del juicio, sino del reconocimiento hecho por Google cuando fue preguntado por el Juez de la Corte de Distrito, y Google contestara que el propósito del uso era enteramente comercial.

Cuando una obra es significativamente comercial, como es el caso, el primer factor se inclina en contra del infractor a menos que el uso sea realmente "transformativo".

Ahora, el uso de Google no fue transformativo.

Copiar un material para ser usado con los mismos propósitos que el titular de derechos no es transformativo.

No hay dudas de que las API de Java sirven con el mismo propósito en Android que en Java.

Oracle utiliza su material protegido para permitir a los programadores en el desarrollo de sus propias aplicaciones, recordar, localizar y hacer correr los programas empaquetados en la plataforma Java.

Google utiliza el material protegido de Oracle para permitir a los programadores recordar, localizar y hacer correr los programas empaquetados para que ejecuten las mismas funciones en Android.

Cuando un programador de Java en el desarrollo de su aplicación, quiere que la misma ejecute una función de abrir una conexión de internet utilizaría la función que conoce y tipearía:

"newURL("[website´sURL]").openConnection()"

Google diseño Android para que un programador tipeara el mismo código fuente (arriba expuesto) para lograr el mismo resultado en la plataforma Android.

Por ello en el juicio Google reconoció que incluir el código declarado, y toda su estructura, secuencia y organización, fue para beneficio de los programadores, quienes tenían ciertas expectativas, incluyendo las de poder acceder al conjunto de las API de Java.

Google expresa que su uso es transformativo porque copió la API de Java diseñadas para computadoras personales y la llevo en forma idéntica a un supuesto nuevo contexto diseñado para teléfonos móviles inteligentes.

La teoría de Google es errónea porque ese no era ni siquiera un nuevo contexto para las API de Java.

Por otro lado, el uso de Google no se relaciona con lo estipulado en el artículo 107 [101] que se refiere a *"críticas, comentarios, enseñanza, reportaje de noticias".*

[101] El artículo 107 (relativo al uso justo), Titulo 17 dispone "Limitaciones de los derechos exclusivos. Uso Justo. "Sin perjuicio de lo establecido en los artículos 106 y 106A, el uso justo de una obra protegida por derecho de autor, incluido el uso por reproducción en copias o registros fonográficos o por cualquier otro medio especificado en esa sección, *con fines tales como críticas, comentarios, informes de noticias, enseñanza* (incluyendo copias múltiples para uso en el aula), *beca o investigación, no constituye una infracción de los derechos de autor.* Para determinar si el uso que se hace de una obra en un caso particular es un "uso justo", los factores a ser considerados incluirán: (1) el propósito y el carácter del uso, incluyendo si dicho uso es de naturaleza comercial o para propósitos educativos sin fines de lucro; (2) la naturaleza de la obra protegido por derechos de autor; (3) la cantidad y la sustancialidad de la parte utilizada en relación a la obra protegida por derechos de autor en su conjunto; y (4) el efecto del uso sobre el mercado potencial o el valor de la obra protegida por derechos de autor. El hecho de que una obra sea inédita no impedirá en sí mismo un hallazgo de uso justo o

En esos casos, la expresión del propósito de la obra protegida esta materialmente alterado de una forma que no se apropia en forma indebida del derecho de crear obras derivadas del autor de la obra originaria.

El uso de Google podría haber sido transformativo si al copiar las API de Java las hubiera usado con un propósito diferente como por ejemplo el de enseñanza a diseñar API.

Ahora, *reempaquetar* el mismo código protegido desde una plataforma (Java) a otra (Android) sin cambiar el propósito no es transformativo.

El derecho de autor le otorga a Oracle el derecho exclusivo de adaptar su obra original.

Así, el mismo derecho de autor no condona una adaptación de la obra sin autorización con fines comerciales y de lucro como uso justo.

Finalmente, existe amplia evidencia que Google intencionalmente copió sabiendo que la obra de Oracle estaba protegida bajo derechos de autor.

La mala fe pesa *en contra* del uso justo.

SEGUNDO FACTOR. LA NATURALEZA DE LA OBRA PROTEGIDO POR DERECHOS DE AUTOR

El mismo Tribunal de Apelaciones del Circuito Federal ha tenido oportunidad de reconocer que el diseño de las API de Java era un proceso creativo y que los desarrolladores de Sun

legítimo si dicho hallazgo se realiza considerando todos los factores anteriores".

Microsystems tuvieron una gran cantidad de opciones en su expresión.

Por su parte, el mismo Google reconoció que las consideraciones funcionales no eran predominantes en el diseño de las API.

De hecho, nadie en el juicio, conforme constancias en el mismo, contradijo que la creación de las API de Java no era creativa, de la misma forma que no existe en el pleito ningún testimonio que haya dicho que la estructura y organización de las API no era creativo.

Por otra parte, la conclusión del Juez del Distrito que menciona que "debido a que las API de Java poseen un rol funcional no pueden ser consideradas creativas" está en contra de todos los casos judiciales de derechos de autor resueltos relacionados a programas de computadoras.

En su lugar, se debe ver el grado de creatividad dentro del universo funcional del programa de computación. Y desde ese punto de vista, no hay duda alguna que las API de Java son creativas.

TERCER FACTOR. LA CANTIDAD Y LA SUSTANCIALIDAD DE LA PARTE UTILIZADA EN RELACIÓN A LA OBRA PROTEGIDA POR DERECHOS DE AUTOR EN SU CONJUNTO

Sin lugar a dudas, Google copió el "corazón" de Java.

Google copió 11.500 líneas de código y la estructura y organización de los 37 paquetes neurálgicos y más importantes de las API de Java.

Según constancias del juicio, Google copió los paquetes de Java que alguien esperaría razonablemente encontrar para ser usados en relación a dispositivos móviles de última generación.

En su oportunidad el Tribunal de Apelaciones del Circuito Federal se encontró con limitaciones para expedirse en relación al uso justo, ya que mencionó que no había suficiente evidencia en relación a que cantidad de los paquetes copiados por Google eran “mandatorios o esenciales” en cualquier programa basado en el lenguaje de programación Java.

Entonces, durante el segundo juicio se estableció que 170 líneas del código de 3 de los 37 paquetes de Java eran necesario para escribir o desarrollar con el lenguaje Java.

Por otra parte, Google reconoció que copió 11.500 líneas de código, es decir 11.330 líneas más de lo necesario para desarrollar en el lenguaje Java.

La Corte de Distrito sugirió que Google había copiado sólo lo suficiente para preservar la consistencia entre las plataformas -queriendo decir con ello- para evitar la confusión entre los programadores de Java y los de Android.

Es decir, Google quería que los desarrolladores confiaran en su familiarización con las API de Java para conducir el éxito comercial de Android.

O como dijo el mismo Tribunal de Apelaciones del Circuito Federal, el objetivo de Google era capitalizar el hecho de que los programadores de software ya estaban entrenados y con experiencia en utilizar las API de Java.

CUARTO FACTOR. EL EFECTO DEL USO SOBRE EL MERCADO POTENCIAL O EL VALOR DE LA OBRA PROTEGIDA POR DERECHOS DE AUTOR

De las constancias del pleito surge que Android y Java son competidoras y que ambos focalizaron en la misma industria con productos similares.

No cabe duda, conforme constancias, que Oracle licenció Java SE y Java ME para los teléfonos móviles por años, antes de que Android fuera lanzado al mercado.

En la época del lanzamiento de Android, Java estaba licenciado aproximadamente en el 80% de los teléfonos móviles, obteniendo Sun Microsystems millones de dólares en concepto de licenciamiento de parte de los fabricantes más grande del mundo de dispositivos móviles. Esto incluyó a Nokia y Danger que se licenciaban en Java SE para sus teléfonos móviles inteligentes.

Posteriormente, debido a Android, los ingresos en concepto de licencias se desplomaron.

El 80% del "*market share"* que Java tenía en los teléfonos móviles paso a ser de Android.

Eso fue debido al resultado directo de competencia en ese mercado.

Por ejemplo, el dispositivo de HTC Dream basado en Android competía directamente con el HTC Touch Pro basado en Java.

Posteriormente, Motorola, Samsung y Sony Ericsson se cambiaron a Android.

Similarmente, Android causó el mismo impacto negativo en el sector de "tabletas de e-readers".

Amazon estaba licenciado en Java SE para su Kindle, y luego se cambió a Android pare el Kindle Fire.

Luego, para su última generación de Kindle, Amazon volvió a Java SE pero requirió un descuento del 97,5% teniendo en cuenta que existía en el mercado en forma gratuita Android.

De tal forma Android causó la imposibilidad para Oracle de explotar cualquier nuevo potencial mercado para Java SE, en relación a los avances de la tecnología relacionados a la telefonía móvil.

La Corte de Distrito se equivoca al afirmar que el mercado de las computadoras personales era el único mercado de Oracle.

Como se expuso Oracle licenciaba Java SE y Java ME para teléfonos móviles, uno de los mercados en donde Android competía con Java. Ahora, independientemente de todo ello, el mercado de teléfonos móviles constituía indudablemente un mercado para nuevas obras derivadas de Oracle.

Daño o potencial daño en cualquier de estos mercados es suficiente.

Por otro lado, la Corte de Distrito mencionó que el mercado de la plataforma Java podría haber sido diezmado sin que Android hubiera tenido que ver debido a que Sun Microsystems hizo disponible las API de Java mediante un modelo de licenciamiento de software de código abierto llamado OpenJDK.

Ahora, bien, teniendo en cuenta lo probado en el juicio OpenJDK no tuvo esos efectos por la sencilla razón que el esquema de licenciamiento de OpenJDK, -que fue la licencia GPLv2- imponía a los licenciatarios fabricantes de los

dispositivos móviles en cumplimiento de los términos de la licencia GPLv2 distribuir las modificaciones que estos hicieran, es decir, las obras derivadas de Java SE en cuanto se produjera la distribución de sus productos.

Con lo cual, esta obligación de tener que distribuir esas "modificaciones" era algo que los fabricantes no desean realizar, ya que querrían conservar sus modificaciones como ventaja competitiva.

De hecho, el mismo Google considero OpenJDK para Android, y finalmente lo descartó por estas mismas razones[102].

[102] Véase la presentación realizada el 8 de Diciembre de 2014 por las organizaciones Software Freedom Law Center y la Free Software Foundation en favor de Google "*Brief of Software Freedom Law Center and Free Software Foundation, Amici Curiae in support of respondent* https://softwarefreedom.org/resources/2014/google_v_oracle-sflc_cert_amicus.pdf donde según estas Google habría actuado erróneamente al asumir ciertas restricciones impuestas por la Licencia GPLv2, diciéndose así: *There is no dispute between the parties that Petitioner can use, copy, modify and redistribute all of the putatively copyrighted material at issue, royalty free, under the terms of the most widely-used free software copyright license, the GNU General Public License, version 2, published by amicus Free Software Foundation. The Free Software Foundation also publishes a Java programming language system, including standard class libraries declaring the same method names, under the GNU GPL license, version 2 or any later version.* It appears that Petitioner wrongly supposed at some time in the past that use of Respondent's Java source code under the terms of GNU GPLv2 *would limit the licensing of the Java programs that third parties could install and run on Android systems.* Andy Rubin, then an important Google executive with responsibility for Android, wrote an email to that effect which was part of the evidence at trial: *[A]s far as GPLing the VM, everything that is linked with the VM would get infected. The problem with GPL in embedded systems is that it's viral, and there is no way (for example) OEMs or Carriers to differentiate by adding proprietary works. We are building a platform where the entire purpose is to let people differentiate on top of it*. (Trial Exhibit 230, at 1). Y por ello se dijo: "This was simply wrong. As users of the

Con lo cual, OpenJDK realmente no tuvo ningún impacto en el negocio de licenciamiento de la plataforma Java.

Por último, Google menciona como ejemplo del expandido uso sin necesidad de licenciamiento de las API de Java al proyecto de Apache Harmony de la Fundación Apache.

Java programming language as supplied historically by Sun Microsystems and now by Respondent Oracle well know, though the Java language and standard classes are available for all to copy, modify and redistribute freely, without payment of royalties, under the GNU GPL, these parties develop, use, and distribute Java programs for execution by Oracle Java under a great variety of proprietary and free software copyright licenses". Petitioner Google is now and has been entitled to use all the material at issue in this case, royalty free, under GNU GPL. Although Petitioner *unaccountably failed to assert and preserve this defense of license at trial, it was in fact at all times licensed under the GNU GPL, version 2, to take all the steps it took in relation to the Java standard class declarations by Respondent Oracle or by its predecessor in interest, Sun Microsystems*.

CAPÍTULO X

SENTENCIA II[103] DEL TRIBUNAL DE APELACIONES DEL CIRCUITO FEDERAL DE EE.UU. RELACIONADA AL USO JUSTO (O LEGITIMO)

ARGUMENTOS RELACIONADOS CON EL USO JUSTO (O LEGÍTIMO)

Recuérdese que el Tribunal de Apelaciones del Circuito Federal en su sentencia del 9 de mayo de 2014 (sentencia I: *Copyrighability"*) sentenció a favor de Oracle en relación a la protección del código fuente declarado y a su estructura, secuencia y organización (sus siglas en inglés "SSO") pero no resolvió al respecto de si el uso realizado por Google encuadraba dentro del uso justo (conforme artículo 107, Titulo 17, USC. "Limitaciones a los Derechos Exclusivos"), sino que ordenó que se llevara a cabo un *segundo juicio* para que se determinara si el uso realizado por Google en relación al código declarado y a la estructura, secuencia y organización de los paquetes de las API de Java podría ser considerado como "uso justo" (o legítimo).

Así las cosas, habiéndose llevado a cabo este segundo juicio, el Jurado después de escuchar durante una semana los argumentos dado por las partes, y luego de tres días de deliberación emitió veredicto unánime a favor de Google y así conforme a ello la Corte de Distrito emitió sentencia (en inglés "*final judgement"*) con fecha 8 de junio de 2016, concediendo que el uso realizado por Google había sido "uso justo" (o legítimo).

[103] Véase en http://www.cafc.uscourts.gov/sites/default/files/opinions-orders/17-1118.Opinion.3-26-2018.1.PDF

Posteriormente a ello Oracle presentó su recurso de apelación, y con fecha 27 de marzo de 2018 el Tribunal de Apelaciones del Circuito Federal, integrado por los jueces O´Malley, Plager y Taranto, revirtió la decisión del grado anterior fallando a favor de Oracle y reconociendo que el uso realizado por Google no constituía un uso justo de acuerdo con lo establecido en el artículo 107, del Título 17 del USC.

El artículo 107[104], Titulo 17 del USC[105] se refiere a las limitaciones de los derechos exclusivos, y establece lo siguiente: *"Limitaciones de los derechos exclusivos. Uso Justo. "Sin perjuicio de lo establecido en los artículos 106 y 106A, el uso justo de una obra protegida por derecho de autor, incluido el uso por reproducción en copias o registros fonográficos o por cualquier otro medio especificado en esa sección, con fines tales como críticas, comentarios, informes de noticias, enseñanza (incluyendo copias múltiples para uso en el aula), beca o investigación, no constituye una infracción de los derechos de autor. Para determinar si el uso que se hace de una obra en un*

---

[104] En su versión original en inglés: "*Section 107. Limitations on exclusive rights: Fair use. Notwithstanding the provisions of sections 106 and 106A, the fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use upon the potential market for or value of the copyrighted work. The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors".*

[105] Ver en https://www.copyright.gov/title17/title17.pdf

caso particular es un "uso justo", los factores a ser considerados incluirán: (1) el propósito y el carácter del uso, incluyendo si dicho uso es de naturaleza comercial o para propósitos educativos sin fines de lucro; (2) la naturaleza de la obra protegido por derechos de autor; (3) la cantidad y la sustancialidad de la parte utilizada en relación a la obra protegida por derechos de autor en su conjunto; y (4) el efecto del uso sobre el mercado potencial o el valor de la obra protegida por derechos de autor. El hecho de que una obra sea inédita no impedirá en sí mismo un hallazgo de uso justo o legítimo si dicho hallazgo se realiza considerando todos los factores anteriores".

La teoría de la defensa del uso justo comenzó como una creación de la doctrina judicial de los EE.UU. y luego fue codificado en el artículo 107 de la ley de derechos de autor de 1976 de los EE.UU.

El uso justo funciona como una limitación de excepción a los derechos exclusivos del autor de la obra protegida que permite el uso de la obra protegida para fines como críticas, comentarios, informes de noticias, educación, investigación, entre otros.

En el lenguaje original se utiliza la palabra *"such as"*, es decir, "tales como" lo que da la pauta conforme a la doctrina de los EE.UU. que la enumeración brindada por el articulo 107 *no es taxativa*, es decir ejemplifica algunas actividades que podrían quedar comprendidas, y de alguna forma podrían ser usados por los jueces de manera referencial.

Por otra parte, el artículo 107 requiere que exista una determinación caso a caso sobre si en una determinada circunstancia existe o no un uso justo, y para ello el mismo artículo 107 establece que se debe hacer el análisis de los cuatro factores expresados en la misma norma.

El análisis a realizarse por los jueces implica realizar un *balance* de estos cuatro factores, en el cual los jueces deben considerar si el objetivo de las leyes de derechos de autor de los EE.UU. de promover el progreso de la ciencia y las artes están mejor representada "permitiéndose dicho uso bajo análisis o impidiéndolo".

Desde un punto de vista del derecho procesal de lo EE.UU., teniendo en cuenta que el *"fair use"* constituye una "defensa afirmativa" contra un reclamo por infracción de derechos, Google tenía la *carga de probar* que los factores jugaban a su favor, sin que ello significase que debía lograrlo respecto de los cuatro factores.

PRIMER FACTOR. EL PROPÓSITO Y EL CARÁCTER DEL USO

Este factor implica si el uso tuvo naturaleza *"*comercial*"* o por el contrario fue para propósitos "educacionales no lucrativos".

A su vez, este factor posee dos componentes, el primero, ya mencionado, es si el uso es de naturaleza comercial o si lo es para fines de educación o de interés público, y el segundo componente se relaciona a si la obra nueva creada es "transformativa" o si simplemente suplanta a la original.

El uso de la obra protegida que es "comercial" tiende a sopesar en contra del uso justo.

Ahora bien, a través de distintos casos judiciales en EE.UU., se ha reconocido que en la mayoría de ellos se busca algún grado o medida de uso comercial, con lo cual, enfatizar excesivamente

en la motivación comercial del copista podría llevar a causar una posición más que restrictiva o exagerada del uso justo.

Google ha sostenido que (1) teniendo en cuenta que Android se licencia mediante una licencia de código abierto (Licencia Apache versión 2) el Jurado podría haber concluido que Android no posee propósitos comerciales, y (2) que el Jurado podría haber razonablemente encontrado que los ingresos de Google provienen de la publicidad de su buscador, el cual es preexistente a Android.

El Tribunal de Apelaciones del Circuito Federal sostuvo que ninguno de ambos argumentos tenía mérito ya que, en primer lugar, se sostuvo que la pregunta no es si el único motivo del uso es la ganancia monetaria, sino si el usuario puede beneficiarse de la explotación del material protegido por derechos de autor sin pagar por ello, y a su vez, aunque los ingresos de Google provengan de la publicidad, y no de Android, el carácter de "comercial" no depende de cómo Google obtenga ganancias.

En definitiva, el beneficio económico directo no es requerido para demostrar "uso comercial".

Por estas razones, el Tribunal de Apelaciones del Circuito Federal entendió que el uso de los paquetes de las API de Java sopesaba en contra del uso justo, en definitiva, Google no había hecho un uso justo.

En relación al requisito de "uso transformativo", si bien la ley de derechos de autor de EE.UU. no lo establece, con la denominación de "transformativo" la Corte Suprema de los EE.UU. ha establecido que el propósito central del primer factor del uso justo es determinar si la nueva obra "es transformativa y, en su caso en que extensión o grado".

Se expresa entonces que un uso es "transformativo" si agrega algo nuevo, con propósitos y carácter distinto, que alteran el primer uso con una "nueva expresión, significado o mensaje".

Una de las preguntas críticas que surgen en el análisis de este uso transformativo, es si la nueva obra creada "reemplaza, sustituye o suplanta el objeto" de la creación originaria o si simplemente agrega algo nuevo.

Así la pregunta anterior debe entenderse dando una mirada a la enunciación proporcionada por el artículo 107 en cuanto a que el uso se refiere a críticas, comentarios, noticias, etc.

Así la Corte Suprema de EE.UU. ha reconocido que las *parodias*, como también otras obras que comentan y producen críticas, son por su naturaleza suficientemente transformativas para encajar claramente en la excepción del uso justo[106].

Por ello para encuadrar en el estándar requerido de "transformativo" la segunda obra en cuestión debe alterar la obra original con una nueva expresión, significado o mensaje, o servir con un propósito diferente de aquel tenido en cuenta por la primera obra.

Cuando el segundo uso posee los mismos propósitos que el de la obra del mismo autor, tal uso debilita y compromete la alegación de que el uso es justo.

También, cuanto más "transformativo" es el carácter de la segunda obra menos importancia cobra el significado de comercialización.

El Juez de Distrito mencionó respecto del código declarado copiado por Google, *que éste servía para cumplir la misma*

[106] Véase el caso *Mattel Inc. V. Walking Mountain Prods., 353 F.3d 792, 800 (9th Cir. 2003)*

función en ambas obras, es decir en Android y Java, porque por definición el código declarado en el lenguaje de programación Java cumple con los propósitos específicos de definición.

Sin embargo, concluyó que el Jurado podría haber encontrado de forma razonable que la selección de algunos -no así de todos- de los paquetes de las API de Java, con su nuevo código implementado "adaptado a las limitaciones de sistema del nuevo ambiente para los teléfonos móviles inteligentes, junto con los nuevos métodos, clases y paquetes escritos o desarrollados por Google para la plataforma de teléfonos móviles inteligentes Android, constituyen un contexto nuevo dándole nueva expresión, nuevo significado o mensaje al código duplicado por Google.

Por otro lado, Oracle sostuvo que el uso de Google no era transformativo porque no altera las API con una nueva expresión, significado o mensaje.

Basándose Oracle en que Google había reconocido en el juicio que el uso de Google de los paquetes de las API de la librería de software estándar de Java tenía el mismo propósito que el de Oracle, Oracle sostenía que era irrazonable que se pudiera entender por el Jurado o por el Juez de Distrito, que Google había producido una "transformación" de las API de tal forma que se pudiera superar el alto grado comercial que poseía el uso por parte de Google.

Por su parte, Google sostenía que, aunque que el código declarado y su estructura, secuencia y organización pudieran ejecutar la misma función en Android que en Java, si podía entenderse que podría existir un propósito diferente, y por ende transformativo, teniendo en cuenta que el objetivo de Android fue crear una plataforma innovadora para los teléfonos móviles inteligentes.

El Tribunal de Apelaciones del Circuito Federal entendió que los argumentos brindados por Google no eran válidos, por lo tanto, el uso de los paquetes de las API de Java por parte de Google no había sido transformativo por las siguientes razones:

(i) no se encontraba listado dentro de la enumeración propuesta por el artículo 107,

(ii) el propósito de los paquetes de las API en Android era el mismo que el propósito de los paquetes en la plataforma Java,

(iii) Google no había efectuado una alteración de la expresión del contenido o del mensaje del material protegido de Oracle,

(iv) los teléfonos inteligentes no eran un nuevo contexto, haciéndose referencia a que ya se había implementado las API de Java por Oracle en el producto *Sava-Je* para teléfonos móviles.

En relación al punto (i) el Tribunal de Apelaciones del Circuito Federal argumenta que en su análisis buscó dentro de los ejemplos listados por el articulo 107 (criticas, comentarios, etc.) y el uso de Google de las API de Java no se encuentra listado dentro de los ejemplos de la norma, pero se agrega, que Google tampoco menciona o sugirió otros.

Agrega el Tribunal de Apelaciones del Circuito Federal, que Google sólo cito el caso *Sony Computer Entertainment, Inc v. Connectix Corp,* donde el mismo Circuito Noveno había manifestado que “otros tipos de uso” relacionados al uso de código de programas de computación constituían uso justo.

Este caso, que ya fue mencionado en un capítulo anterior, estableció que la ingeniería inversa y el código intermedio copiado por *Connectix* constituía uso justo o legítimo porque su propósito fue el de obtener acceso -y la decompilación fue la única manera- a material no protegible del software propiedad de *Sony* con el propósito de crear un nuevo programa de computación para permitir a los consumidores jugar videos juegos -diseñados para la consola de *Sony*- en sus computadoras.

En dicho caso judicial, se entendió que había existido un uso "transformativo modesto", en donde *Connectix* había creado un nuevo producto con un nuevo código y que la copia intermedia fue utilizada para producir un producto que fuera compatible (con el producto de *Sony*).

Al respecto dice el Tribunal de Apelaciones del Circuito Federal, tomando los argumentos de Oracle, que ese nivel "modesto" de uso transformativo es más transformativo que del uso que hizo Google, que fue el de copiar exactamente el código fuente declarado para atraer a los programadores a la nueva plataforma incompatible de Google.

De hecho, y probado en juicio está que Google concedió que las declaraciones de código y la estructura secuencia y organización de ello fue para beneficio de los programadores, quienes familiarizados con el lenguaje de programación Java, tenían expectativas relacionados a las API.

Por lo tanto, el hecho de que Google haya duplicado el código declarado y su estructura, secuencia y organización, y utilizado ello, con el mismo propósito que el material original de Oracle seriamente debilita la posición de que el uso de Google pueda ser considerado como uso justo o legítimo.

A su vez argumentar que el uso es transformativo porque se seleccionó y utilizó sólo el código fuente declarado y la estructura, secuencia y organización de 37 paquetes de los 166 de la plataforma Java SE, y que luego Google escribiera su propio código de implementación, no es un argumento válido, ya que tomar y seleccionar sólo partes de la obra protegida en sí mismo no es transformativo.

En realidad, el volumen o cantidad de material copiado es relevante para el análisis del uso justo en general, pero se debe pensar también, en la calidad y la importancia del material copiado, y no sólo en su cantidad relativa con respecto a la totalidad de la obra en general.

En el caso, sostiene el Tribunal de Apelaciones del Circuito Federal que lo copiado por Google es *cualitativamente* significativo.

Por último, el principal argumento de Google es que Android es transformativo porque los 37 paquetes de las API de Java y su estructura, secuencia y organización fueron incorporados a un nuevo contexto, el de los teléfonos móviles inteligentes.

Pero dice el Tribunal de Apelaciones del Circuito Federal, conforme a las constancias del juicio, que las API de Java SE estuvieron en los teléfonos móviles inteligentes antes de que Android entre en el mercado.

Específicamente surge del litigio que Oracle presentó pruebas en relación a que la plataforma Java SE estuvo en el teléfono móvil *SavaJe* y que Oracle licenció Java SE a fabricantes de teléfonos móviles como Nokia y Danger.

Por lo tanto, como Java SE ya se encontraba en los teléfonos móviles inteligentes, Google no transformó el material protegido en un nuevo contexto.

Por lo expuesto anteriormente es que el Tribunal de Apelaciones del Circuito Federal consideró que la *copia idéntica, para una idéntica función y propósito, sin cambios en la expresión de su contenido o mensaje, y un mero cambio en su formato* (de computadoras de escritorios a tabletas y teléfonos móviles inteligentes) es insuficiente para concluir que existe una transformación.

Por otro lado, lo contrario, podría implicar una invasión a los derechos del autor de crear obras derivadas basados en su propia obra.

Por último, en la evaluación del factor de propósito y el carácter del uso, el Circuito Noveno (y no todos los Circuitos[107]) aplica la regla general que la parte que reclama la aplicación de la excepción de uso justo debe actuar de una manera compatible con los *principios de buena fe*.

De todas formas, mientras que la mala fe juega en contra del uso justo, la buena fe del copista no juega a favor del uso justo.

Así la cosas a solicitud de Oracle, la Corte de Distrito instruyó al Jurado que como parte de la revisión del primer factor de uso justo considerase si Google había actuado con mala fe o no.

A su vez, como a Oracle se le permitió aportar evidencia que Google había actuado de mala fe, la Corte de Distrito permitió que Google aportara evidencia para probar su buena fe.

Entonces, Oracle aportó prueba sugiriendo que Google tenía la "necesidad de copiar las API de Java como un acelerador" para

[107] Se afirma que este criterio se encuentra basado en la decisión de la Corte Suprema de los EE.UU., que en el caso *Harper & Row*, SCOTUS dijo que el uso justo presupone *"good faith"* and *"fair dealing"*.

introducir a Android en el mercado, pero sabiendo que necesitaba una licencia de Oracle para utilizar Java SE.

Por otro lado, Google presentó pruebas en relación que el código declarado y la estructura, secuencia y organización "*era libre de usarse y reimplementarse*" desde el punto de vista de la práctica de los desarrolladores, y además porque la "disponibilidad de las implementaciones independientes de las API de Java", contribuían a la popularidad del lenguaje de programación de Java, las que Sun Microsystems había promocionado como "libres para ser usados por todos".

Teniendo en cuenta la prueba aportada por ambas partes, el Juez de Distrito consideró que podría ser considerado que la utilización de las API de Java por parte de Google como acelerador podría estar basado en la buena fe de creerse que el código declarado y la estructura, secuencia y organización eran libres de ser usadas (las cuales Google utilizó) y que una licencia sólo era necesaria para el código implementado (que Google no utilizó, sino que creó su propia implementación).

El Tribunal de Apelaciones del Circuito Federal, asumió que el Jurado no había encontrado evidencia que lo persuadiera de la mala fe de Google, ello porque la prueba aportada por Google mostrando su buena fe fue relevante para refutar la prueba para mostrar su mala fe.

Por lo cual, teniendo en cuenta el uso comercial y no transformativo realizado por Google, se concluyó al respecto del primer factor que jugaba en contra de otorgarse una excepción por uso justo o legítimo.

SEGUNDO FACTOR. NATURALEZA DE LA OBRA PROTEGIDA

Este segundo factor se relaciona con el reconocimiento de que algunas obras protegidas por el derecho de autor de EE.UU. están más cerca que otras obras del "núcleo de la protección conferida por el derecho de autor", con la consecuencia de que el uso justo es más difícil de establecer cuando se trata de la copia del tipo de las últimas obras, es decir de aquellas que se encuentran más alejadas de ese núcleo de protección.

Así entonces, este factor se trata de si la obra es de carácter "informativo o creativo".

Se ha reconocido por el Circuito Noveno y otros Circuitos, que este segundo factor no es típicamente significativo en el análisis total del balance del uso justo.

El Tribunal de Apelaciones del Circuito Federal en su sentencia de mayo de 2014 reconoció que el código declarado y la estructura, secuencia y organización de los 37 paquetes de las API de Java eran suficientemente creativas para calificarlas como protegidas bajo la ley de derechos de autor.

Ahora bien, también dijo que los aspectos funcionales de los 37 paquetes podrían llegar a ser relevantes para la defensa de uso justo o legítimo de Google.

Así las cosas, Oracle mencionó que al menos tres paquetes de Java eran necesarios para utilizar el lenguaje de programación Java, pero no presentó pruebas explicando de cómo se distinguían la funcionalidad y la creatividad de esas declaraciones respecto del resto de los paquetes de las API de Java.

Google por su parte menciona que conforme a la prueba rendida en el juicio las declaraciones y la estructura, secuencia y organización son funcionales.

Entonces, si bien es cierto que las API de los 37 paquetes de Java poseen un cierto grado de creatividad, no es menos cierto que las consideraciones de funcionalidad son importantes y substanciales.

Por lo cual, basándose en ello, se dispuso por el Tribunal de Apelaciones del Circuito Federal que el segundo factor es en favor de conceder la excepción del uso justo a favor de Google.

TERCER FACTOR. CANTIDAD Y SUSTANCIALIDAD DE LA PARTE UTILIZADA EN RELACIÓN A LA OBRA PROTEGIDA POR DERECHOS DE AUTOR EN SU CONJUNTO

El tercer factor se focaliza en la "cantidad y sustancialidad de la porción utilizada de la obra protegida" (Java SE), no de la obra en infracción (Android).

El lenguaje del artículo 107 deja claro que lo tomado o utilizado no podría excusarse por el simple hecho de que es insustancial respecto de la obra en infracción.

Y por su parte el hecho de que una parte sustancial de la obra en infracción haya sido copiada en forma idéntica de la obra originaria es evidencia del valor cualitativo del material copiado.

Con lo cual, si bien la copia total no excluye el uso justo *per se*, si milita contra el hallazgo o existencia de uso justo o legítimo.

A su vez, este factor es flexible y no es solamente una determinación de porcentaje de la obra protegida utilizado.

En realidad, se trata de buscar que monto y valor cualitativo de la obra original fue utilizada en relación a la justificación de su uso.

Entonces, el porcentaje del material copiado "no es dispositivo" cuando la porción copiada es cualitativamente significativa.

Menciona el Tribunal de Apelaciones del Circuito Federal que es cierto que el tercer factor no podría ir en contra del supuesto infractor, incluso, aun cuando se haya copiado la totalidad de la obra, *si se toma sólo aquello o no más que lo necesario para su uso previsto.*

Ahora, se afirma que ello es así cuando en realidad el uso previsto es transformativo.

En las presentes actuaciones judiciales, se encontró por el Tribunal de Apelaciones del Circuito Federal que el uso de Google no era transformativo, y de hecho Google concedió en el trámite del juicio que podría haber escrito sus propias API y *que el propósito de su copia fue hacer a la plataforma Android atractiva para los programadores*.

En relación a este factor, se explicó por el tribunal inferior (Juez William Alsup) que podría considerarse razonable que la copia del mínimo de los 37 paquetes de las API y su estructura, secuencia y organización por parte de Google fue necesario para preservar la consistencia en el uso entre las plataformas Java y Android, es decir copiar el código declarado y la estructura, secuencia y organización, y no así el código fuente implementado por Oracle, por lo tanto, que lo copiado no era más que lo razonable o necesario.

En tomar esta conclusión, se mencionó por el Juez de Distrito que el número de líneas de código copiadas de Oracle era una

pequeña fracción del uno por ciento de la obra protegida (Java SE) (y aún menor respecto de Android).

El Tribunal de Apelaciones del Circuito Federal manifiesta que ello es un error, por cuanto en realidad las partes estipularon que únicamente *170 líneas de código* eran necesarias para escribir en el lenguaje Java, y sin embargo está probado en la presente actuación judicial que Google copió 11.500 líneas de código fuente declarado, es decir, 11.330 líneas más de código que las necesarias para escribir en Java.

Si bien Google argumenta que copió una pequeña parte de Java, unas 11.500 líneas de código fuente declarado de casi un total de 2.86 millones de líneas de código de la librería estándar de la plataforma Java SE, no es menos cierto que Google copió la totalidad de la estructura, secuencia y organización de los 37 paquetes de las API de Java.

Se afirma por el tribunal de alzada, que también se enfatiza por el Juez de Distrito que el deseo de Google era preservar la consistencia entre los sistemas, es decir evitar la confusión a los programadores de Java respecto del sistema de Java y el sistema de Android.

Ahora bien, resulta probado en las actuaciones que Google no buscó la consistencia entre ambas plataformas, y *de hecho en su recurso de apelación Google no menciona ningún argumento relacionado a la interoperabilidad*.

A su vez, si bien varios *Amicus curiae* (amigos del tribunal) de Google en la etapa de apelación mencionan el asunto de la *interoperabilidad*, el Tribunal de Apelaciones del Circuito Federal mantiene que Google *ha abandonado sus argumentos de interoperabilidad en la etapa de apelación*.

Dice el Tribunal de Apelaciones del Circuito Federal, que este cambio no le sorprende ya que existe una cantidad suficiente de prueba irrefutable que Google específicamente diseño Android para que no sea compatible con la plataforma Java, y por ende para no permitir la interoperabilidad de los programas con Java.

En realidad, Google buscó capitalizar sobre la circunstancia de que los desarrolladores de software ya estaban entrenados y poseían experiencia en la utilización de los paquetes que conforman la librería estándar de Java.

Ahora bien, no existe ningún derecho inherente de copiar con el propósito de capitalizar la popularidad de una obra protegida.

Es decir, tomar material de una obra protegida que fuera familiar a los desarrolladores, para crear obras similares diseñadas para ser populares para esos mismos desarrolladores, no constituye uso justo.

Entonces, aun asumiéndose de que lo copiado fuera una pequeña cantidad, no es razonable pensar que el material copiado es cualitativamente insignificante, particularmente cuando el material copiado era importante para crear la plataforma Android.

Se agrega, según propias palabras de Google *"fue una buena práctica comercial aprovechar la comunidad existente de desarrolladores, minimizando el monto de nuevo material creado y maximizando el conocimiento existente"*

Por ello, concluye el Tribunal de Apelaciones del Circuito Federal que, en el mejor de los casos este factor es neutral o favorece a Oracle.

CUARTO FACTOR. EL EFECTO DEL USO SOBRE EL MERCADO POTENCIAL O EL VALOR DE LA OBRA PROTEGIDA POR DERECHOS DE AUTOR

Este cuarto factor refleja la idea de que el uso justo se limita a que la copia realizada por terceros materialmente no perjudique la comerciabilidad de la obra copiada.

Este factor requiere que los tribunales no sólo consideren la extensión del daño causado en el mercado por acciones particulares del presunto infractor, sino también, si de la conducta irrestricta y generalizada que lleva a cabo el acusado podría resultar un impacto sustancialmente adverso en el mercado potencial para la obra original.

La Corte Suprema de EE.UU. ha reconocido en el caso *Harper & Row* que el cuarto factor *es indudablemente el elemento individual más importante del uso justo*.

Y en un caso posterior a *Harper*, en *Campbell*, la Corte de los EE.UU. enfatizó que ninguno de los cuatro factores puede ser mirado individualmente, que todos deben ser explorados, y que su resultado debe ser analizado conjuntamente, todo ello a la luz del propósito de la ley de derechos de autor de EE.UU.

También se ha expresado por la Corte Suprema que el daño en el mercado es una cuestión de grados, y que la importancia de este factor podría variar, no sólo por el volumen del daño sino por la fuerza que posean el resto de los otros tres factores del uso justo.

Por otro parte, en el Noveno Circuito se ha indicado que donde el "uso es comercial y no transformativo" existe una presunción que indica un daño en el mercado (*Disney Enters., Inc v. VidAngel, Inc- 9no Circuito, año 2017*).

Este caso judicial, toma en realidad una doctrina de un caso anterior *Sony Corp. Of America v. University City Studios, Inc (1984)* en donde la Corte Suprema de EE.UU. mencionó que cuando el uso es comercial la probabilidad de daño futuro puede ser *presumida*, pero si el propósito del uso no es comercial la probabilidad de daño debe ser *probada*.

Después del caso *Sony*, la Corte Suprema de EE.UU. clarificó que el impacto en el mercado debe ser analizado al igual que el resto de los factores teniéndose presente una "sensibilidad en el equilibrio de intereses", y que la interpretación de *Sony* no era correcta.

Por esta razón, explica el Tribunal de Apelaciones del Circuito Federal, en el caso particular se ve obligado a seguir el criterio de la Corte Suprema de EE.UU. que se encuentra por encima de la doctrina del Circuito Noveno del año 2017 del caso *Disney*.

Entonces, en la evaluación del cuarto factor, los jueces deben considerar no sólo el daño actual o potencial del mercado para la obra protegida, sino también el daño del *"mercado para los potenciales usos derivados*", incluyéndose aquellos que los creadores de la obra protegida generalmente desarrollarían o licenciarían a otros para desarrollarlos.

Por lo que se debe analizar entonces el impacto del uso en los ingresos por potenciales licencias para mercados tradicionales, razonables o con probabilidades de ser desarrollados.

Téngase presente que un titular de derechos posee facultades exclusivas para determinar ¿cuándo? ¿cómo? y ¿de qué forma? lanzar el material protegido a nuevos mercados, sea por el mismo titular de derechos o mediante licenciamiento a terceros.

Así en el Noveno Circuito se ha reconocido que aún en el caso que un autor haya desautorizado cualquier intención de publicar su obra dentro del periodo de protección, tiene derecho a la protección bajo la ley de derechos de autor, ya que la consideración relevante es el mercado potencial, y, en segundo lugar, tiene derecho a cambiar de opinión.

En definitiva, solo el titular de derechos de autor tiene la facultad de decidir de entrar a un mercado o no.

En el caso particular, la Corte de Distrito expresó que podría interpretarse que el uso del código declarado y la estructura, secuencia y organización en Android no causaba daño en el mercado para la obra protegida (Java SE), teniendo en cuenta que su mercado, el de Java SE, era el de las computadoras de escritorios y las portátiles.

Para llegar a esta conclusión el Juez de Distrito mencionó que antes de que Android fuera publicado y comercializado, Sun Microsystems había hecho disponible todos los paquetes de las API de Java bajo una licencia de código abierto, la licencia GPLv2 o Licencia Pública General, versión 2.

Dicha implementación de las API de Java de código abierto de Sun Microsystems llevo el nombre de “OpenJDK”.

Entonces, el Juez de Distrito interpretó que se podría concluir que el impacto de Android en el mercado de la obra protegida, podría ser lo mismo que lo que Sun Microsystems esperaba que fuera a través del proyecto de OpenJDK.

Expresa el Tribunal de Apelaciones del Circuito Federal que conforme surge de las constancias del juicio con relación al daño en el mercado actual, Java SE había sido usado por años en dispositivos móviles, incluyéndose a los primeros teléfonos inteligentes, todo ello antes que Android.

Puntualmente existen constancias en el pleito que incluía Java SE en los teléfonos inteligentes de Nokia, Danger, SavaJe y Blackberry.

Por lo tanto, que Android haya competido directamente con Java SE en el mercado de teléfonos móviles es suficiente para rechazar los argumentos de Google.

Con relación a las "tabletas", la prueba rendida en el juicio demuestra que Oracle licenciaba Java SE en Amazon Kindle.

Después de la publicación de Android, la empresa Amazon tenía dos opciones que competían entre sí, la de Oracle con Java SE (comercial con pago de royalties), y la de Google con Android (open source, con licencia Apache v2).

Amazon escogió Android.

Es decir, Amazon no escogió la versión *open source* de Oracle OpenJDK, sino Android.

De la prueba rendida en el juicio también surge que los fabricantes de dispositivos no veían a OpenJDK como una opción comercialmente viable, teniendo en cuenta los términos de su licencia GPLv2, es decir, que cualquier modificación en los paquetes de Java, al distribuirse el producto, gatillaba la obligación de compartir esas modificaciones.

Por ello, la prueba en el juicio demuestra que Android *fue usado como un sustituto para* Java SE y eso tuvo un impacto directo en el mercado.

Agrega el Tribunal de Apelaciones del Circuito Federal que aún para el caso de que no estuviera en discusión que Oracle hubiese licenciado Java SE en los teléfonos móviles antes de que Android fuera lanzado, el punto es que el "uso justo" se focaliza

no sólo en el daño en un “mercado actual”, sino también en el daño en un “mercado potencial”.

Entonces el tribunal inferior ha concentrado su teoría sólo en el mercado actual -computadoras de escritorios y portátiles-, es decir en el mercado en que Oracle ya había entrado, cuando en realidad debió haber considerado:

(i) ¿cómo la copia de Google podía haber afectado los potenciales mercados?
(ii) que Oracle podría entrar en un futuro [en tal mercado], o incluso,
(iii) las obras derivadas que Oracle podría haber creado, o
(iv) licenciado o autorizado a otros para que las creen (licenciatarios de Oracle).

Por ejemplo, licenciar Java SE para teléfonos móviles que incrementen la capacidad de procesamiento era uno de los potenciales mercados.

A su vez, el hecho de que Oracle y Google tuvieran largas negociaciones en materia de licenciamiento de Java SE, demuestra que la intención de Oracle era licenciar Java SE para los dispositivos móviles incluyendo a la nueva generación de teléfonos inteligentes.

Entonces, los “teléfonos inteligentes” constituye un mercado tradicional, razonable o probable a ser desarrollado.

Google sostiene que Java SE y Android no compitieron en el mismo mercado porque Oracle no era un fabricante de dispositivos, y, además no había construido aún su propia plataforma para teléfonos inteligentes.

El Tribunal de Apelaciones del Circuito Federal rechazó estos argumentos sobre la base de la circunstancia de que el hecho que

Oracle nunca haya construido un dispositivo móvil es irrelevante porque la definición de "mercado potencial" incluye licenciar a otros para que desarrollen productos derivados.

La misma suerte corre el hecho de que Oracle no haya desarrollado una plataforma para teléfonos móviles, ya que un mercado es un "mercado potencial" aún, cuando el titular de derechos no tiene planes inmediatos para entrar en él o no haya tenido éxito para entrar en él.

De nuevo, el Tribunal de Apelaciones del Circuito Federal menciona que la *prueba indiscutida* de la causa muestra como mínimo que Oracle intentó licenciar Java SE para los teléfonos móviles inteligentes, y no hay ninguna prueba en el juicio que muestre y que pueda soportar una posición contraria a ello.

Por las razones expuestas el Tribunal de Apelaciones del Circuito Federal entendió que teniendo en cuenta las constancias de la causa que demuestran un actual y potencial daño, concluye que la conducta de Google resultaría en un impacto adverso sustancial para el mercado potencial para la obra original y sus obras derivadas, por lo que de acuerdo con ello sostuvo que el cuarto factor se encuentra a favor de Oracle.

EL BALANCE DE LOS CUATRO FACTORES DEL USO JUSTO

Habiendo realizado el Tribunal de Apelaciones del Circuito Federal un estudio individual de los cuatro factores, concluyó que permitir que Google explotara comercialmente la obra protegida de Oracle *no contribuía con el avance de los propósitos del sistema de derecho de autor de los EE.UU.*

No hay nada justo o legítimo en tomar una obra protegida, copiarla exacta y usarla para los mismos propósitos y función que la obra original en una plataforma en competencia.

Nuevamente, aun ignorándose la prueba arrimada al juicio y asumiendo que Oracle no había licenciado Java SE en el contexto de los teléfonos móviles inteligentes, sin duda los teléfonos móviles inteligentes constituyen un "mercado potencial".

Por ello, el Tribunal de Apelaciones del Circuito Federal expresa que los factores *uno* y *cuatro* están en contra del uso justo, el factor *dos* a favor del uso justo, y el factor *tres*, en el mejor de los casos, es neutral.

Entonces sopesado todos los factores en su conjunto, se concluye que el uso de Google del código declarado de los 37 paquetes de las API de Java y de su estructura, secuencia y organización *no constituyen uso justo conforme a lo establecido en el artículo 107 del Título 17, USC.*

CAPITULO XI

RECURSO DE APELACIÓN (PETICIÓN II *WRIT OF CERTIORARI)* ANTE LA CORTE SUPREMA DE JUSTICIA DE EE.UU. INTERPUESTO POR GOOGLE CONTRA LA SENTENCIA II DEL TRIBUNAL DE APELACIONES DEL CIRCUITO FEDERAL

Google apeló la sentencia del Tribunal de Apelaciones del Circuito Federal presentado su Petición II de *Write of Certiorari* con fecha 24 de enero de 2019 ante la Corte Suprema de los EE.UU., la cual fue respondida por Oracle con fecha 27 de marzo de 2019 y posteriormente, nuevamente Google con fecha 10 de abril de 2019 respondió las respuestas de Oracle.

Por otra parte, la Corte Suprema de Justicia solicitó la opinión del Abogado General de los Estados Unidos, y con fecha 27 de septiembre de 2019, el Abogado General de EE.UU., Noel J. Francisco, emitió opinión expresando a la Corte Suprema de Justicia que en su visión la petición presentada por Google debía ser denegada[108].

Posteriormente el 15 de noviembre de 2019 la Corte Suprema de Justicia decidió aceptar la Petición II de *Writ of Certiorari* presentada por Google.

Concedido el mismo, Google abrió el debate presentando sus fundamentos del *Writ of Certiorari* con fecha 6 de enero de 2020[109].

---

[108] Véase en https://www.supremecourt.gov/DocketPDF/18/18-956/117359/20190927165110897_18-956%20Google.pdf

[109] Véase en https://www.supremecourt.gov/DocketPDF/18/18-956/81532/20190124110509177_Google%20cert%20petition.pdf

Oracle hizo lo suyo con fecha 12 de febrero de 2020[110] solicitando se confirme la sentencia del Tribunal de Apelaciones del Circuito Federal.

Posteriormente a ello, fueron presentadas diferentes opiniones de los *Amicus Curiae* (amigos del tribunal), y en octubre de 2020 se escucharon los argumentos orales de las partes ante los jueces de la Corte Suprema de Justicia de EE.UU.

De esta forma la causa quedó en orden para dictarse sentencia por la Corte Suprema de Justicia de EE.UU.

ARGUMENTOS DE GOOGLE CONTRA LA SENTENCIA II DEL TRIBUNAL DE APELACIONES DEL CIRCUITO FEDERAL

En su Petición II de *Writ of Certiorari* Google expresa a la Corte Suprema que el presente caso presenta dos cuestiones:

(i) la primera, concerniente a la protección por derechos de autor, y
(ii) la segunda, relacionada al uso justo de las *interfaces de software*.

La primera pregunta que Google realiza es si la protección de los derechos de autor de EE.UU. se extiende a las *interfaces de software*, y, en segundo lugar, si el uso de esas interfaces de software[111] por Google en un nuevo contexto de creación de un

[110] Véase en https://www.supremecourt.gov/DocketPDF/18/18956/132891/20200212180251262_200208a%20Resp%20Brief%20for%20efiling.pdf

[111] Véase en https://www.supremecourt.gov/docket/docketfiles/html/qp/18-00956qp.pdf

programa de computación (plataforma Android) constituye uso justo.

Menciona Google que Sun Microsystems originariamente desarrolló la plataforma Java la cual incluye el lenguaje de programación Java, y que las interfaces de software en cuestión son parte de las API del lenguaje de programación Java.

Así, Sun Microsystems alentó a los desarrolladores de Java a aprender el lenguaje de programación Java promocionando el uso de las interfaces de software o declaraciones de las API de Java para acceder al código pre-escrito de la librería estándar de Java para así llevar a cabo determinadas funciones.

Las interfaces de software, por lo tanto, facilitaron el desarrollo de programas en el lenguaje de programación Java.

Google equipara la relación entre el código declarado y el código implementado con la interacción entre un teclado y un programa de procesador de texto.

Cuando se desea escribir y se tipea una letra en particular se causa que la letra tipeada se escriba en un programa de procesador de texto.

Entonces de igual forma el desarrollador invoca una función particular mediante el uso del código declarado para hacer correr el código implementado.

Entonces, permitiéndole a los programadores el acceso a la librería de funciones pre-escritas de una manera estándar, las API de Java facilitan la creación de programas en el lenguaje Java para diferentes plataformas, así como lo hace el estándar de teclado QWERTY que facilita la creación de documentos permitiendo la eficiencia del tipeado sin importar el programa de procesador de texto que se esté utilizando.

Con lo cual lo único que un desarrollador necesita hacer para invocar un método de Java es usar los *comandos abreviados* (en inglés "*shorthand commands"*) que derivan del mismo código fuente declarado del método de Java.

Google entendió que los desarrolladores de aplicaciones querrían utilizar todo el conocimiento previo obtenido al respecto del lenguaje de programación Java al desarrollar nuevas aplicaciones en la plataforma Android en lenguaje Java.

Entonces, ello implicaba "todo el conocimiento adquirido" en relación al *código fuente declarado* de los métodos de Java (API) y a los *comandos abreviados* respectivos para invocarlos.

Por lo tanto, para que estos *comandos abreviados* conocidos por los programadores de Java en oportunidad de desarrollar programas en la plataforma Java, pudieran ser aprovechados por ellos en el desarrollo de aplicaciones en la plataforma Android, Google tuvo que replicar exactamente *la sintáxis del código declarado y la estructura de las API*.

Cualquier cambio, o modificación que Google hubiese realizado en el código fuente declarado, hubiera tenido como consecuencia directa que los desarrolladores se hubiesen visto impedido de poder utilizar esos *"comandos abreviados"*, por ende, esto los hubiera forzado a aprender *nuevos comandos abreviados* para cada una de las funciones que conocían.

Es decir, si Google hubiera utilizado diferentes interfaces (código fuente declarado) en Android, los desarrolladores tendrían que haber aprendido las nuevas interfaces o código declarado para operar los métodos pre-escritos que ellos ya conocían del lenguaje de programación Java.

Por tanto, Google utilizó las mismas declaraciones para ciertos métodos de los 37 paquetes de las API de Java.

Por otro lado, para cada uno de esos métodos, Google escribió su propio código fuente implementado adaptándolo para acomodarlo a los desafíos del nuevo entorno de teléfonos móviles inteligentes.

De lo contrario, los desarrolladores habrían quedado atrapados dentro de la plataforma Java (la cual es controlada por Oracle) y además hubieran sido impedidos de reutilizar su propio código fuente, o del conocimiento familiar de las interfaces y comandos abreviados, en la plataforma Android o en cualquier otra plataforma de desarrollo.

A su vez, prohibir a Google el uso de las declaraciones o interfaces de las API de Java permitiría a Oracle acumular poder de mercado a través del derecho de autor, bloqueando a los desarrolladores que han invertido en aprender el lenguaje de programación Java y haciendo a ellos difícil usar sus cualidades para programar en nuevas plataformas.

Por el contrario, que Oracle posea el control significaría el bloqueo de la competencia entre plataformas de desarrollo al acceso por parte de los desarrolladores entrenados en el lenguaje de programación Java.

Teniendo en cuenta que las interfaces son centrales para que los desarrolladores puedan operar entre programas, ejercer el control sobre las interfaces, *hace nacer barreras que atentan contra la competencia e innovación*.

Por su parte Google menciona que el Tribunal de Apelaciones del Circuito Federal ha cometido serios errores en su análisis de uso justo, y por ello solicita a la Corte Suprema reversar el fallo.

Manifiesta, que el error más grave que ha cometido el Tribunal de Apelaciones del Circuito Federal al realizar el análisis del “uso transformativo” fue centrar su análisis

únicamente en el material que Google había reusado (las declaraciones de las API de Java) y evaluar si el material en sí mismo se transformaba en la nueva obra (Android).

En su lugar habría que haberse focalizado en la "nueva obra" *como un todo*, preguntándose si la nueva obra agrega algo nuevo -a la obra protegida- con nuevos propósitos o diferente carácter.

La nueva obra creada por Google (Android) y el uso de las declaraciones de las API de Java son transformativas.

Google creó una nueva plataforma para teléfonos móviles inteligentes.

Por el contrario, Java SE fue diseñada para computadoras de escritorios y servidores.

La nueva plataforma Android tuvo que acomodar ciertas limitaciones, como capacidad de memoria y vida útil de la batería, que no eran aplicables a Java SE.

A su vez, como la plataforma Android debía ser desarrollada para el entorno de los teléfonos móviles inteligentes, Google tuvo que desarrollar todo su código implementado para Android, incluyendo el código implementado para las declaraciones de las API de Java.

Por ello, Google creó una nueva librería de software para Android, incluyendo nuevas declaraciones y código implementado para las funciones necesarias para operar en los teléfonos móviles inteligentes, en pantallas táctiles, *web-browsing* y *location awareness.*

Por otro lado, en relación al análisis del cuarto factor, el Tribunal de Apelaciones del Circuito Federal concluyó que el uso de las declaraciones de las API de Java por parte Google causaban daño al mercado actual de Oracle, ya que Java SE había

sido usado supuestamente en el mercado de telefonía móvil antes del nacimiento de Android.

Incluso dijo el Tribunal de Apelaciones del Circuito Federal, que, aunque no hubiera habido daño actual existía de cualquier manera daño potencial al mercado de Oracle porque los teléfonos móviles inteligentes constituían un mercado tradicional, razonable, o probable para ser desarrollo para Java SE.

Google afirma que el Tribunal de Apelaciones del Circuito Federal llego a dicha conclusión porque reviso y reverso en forma incorrecta la decisión de determinación de hechos implícitos del Jurado.

Por ello su decisión respecto de este factor es incorrecta.

CAPITULO XII

ARGUMENTOS DE OPOSICIÓN DE ORACLE A LA PETICIÓN DE CONCESIÓN DEL *WRIT OF CERTIORARI II* PRESENTADO POR GOOGLE

Habiéndosele corrido traslado a Oracle de los argumentos de la Petición II de "*Writ of Certiorari*" presentado por Google, Oracle con fecha 27 de marzo de 2019 presentó su escrito de oposición.

Oracle manifiesta que la primera mitad de la Petición II de *Writ of Certiorari* de Google, ya ha sido rechazada por la Corte Suprema de EE.UU.

Respecto de la segunda mitad Oracle menciona que no existe conflicto de interpretación entre los distintos Circuitos de Apelaciones como expresa Google, en relación a la interpretación del artículo 102 del Título 17 USC.

En forma previa en este mismo caso Google presentó una Petición I de *Writ of Certiorari*[112] en relación a si el código fuente de los programas de computación propiedad de Oracle se encontraban protegido bajo las leyes de derechos de autor.

La Corte Suprema en el año 2015 invitó al Procurador General de los EE.UU. a dar su opinión al respecto, y éste dijo que el código de Oracle estaba protegido por las leyes de derechos de autor, además de mencionar que el argumento de Google en relación a la división de interpretación entre los Circuitos no tenía mérito alguno, por lo tanto, recomendó a la Corte Suprema no revisar el caso.

[112] Con fecha 6 de octubre de 2014

Posteriormente la Corte Suprema denegó la Petición I de *Writ of Certiorari* de Google.

Google, ahora intenta hacer la misma pregunta a la Corte Suprema dándole los mismos argumentos.

Pero, Google no ha identificado que algo haya cambiado.

Por lo tanto, sostiene Oracle que la Petición II de *Writ of Certiorari* de Google debe ser rechazada.

Teniendo en cuenta ello, Oracle menciona que las preguntas que la Corte Suprema debería responder son las siguientes:

> 1) ¿La ley de derechos de autor de EE.UU. protege el código fuente de los programas de computación de Oracle que Google admitió que eran originales y creativos, y que Oracle podría haberlos escritos en una gran cantidad de maneras diferentes para ejecutar las mismas funciones?
>
> 2) ¿Fue correcta la decisión del Tribunal de Apelaciones del Circuito Federal en cuanto sostuvo que:
>
> (a) no constituía uso justo o legítimo como una cuestión de derecho (en inglés "*as a matter of law"*) que Google copiara el código fuente de Oracle en una plataforma comercial (Android) en competencia con la plataforma Java con el propósito de atraer a los programadores de Java, y que,
>
> (b) Google podría haberla escrito sin copiarla de Sun Microsystems/Oracle, y que,
>
> (c) la conducta de Google al copiar causó daños sustanciales en el mercado actual y potencial para las obras protegidas de Oracle?

Oracle sostiene que la decisión del Tribunal de Apelaciones del Circuito Federal fue correcta porque aplicó la doctrina del derecho de autor establecida y acordada en los EE.UU.

Oracle pasó años desarrollando Java y gasto cientos de millones de dólares desarrollando una obra exitosa como la plataforma Java.

Google rechazó el ofrecimiento de licenciamiento ofrecido por Oracle, y en su lugar, copió sin autorización de Oracle las porciones de código más importantes de ella para introducirla en su plataforma Android en competencia con la plataforma Java con el propósito de atraer a los programadores aficionados, fanáticos o *fans* de la plataforma Java de Oracle.

Naturalmente ello, causó un daño incalculable en el mercado para Oracle.

Sin importar todo esto, Google menciona que la copia realizada debe ser permitida *para una cierta categoría de código fuente*, la cual describe Google como *"líneas de código de computadora que permite a los desarrolladores operar el código de la librería pre-escrita usado para ejecutar tareas particulares"*.

Por otro lado, Google confirmó en el juicio que a propósito construyó su plataforma Android para ser incompatible con la plataforma Java.

Por ello, no tiene sentido analizar en el caso las implicancias de la interoperabilidad.

Por otro lado, Google no ha citado durante todo el trámite del juicio ningún caso judicial en el cual alguna vez se haya sostenido que copiar semejante cantidad de líneas de código como su estructura, secuencia y organización en otra obra en competencia, pueda constituir uso justo.

Oracle creó y continúa desarrollando la plataforma Java 2 Edición Estándar o Java SE o plataforma Java.

La plataforma Java es una de las obras de computación más importantes y revolucionarias de la industria de software.

Java hace que sea fácil desarrollar y hacer correr aplicaciones conocidas o populares ("apps") escritas en el lenguaje de programación Java.

Dos características fueron las que contribuyeron a la popularidad de la plataforma Java.

La primera fue, que, a diferencia de otras plataformas, Java permitía a los programadores de aplicaciones escribir programas que pudieran correr en diferentes equipos de computación/sistemas operativos, sin tenerse la necesidad de rescribir los programas.

De ahí el principio *"write once, run anywhere".*

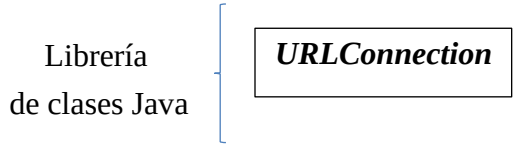
En segundo lugar, la plataforma Java contiene miles de programas pre-escritos (incluidos dentro de la librería estándar de Java), con lo cual los programadores pueden utilizar estos programas pre-escritos por Oracle con la finalidad de escribir sus propios programas en lugar de tener que escribir ellos mismos dichas funciones desde cero.

Por su parte, cada uno de estos programas posee un código fuente declarado y un código fuente implementado.

El código declarado es como el título de un capítulo y las oraciones temáticas de una obra literaria, así introduce nombres, y describe cada uno de los programas pre-escritos para ayudar a los programadores a aprender y recordar "¿qué es lo que hacen esos programas pre-escritos?"

Entonces:

a) programa de Oracle incluido dentro de la librería de software estándar de Java o librería de clases de Java (*Java Class*):



b) código fuente declarado por Oracle del programa *URLConnection*:

```
Public URLConnection openConnection() throws
java.io.IO.Exception
```

c) programador que escribiendo código en el lenguaje de programación Java quiere utilizar el programa pre-escrito de Oracle *URLConnection* incluido en la librería de software estándar de Java (y no desea escribir desde cero dicho código fuente con dicha funcionalidad) para que su propio programa desarrollado en lenguaje Java pueda conectarse a la red internet. Para ello, este programador debe invocar el código fuente declarado por Oracle, escribiendo "su propio código fuente" dentro de su programa, tipeando el comando abreviado establecido por Oracle:

```
New URL(´http//www.bankofamerica.com´).open
Connection().
```

d) Cuando el programa comienza a correr la plataforma Java reconoce el código declarado y así invoca el correspondiente

código fuente implementado para realizar la conexión con www.bankofamerica.com

Por otro lado, contrario a la analogía expuesta por Google, el código fuente declarado es mucho más expresivo que las letras de un "teclado".

En primer lugar, cada letra de teclado es un carácter singular, mientras que el código fuente declarado puede ser extremadamente extenso y expresivo:

Public abstract void verify (PublicKey key, String sigProvider) throws CertificateException,No-SuchAlgorithmException, InvalidkeyException,NosuchProviderexception,SignatureException

En segundo lugar, mientras que "ASDFG" no expresa absolutamente nada a nadie, el código fuente declarado de Oracle cumple con las siguientes premisas:

(i) comunicar a los programadores que es lo que hace cada programa,
(ii) establecer cómo se relacionan con otros programas, y,
(iii) establecer que es necesario hacer para que el programa funcione.

En tercer lugar, en relación a la "complejidad", no hay punto de comparación entre la relación entre las 26 teclas [de un teclado QWERTY] entre sí y la organización de la plataforma Java entre sus más de 30.000 programas.

En relación a las formas de *licenciamiento de Java*, es crucial mencionar una distinción -que Google omite mencionar en sus presentaciones- entre:

1) *"programadores de "apps"* quienes sólo utilizan la plataforma Java para crear o desarrollar sus aplicaciones. Estas aplicaciones corren en tabletas y teléfonos móviles inteligentes. Oracle ofrece a los programadores de aplicaciones una licencia sin costo alguno, y

2) los *"desarrolladores de plataformas"* quién copian la plataforma Java para comercializarla en sus propias plataformas.

Oracle recupera su inversión en la plataforma Java principalmente licenciando a:

1) *fabricantes de hardware o dispositivos* que quieren incorporar el programa de software de la plataforma Java en sus propios dispositivos (PC´s, tabletas, teléfonos) para correr la gran variedad y cantidad de aplicaciones desarrolladas por los programadores de aplicaciones,

2) *desarrollares de plataforma en competencia* que quieren utilizar los programas de Oracle para comercializarlos en sus propias plataformas. Entonces, cualquier desarrollador de plataforma que no quiera tomar una licencia de Oracle está en su derecho y libertad de desarrollar su propia plataforma con idénticas funciones sin copiar la plataforma Java. Así lo han hecho Apple y Microsoft.

Pero si una empresa que es un desarrollar de plataforma quiere copiar o adaptar la plataforma Java, Oracle exige y requiere que se lo haga tomando la licencia respectiva.

Y a su vez, si la empresa desarrolladora de plataforma quiere mantener sus modificaciones "propietarias", pues entonces debe

tomar una licencia comercial de Oracle, y pagar a Oracle las tarifas correspondientes.

Y, por otro lado, a estas empresas con licencias comerciales, para preservar el "*write once, run anywhere*" Oracle les exige que cumplan con los requerimientos de compatibilidad (TCK).

Por otro lado, la opción de código abierto denominada OpenJDK está disponible, y puede utilizarse todos y cada uno de los paquetes de las API de Java sujeto a las condiciones de la licencia GPLv2.

Bajo este esquema de licenciamiento, Java se convirtió en la plataforma líder para desarrollar y hacer correr aplicaciones en teléfonos móviles.

Antes de Android cada una de las compañías que quería utilizar la plataforma Java tomaba una licencia comercial.

Entre ellos estuvieron los fabricantes de teléfonos inteligentes como Blackberry, Nokia y Danger.

Por otro lado, compañías como IBM y Oracle (antes de adquirir a Sun Microsystems) que deseaban copiar sólo el código fuente declarado y la estructura secuencia y organización, también tomaron una licencia comercial.

Por otro lado, la primera pregunta presentada por Google a la Corte Suprema es si *¿la protección de derechos de autor se extiende a las interfaces de software?*

El término "interfaces de software" es un término inventado por Google para su Petición II de *Writ of Certiorari*, el cual no se encuentra definido en la ley de derechos de autor de EE.UU., en las constancias del juicio, en las opiniones del Tribunal de Apelaciones del Circuito Federal, y en ninguno de los casos citados por Google.

Es tan insignificante o sin sentido como preguntar si la protección de los derechos de autor se extiende a la “interface verbal”.

Para saber si el derecho de autor se extiende a una obra en particular, uno necesita ser preciso acerca de que obra se trata.

En el presente juicio la obra protegida en cuestión es de un conjunto intrincado de 37 paquetes de miles de programas de computación y que constituye uno de los programas de computación de plataformas más creativos y más populares que hayan sido escritos.

Así, Google copió 11.500 líneas de código fuente declarado, cada una consistente en palabras y símbolos que expresan y comunican cierta información al programador, como así también Google copió la organización de los programas de Oracle.

Entonces, la cuestión es sí tal obra ¿está totalmente despojada de la protección de los derechos de autor?

Como lo ha expuesto el Procurador General de los EE.UU. en el año 2015 en su opinión recomendado a la Corte Suprema la denegación de Petición I de *Writ of Certiorari* de Google, no hay nada en el código fuente declarado de Oracle en cuestión que lo haga materialmente diferente de otro tipo de código fuente de un programa de computación, y Google no ha identificado conflicto de autoridad acerca de la protección bajo derechos de autor de los programas de computación, de su código fuente o de su estructura, secuencia y organización.

Google menciona que existe una suerte de división entre los diferentes Circuitos de Apelación de los Estados Unidos en cuanto a la interpretación y aplicación del artículo 102, inciso b) relacionado al “código de computadora”, y su relación entre el artículo 102 inciso a) y en artículo 102 inciso b).

Oracle menciona que no hay división, que no hay tal misterio y, por lo tanto, los fundamentos del Tribunal de Apelaciones del Circuito Federal son correctos en este aspecto.

Oracle menciona que la Petición II de *Writ of Certiorari* presentada por Google posee un error fundamental.

La pregunta realizada por Google a la Corte Suprema en relación a la "protección bajo derechos de autor" se focaliza exclusivamente acerca del código fuente declarado, de lo que Google llamó en su escrito "interfaces de software".

Ahora bien, dicha pregunta no se extiende a la parte del fallo del Tribunal de Apelaciones del Circuito Federal que reconoció la protección bajo el sistema de derechos de autor a la estructura, secuencia y organización de las API.

De hecho, la petición de Google nunca hace referencia a la protección de la estructura y organización.

La única mención que realiza Google en relación a la estructura y organización es la afirmación que el código declarado "incorpora" la estructura y organización de los paquetes de API.

Pero Google no disputa que la estructura y organización tiene derecho a la protección bajo derechos de autor con independencia del código fuente declarado.

Google hubiera sido igualmente responsable, aunque no hubiese copiado ni una sola línea de código declarado si sólo hubiera copiado o duplicado, como lo hizo, la estructura y organización de los paquetes de las API.

Por otro lado, Google menciona, como lo hizo en su petición I anterior, que la Corte Suprema debe revisar la decisión del

Tribunal de Apelaciones del Circuito Federal para "evitar el devastador impacto en el desarrollo de software".

Si eso hubiese sido cierto, sería esperable que Google hubiese reunido y acompañado prueba de la supuesta devastación en estos cuatro años desde que la Corte Suprema denegó la primera petición de *Writ of Certiorari* de Google en el año 2015.

Pero lo único que ha ofrecido Google son conjeturas de sus abogados, lo mismo que de todos sus *Amicis* (amigos del tribunal), muchos de los cuales promulgan estar a favor de una protección legal más débil para el software.

Google no menciona nada que sugiera que los programadores han abandonado el desarrollo de software mediante la reutilización de bloques de código, o que se ha dejado de hacer los programas interoperables.

Por el contrario, la innovación del software ha prosperado.

Véase el documento llamado *"The Growing $ 1 Trillion Economic Impact of Software"* (sept. 2017) en http://tinyurl.com/y77xjgke

Google menciona que el fallo del Tribunal de Apelaciones del Circuito Federal es inconsistente con la "ley establecida" (en inglés "*settled law"*) en los EE.UU., práctica y expectativas ¿Qué ley?

Una línea ininterrumpida de casos judiciales, han reconocido la protección bajo el sistema de derechos de autor de EE.UU. a código original con menor grado de creatividad que las API de Java y se ha dispuesto que la copia de código no es justa cuando se la incorpora a un productor competidor.

Ni Google ni sus *Amicis* (amigos del tribunal) han citado caso judicial alguno -de ningún Circuito- en donde se haya declarado

que es legal copiar semejante cantidad de código (y su estructura y organización) y que es legal usarla con el mismo propósito y función de la obra originaria (Java) en una plataforma en competencia (Android).

Por otro lado, ¿Qué práctica establecida?

En su petición I anterior, Google ofreció ejemplos de supuestas prácticas comerciales de copiar software sin restricciones o en forma libre.

Oracle los refutó, demostrando que ello constituía una prerrogativa o decisión a instancia de:

(i) el uso autorizado o licenciado, por ejemplo, bajo una licencia de código abierto, o
(ii) de material que no tenía protección bajo el sistema de derechos de autor.

Google y sus *Amicis* (amigos del tribunal) declaran erróneamente "a quién esta decisión afecta".

Este caso se trata de que Google copió sin autorización una obra protegida por el sistema de derechos de autor de EE.UU., de Oracle.

Los "*programadores de apps"* no se encuentran afectados por esta decisión.

De hecho, estos programadores son libres de escribir cualquier programa que ellos deseen utilizando los programas pre-escritos por Oracle disponibles en los paquetes de las API de Java, y sin costo alguno.

La decisión sólo requiere a los "*desarrollares de plataformas*" usualmente corporaciones con tremenda cantidad de recursos como Google, tomar una licencia para incorporar Java dentro de un producto comercial substituto (Android).

Por otro lado, Google y sus *Amicis* (amigos del tribunal) exageran el ámbito del fallo del Tribunal de Apelaciones del Circuito Federal, ya que se da la impresión de que el fallo del Tribunal de Apelaciones del Circuito Federal se refiere o abarca a que cualquier cosa que pueda ser etiquetado como "interface" está protegida bajo el sistema de derechos de autor y no puede ser copiado, y ello, no es así, ya que no se dijo ello por el Tribunal de Apelaciones del Circuito Federal.

Por ello, de nuevo, Google quiere capitalizar las imprecisiones de sus propios términos utilizados "interfaces de software".

No hay una ley o disposición legal para "interfaces de software".

Como sucede con cualquier tipo de obra, que una interface de software posea protección bajo derechos de autor o que pueda ser copiada justamente o legítimamente depende "de que es y de cómo es usada".

La decisión del Tribunal de Apelaciones del Circuito Federal se refirió a la plataforma de software en particular de Oracle, que posee miles de programas con una estructura más que compleja- y al uso en particular que Google hacía de ella.

Por otro lado, algunas interfaces son simplemente una cadena de caracteres no protegibles, careciendo de originalidad y expresión.

Otras pueden ser más expresivas, pero, sin embargo, puede ser considerado justo copiarlas para los propósitos correctos, como por ejemplo la investigación.

Ahora bien, para la mayoría de las interfaces, la pregunta casi nunca surge, ya que como describen los *Amicis* (amigos del tribunal), teniendo en cuenta los fuertes incentivos que existen

en el mercado, las compañías permiten reutilizar libremente u ofrecer términos de licenciamiento flexibles para que terceros pueden “crear o desarrollar productos interoperables, o que, puedan interoperar con los productos de esas compañías, -que son los titulares de derechos de esas interfaces-.

Ahora bien, este es “su derecho”, su decisión, pero eso no significa que ello borre los derechos legales de otras compañías que desean comercializar sus obras protegidas de una manera diferente.

Por todas estas razones, Oracle sostiene que la petición de Google debe ser denegada.

CAPITULO XIII

SENTENCIA DE LA CORTE SUPREMA DE JUSTICIA DE LOS EE.UU.

La Corte Suprema de Justicia de EE.UU. (SCOTUS), integrada por los jueces[113] Samuel A. Alito, Clarence Thomas, John G. Roberts, Stephen G. Breyer, Sonia Sotomayor, Brett M. Kavanaugh, Elena Kagan, Neil M. Gorsuch, y Amy Coney Barrett (que no tomó parte en las consideraciones y decisión del caso -tal vez debido a que ocupó su lugar el 27 de Octubre de 2020-) con fecha 5 de Abril de 2021 en una votación dividida de 6 votos (emitieron opinión Breyer, Roberts, Sotomayor, y adhirieron Kagan, Gorbush y Kavanaugh) contra 2 votos (Thomas emitió en contra, adhiriendo Alito) falló a favor de la empresa Google aplicando la teoría del "uso justo o legítimo" (o en inglés "*fair use*").

En su sentencia SCOTUS no trata la cuestión de si los paquetes de las API de Java que conforman la librería estándar de Java, y su estructura, secuencia y organización son protegibles bajo el sistema de derechos de autor de los EE.UU., sino que *asumió* que la sentencia I dictada el 9 de mayo de 2014 por el Tribunal de Apelaciones del Circuito Federal (Circuito Noveno) de los EE.UU., la cual había reconocido la protección bajo el sistema de derecho de autor de EE.UU. de los paquetes de las API de Java y su estructura, secuencia y organización, *no era necesaria modificarla*.

Por lo tanto, la sentencia de SCOTUS *no modifica* el fallo de del Tribunal de Apelaciones del Circuito Federal en cuanto a la protección de las API de Java y de su estructura, secuencia y

[113] Véase en https://www.supremecourt.gov/about/justices.aspx y https://www.supremecourt.gov/about/biographies.aspx

organización bajo derechos de autor, y *sólo modifica* la sentencia II dictada el 27 de marzo de 2018 por el Tribunal de Apelaciones del Circuito Federal, *reversándose la decisión allí establecida*, y expresándose por SCOTUS que el uso de las API de Java y su estructura, secuencia y organización por parte de Google había sido "justo o legítima" ya que Google *había tomado solamente aquello que era necesario para garantizar la interoperabilidad*.

El presente caso se relaciona con dos límites establecidos en la ley de derechos de autor de EE.UU.

(i) El primero de ellos la ley lo establece en su artículo 102, inciso b) Titulo 17 USC y dice que la protección conferida por las leyes de derecho de autor de EE.UU. no se extiende a *"cualquier idea, procedimiento, proceso, sistema, método de operación, concepto, principio, o descubrimiento..."*

(ii) El segundo límite a los derechos de autor está marcado en el artículo 107 del Título 17 USC en el cual se establece que *el autor o titular de derechos no puede impedir a otro de efectuar un "uso justo" respecto de la obra protegida* por su autor o titular de derechos.

La petición efectuada por Google a la Corte Suprema de Justicia solicita aplicar ambas limitaciones a la copia efectuada por Google en el presente caso.

La Corte Suprema de EE.UU. *asumió que las líneas copiadas por Google pueden ser protegidas, y al respecto del uso realizado por Google de esas líneas de código, consideró que constituía "uso justo"*.

VOTO DE LA MAYORIA

En el VOTO DE LA MAYORIA liderado por la opinión del Juez BREYER, se dijo al respecto *"para los propósitos de exposición se comenzará con el segundo factor".*

Veremos con posterioridad cuando se refiera al voto disidente ¿qué se expresa respecto a este comienzo de análisis por el voto de la mayoría?

SEGUNDO FACTOR. NATURALEZA DE LA OBRA PROTEGIDA

La API de Java de Sun Microsystems es una "interface de usuario". Provee una manera a través de la cual los usuarios (aquí los programadores) pueden "manipular y controlar" a los "programas informáticos para que realicen tareas" a través de un menú de comandos.

Así las cosas, las API de Java reflejan la división hecha por Sun Microsystems de posibles tareas que una computadora "podría realizar" dentro de un conjunto actual de tareas que cierto tipo de computadoras "realizará".

De hecho, Sun Microsystems decidió por ejemplo que una de sus API pudiera llamar a una tarea para que se compare cuál de dos números es el mayor.

Nadie en estos actuados menciona que las decisiones acerca de lo que constituye una tarea sea protegible por derechos de autor, pero, sin embargo, si es argumentable como de protegible, que una decisión de *cómo denominar y organizar dichas tareas lo sea*, por ejemplo, la decisión de nombrar a cierta tarea como "***max***" o el lugar de "***max***" dentro de una clase llamada "***Math***".

En su forma de verlo, la Corte Suprema menciona que la tecnología en cuestión posee tres elementos esenciales.

(i) El primero, la API *incluye* el "código implementado", el cual instruye a la computadora sobre los pasos que deben ser llevados a cabo para cada tarea. Al respecto, Google escribió su propio "código fuente implementado", que realizaría cada una de esas tareas que sus API llamarían.

(ii) En segundo lugar, las API de Java de Sun Microsystems, asocia a un comando en particular denominados "métodos de llamada" ("*method call"*), con la llamada de cada tarea. Así ***java.lang***, es parte del comando que llamará al programa que instruye a la computadora de llevar a cabo la operación de "número mayor". Oracle no argumenta que el uso de esos comandos por parte de los programadores, violen los derechos de autor de Oracle.

(iii) En tercer lugar, las API de Java de Sun Microsystems *contienen código*, llamado código declarado, que asocia un método de llamada con el "lugar" particular en una computadora que contiene el código implementado que necesita. El código declarado denomina las tareas particulares dentro de la API y organiza tales tareas o métodos, dentro de "clases" y "paquetes".

Oracle argumenta que Google viola sus derechos de autor en la utilización del código declarado de las API de Java.

El código declarado en cuestión se asemeja a otras obras protegidas en el sentido de que “es parte de un programa de computación”.

Así, el Congreso de los EE.UU. ha establecido que los programas de computación están sujetos a derechos de autor.

Ahora bien, aquel código declarado, difiere de otros tipos de código de computadoras protegible bajo los derechos de autor.

Así, el código declarado:

a) se encuentra *inextricablemente unido* junto con un sistema general, de división de tareas de cómputo, que nadie reclama que está sujeto a derechos de autor.

b) también está *inextricablemente unido* con la idea de organizar esas tareas en clases y paquetes, que dicha idea tampoco es protegible bajo la ley de derechos de autor y también,

c) está *inextricablemente unido* al uso especifico de comandos conocidos por los programadores, llamados en el juicio como métodos de llamadas (en inglés “*method calls”*), como ser ***java.lang.Math.max*** que Oracle no disputa, y a su vez,

d) está *inextricablemente unido* con el código implementado, el cual es protegible por derechos de autor, pero que no ha sido copiado.

Entonces, menciona la Corte Suprema, que para escribir la implementación de un programa, conforme surge de las constancias del juicio, se requiere un balance en las consideraciones acerca de que tan rápido una computadora podría ejecutar una tarea determinada o del tamaño de la memoria necesaria requerida para ejecutar esas tareas.

Esto constituye un proceso creativo que fue necesario para desarrollar Android para ser usado no en computadoras de

escritorios o en computadoras portátiles, sino en un nuevo contexto, el de los teléfonos móviles inteligentes.

El código declarado (que es inseparable de los métodos de llamadas de los programadores) incorporan un diferente tipo de creatividad.

Sun Microsystems en la creación de Java, trato de encontrar nombres para su código declarado que provocara intuitivamente una manera fácil de ser recordado.

La idea era atraer a los programadores para que pudieran aprender el sistema de Java, y así, lograr su desarrollo.

Así se dijo en el juicio, que el código declarado está orientado al usuario (en inglés "*user facing"*) y debe estar diseñado y organizado de forma tal que sea intuitivo y entendible para que los desarrolladores los puedan invocar.

También conforme constancias del juicio, Sun Microsystems quería lograr "abrir" las API y luego competir en las implementaciones.

Todas estas características significan, como parte de la interface de usuario, que el código declarado difiere de alguna manera con otros programas de computadora.

Como otros programas de computadora, es funcional por naturaleza. Pero, a diferencia de otros programas, su uso esta inherentemente unido con ideas no protegibles por derecho de autor (división general de tareas y organización) y nuevas expresiones creativas (código implementado de Android).

Así a diferencia de otros programas, su valor deriva del valor de aquellos que no poseen derechos de autor, es decir, de los programadores, que invierten su propio tiempo y esfuerzo en aprender el sistema de las API.

Y a diferencia de otros programas, su valor recae en sus esfuerzos para alentar a los programadores para aprender y usar el sistema de manera tal que estos usarán (y seguirán haciéndolo) programas relacionados implementados por Sun Microsystems, que Google no ha copiado.

Se expresa por la Corte que, aunque el derecho de autor protege varias formas de escritos, también se ha mencionado la necesidad de reconocer que existen ciertas obras protegidas que están más cerca que otras del núcleo de protección del derecho de autor.

Por lo tanto, por las razones descriptas, el código declarado, como protegible, se encuentra más lejos que otros programas de computación -como el código implementado- del núcleo de protección por los derechos de autor.

Por lo tanto, el segundo factor se inclina por el uso justo.

PRIMER FACTOR. EL PROPÓSITO Y EL CARÁCTER DEL USO

En el contexto del uso justo la Corte Suprema ha considerado que el uso al copiar debe agregar algo nuevo, con un propósito y un carácter diferente, alterando la obra protegida, con una nueva expresión, significado o mensaje.

A esto se lo ha denominado como uso “transformativo” describiéndose así, que ese uso -cuando se copia- debe agrega algo nuevo e importante.

Google copió porciones de las API de Java de Sun Microsystems en forma precisa, y lo hizo en parte por las mismas razones que tuvo Sun Microsystems cuando las creó, es decir,

permitir a los programadores invocar o llamar a los programas implementados para que cumplirían con determinadas tareas.

Ahora bien, expresa la Corte Suprema que podría decirse que virtualmente todos los usos no autorizados de programas de computación (por ejemplo, para investigación o enseñanza) harían lo mismo, y entonces detenerse en este estadio limitaría severamente el ámbito del uso justo en el contexto funcional de los programas de computación.

Entonces, para determinar si un uso es *transformativo* se debe ir más lejos y así debe examinarse con mayor detenimiento el propósito y carácter de lo copiado.

En el caso, el uso de Google de las API de Java buscó crear un nuevo producto.

Busco expandir el uso y la utilidad de los teléfonos móviles inteligentes basados en Android. Este nuevo producto, Android, ofrece a los programadores una herramienta altamente creativa e innovadora para el ambiente de los teléfonos móviles inteligentes.

Entonces, en el sentido de que Google usó partes de las API de Java para crear una nueva plataforma que permitiera ser rápidamente usada por los programadores, este uso es consistente con el progreso "creativo" que es el objetivo constitucional del derecho de autor en sí mismo.

El objetivo primario del sistema de derecho de autor no es compensar a los autores por su trabajo, sino "Promover el Progreso de la Ciencia y las Artes aplicadas".

Por ello, conforme a las constancias del juicio surge que Google limitó el uso de las API de Java a tareas específicas de Android.

También surge que Google copió las API de Java, pero sólo aquellas que eran necesarias para incluir las tareas que serían útiles en los programas para teléfonos móviles inteligentes.

Y también que se lo hizo solo en la medida de lo necesario para permitir a los programadores invocar aquellas tareas, sin descartar la porción del lenguaje de programación con la cual ya estaban familiarizados, evitándose de que tuvieran que aprender uno nuevo.

Por ello, Google a través de Android, proveyó una nueva colección de tareas que operan en un ambiente de computación diferente, y estas tareas son llevadas a cabo a través de la utilización del nuevo código implementado que Google escribió y diseño para poder operar en ese nuevo ambiente.

Conforme a las constancias del juicio a ello se lo denominó *"reimplementación"*, que sería una suerte de construcción de un sistema que reutiliza las mismas palabras y sintaxis de un sistema existente.

Existen numerosas formas en cuanto a cómo la reimplementación de una interface puede promover el desarrollo de programas de computación.

Así se explicó como las interfaces son necesarias para que dos programas de computación se puedan comunicar entre ellos, de cómo la reutilización de las API es común en la industria, y que la misma Sun Microsystems había usado interfaces preexistentes en la creación de Java.

También que los ejecutivos de Sun Microsystems pensaban que la diseminación del uso del lenguaje de programación Java, incluyéndose el uso en las plataformas de teléfonos móviles inteligentes beneficiaría a Sun Microsystems.

Los *Amici* (amigos del tribunal) presentados en el juicio apoyando la posición de Google mencionaron estos mismos puntos.

Con respecto a las consideraciones de uso comercial y buena fe, la Corte Suprema menciona que no hay duda que un hallazgo de uso no comercial inclina la balanza en favor del uso justo.

Ahora, explica que lo contrario no es necesariamente cierto, ya que en muchos casos que se ha determinado uso justo dicho uso era comercial. Por ejemplo, se menciona que el artículo 107 incluye en su listado como ejemplo a los "reportajes de noticias", que mayormente son realizados con la finalidad comercial y de obtener ingresos.

Ahora bien, aunque el uso de Google sea comercial ello no es dispositivo del primer factor, y ello particularmente a la luz del rol inherentemente transformativo que jugó la reimplementación del nuevo sistema de Android.

Por otro lado, con respecto a la mala fe dijo la Corte Suprema que con la fuerza en que el resto de los factores se inclinaban hacia la consideración de un uso justo, este punto en particular no era determinante en el contexto del juicio.

Con lo cual la Corte Suprema concluye que el propósito y carácter de la copia realizada por Google fue "transformativo", y por lo tanto que el factor primero sopesa en favor del uso justo.

TERCER FACTOR. CANTIDAD Y SUSTANCIALIDAD DE LA PARTE UTILIZADA EN RELACIÓN A LA OBRA PROTEGIDA POR DERECHOS DE AUTOR EN SU CONJUNTO

En el análisis del tercer factor se expresa que si se considerase el código declarado en *forma aislada* la cantidad copiada por Google sería relevante.

Google copió el código declarado de 37 paquetes de las API de Java en un total aproximado de 11.500 líneas de código.

Estas líneas de código, representa todo el código declarado necesario para llamar o invocar los cientos de diferentes funciones o tareas.

Por el contrario, si se mirase la *totalidad* de los paquetes de las API de Java que incluye el código declarado más el código implementado por Sun Microsystems que suman un total de 2.86 millones de líneas de código, la cantidad de líneas copiada por Google sería poco representado esas 11.500 líneas un 0,4%.

Por lo tanto, la pregunta es si esas 11.500 líneas de código copiadas por Google deben ser vistas en forma "aislada" o como "un todo".

En casos anteriores se ha sostenido que una pequeña cantidad copiada podría quedar por fuera del ámbito del uso justo cuando lo copiado constituye el "corazón" de la expresión de la obra original.

Pero también se ha sostenido que una gran cantidad de material copiado podría quedar comprendido dentro del ámbito del uso justo cuando lo copiado captura poco de la expresión de la obra o si lo copiado es central o neurálgico para su propósito.

Así, si se ha copiado una sola oración de una novela, tal copia podría bien ser insustancial.

Pero si esa oración copiada establece una de las historias más cortas del mundo de tal forma que ese supuesto insustancial material copiado es el total de la novela, la cuestión sería muy

diferente. (citándose por la Corte Suprema la novela de Augusto Monterroso[114] reconocida como la obra más corta en la historia de la literatura *"The Dinousaur, in Complete Works & Others Stories 42 por E. Grossman transl. 1995* que decía en su versión original en castellano "Cuando despertó, el dinosaurio todavía estaba allí" *"when he awoke, the dinousar was still there"*)

Entonces expresa la Corte que de las características de lo copiado por Google sugiere que la manera de enfocar el asunto es mirando las líneas de código que Google no copió.

Entonces, lo copiado por Google no fue porque lo considerase creativo, o por su belleza, o incluso en algún sentido por su propósito. Por el contrario, Google lo copió porque los programadores ya habían aprendido a cómo usar las API de Java, y sería muy difícil, e incluso tal vez prohibitivo, atraer a los programadores para que construyeran o desarrollaron en la plataforma Android sin ellas.

La sustancialidad del factor generalmente pesa a favor del uso justo cuando como en el caso, el monto de lo copiado se encuentra atado a un propósito válido y transformativo.

El objetivo de Google no era solamente hacer que el lenguaje de programación de Java sea el usado en la plataforma Android, sino que, además era permitir a los desarrolladores hacer uso de su conocimiento y experiencia ya ganada utilizando las API de Java cuando estos escribieran nuevos programas para teléfonos móviles inteligentes en la plataforma Android.

---

114 Véase https://en.wikipedia.org/wiki/Augusto_Monterroso y http://biographiesii.blogspot.com/2018/12/augusto-monterroso.html
Augusto Monterroso, who has died aged 81, wrote the shortest story in the history of literature; entitled The Dinosaur, it ran: "When he woke up, the dinosaur was still there."

Es cierto que Google podría haber creado su propio código declarado, ahora no es menos cierto que podría pensarse que con ello Google no podría haber logrado su objetivo básico.

En este sentido, el código declarado era el necesario para desbloquear las "energías creativas de los programadores".

Y Google necesitaba de esas energías para crear su propio sistema innovador Android.

Por todas estas razones, se entendió que el tercer factor de sustancialidad se inclina a favor del uso justo.

CUARTO FACTOR. EL EFECTO DEL USO SOBRE EL MERCADO POTENCIAL O EL VALOR DE LA OBRA PROTEGIDA POR DERECHOS DE AUTOR

El cuarto factor se focaliza en el "efecto" que la copia tiene en el mercado o el valor de la obra original.

Ahora, menciona la Corte Suprema que el análisis cuando se trata de programas de computación puede ser más complejo.

En algún punto se requiere que los jueces consideren cuál es el monto de dinero que la obra protegida podría perder.

Entonces, la Corte Suprema en casos anteriores ha sostenido que la copia exacta (en inglés "*verbatim"*) de la obra original en su totalidad con propósitos comerciales puede producir la sustitución en el mercado de la obra originaria.

Por ejemplo, realizar una película de un libro podría significar una pérdida potencial o presunta para el autor o titular de los derechos de la obra originaria.

Dichas pérdidas se consideran normalmente en conflicto, y no comulgan con los principios básicos establecidos por el sistema de derechos de autor que es proveedor a los autores con derechos exclusivos que estimula la expresión creativa.

Pero, la potencial pérdida de ingresos no lo es todo.

Es decir, no sólo se debe considerar el monto potencial de la pérdida sino también la "fuente de la pérdida".

Así, en un caso anterior la Corte Suprema sostuvo que una parodia "letal" como podría ser una crítica mordaz en un teatro, puede acabar con la demanda de la obra original. Y este tipo de daño, aun cuando pueda trasladarse directamente a la pérdida de ingresos, no se encuentra reconocida bajo el sistema de derechos de autor.

Por otro lado, se debe tener presente el "beneficio público" que la copia podría probablemente producir.

Entonces habría que preguntarse, ¿estos beneficios están relacionados con la preocupación del derecho de autor por la creación de nuevas expresiones? o ¿son o no comparativamente importantes, cuando se los compara con las pérdidas probables de ingresos?

La Corte Suprema menciona que no argumenta que estás preguntas sean siempre relevantes a la hora de realizar el análisis de uso justo, o incluso menos en el ámbito de los programas de computación, o que estás preguntas sean las únicas que la Corte Suprema podría hacerse.

Pero, si afirma que estas preguntas son relevantes en este caso para ayudar a determinar los probables efectos que la *reimplementación* de Google tiene en el mercado.

Entonces, en relación a la probabilidad de pérdida de ingresos se podría pensar -por el Jurado- que Android no causó daño en el mercado actual o potencial para Java SE.

Y también de que el mismo Sun Microsystems (ahora, Oracle) no habría sido capaz de entrar a estos mercado- actual y potenciales- con éxito, y ello con independencia de si Google hubiese copiado o no las API de Java.

Afirma la Corte Suprema, que conforme surge de la prueba rendida en autos, sin tener en cuenta la tecnología de la plataforma Android para los teléfonos móviles inteligentes, Sun Microsystems estaba muy mal posicionado para tener éxito en el mercado de los teléfonos móviles.

Por otro lado, la prueba mostró que el mercado primario de Sun Microsystems era el de computadoras de escritorios y el de las computadoras portátiles o notebooks.

También surge de la prueba que muchos de los esfuerzos realizados por Sun Microsystems para ingresar al mercado de teléfonos móviles no había tenido éxito.

Antes de Android, en el año 2006 los ejecutivos de Sun Microsystems proyectaron una baja de ingresos en los teléfonos móviles a causa de la tecnología emergente de los teléfonos móviles inteligentes.

Por otro lado, existe constancias en el juicio de las que surgen que los dispositivos que utilizan la plataforma Android son distintos de aquellos que están licenciados con la tecnología de Sun Microsystems.

Así surge de la prueba que la industria en general distingue entre teléfonos móviles inteligentes y “teléfonos móviles con funciones”.

Y respecto de los dispositivos que utilizan la tecnología creada por Sun Microsystems, surge que uno de esos dispositivos no poseía una pantalla táctil, mientas que otros no poseían un teclado QWERTY, y otros como Kindle, si bien utilizaba Java, sus modelos más avanzados como Kindle Fire fueron construidos con el sistema operativo de Android.

De ello surge, que, en lugar de reutilizarse código de computadoras de escritorio y portátiles, la plataforma Android de Google fue parte de un mercado distinto y más avanzado que el mercado del software Java SE.

Tomando todos estos antecedentes en forma conjunta surge que el negocio de teléfonos móviles de Sun Microsystems estaba en declive mientras que el mercado demandaba con mayor exigencia cada vez más de una nueva forma de tecnología de teléfonos móviles inteligentes, que Sun Microsystems nunca había sido capaz de ofrecer.

Por otra parte, también surge de las constancias del pleito que Sun Microsystems previó un beneficio como consecuencia de la amplia utilización del lenguaje de programación Java en una nueva plataforma como la de Android.

Así se dijo *"una vez que una API comienza a reimplementarse, se sabe que será un éxito"*.

Y como hay dos mercados en cuestión, los programadores entrenados en el lenguaje de programación Java en relación al trabajo efectuado en el mercado de los teléfonos móviles inteligentes son capaces de llevar ese aprendizaje al otro mercado, el de las computadoras portátiles.

Si bien Oracle presentó prueba de lo contrario, y además el Tribunal de Apelaciones del Circuito Federal mencionó que el cuarto factor militaba en contra del uso justo, en parte porque

Sun Microsystems había intentado entrar en el mercado de Android, lo cierto es que ni de los esfuerzos realizados por Sun Microsystems para licenciar la tecnología, ni de la prueba aportada por Oracle en el juicio se puede contrarrestar la otra prueba que indica que como mínimo hubiera sido difícil para Sun Microsystems lograr entrar al mercado de los teléfonos móviles inteligentes, y todo ello aún en el caso de que Google no hubiese copiado los 37 paquetes de las API de Java.

Por otro lado, lo copiado por Google ayudó a Google a obtener una cantidad de ingresos más que importantes con su plataforma Android (aproximadamente unos 42 billones de dólares).

Y, por otro lado, la persecución de los derechos de autor de las API de Java podría darle a Oracle, una parte significativa de esos ingresos.

Cuando una interface, como la de las API o un programa de hojas de cálculo es lanzado al mercado, puede atraer a sus nuevos usuarios porque su expresión es de calidad, como puede serlo una mejor pantalla o que es superior funcionalmente. Con el paso del tiempo, podrían tener valor por razones diferentes, es decir porque sus usuarios incluyendo programadores, también los usan. Es decir, ya han aprendido a como trabajar con ellos.

Entonces, en el juicio existe vasta evidencia de que Google tenía el deseo de utilizar las API de Java.

Ahora bien, la fuente rentable de ingresos de Android está muy relacionado y tiene mucho que ver con la inversión de "terceras partes" (los programadores) en los programas de Java de Sun Microsystems. En consecuencia, no tiene que ver con la inversión de Sun Microsystems en la creación de las API de Java.

No hay razón para pensar que el sistema de derechos de autor de EE.UU. busca proteger la inversión de terceras partes por el aprendizaje logrado de como operar una obra creada.

Entonces, permitir la prosecución de los derechos de autor de Oracle podría ir en contra del interés general.

Explica la Corte Suprema, que teniéndose en cuenta los costos y las dificultades de producir API alternativas, y de que estas posean un similar atractivo para los programadores, permitir la prosecución en este caso haría que el código declarado de las API de Java opere como un "*cerrojo"* limitando el futuro creativo de nuevos programas.

Sólo Oracle tendría la llave.

Sin duda, el resultado podría ser muy beneficioso para Oracle (o de otras compañías que sostengan la protección de las interfaces de computación).

Pero esas ganancias bien podrían fluir de mejoras creativas, nuevas aplicaciones, y nuevos usos desarrollados por usuarios quienes han aprendido a trabajar con tales interfaces.

En este sentido, el *"cerrojo"* interferiría con los principios básicos del derecho de autor.

Así, la Corte Suprema menciona que ha sostenido en casos anteriores que el intento de monopolizar el mercado haciendo imposible para otros de competir va en contra de los propósitos de la ley de derechos de autor de promover las expresiones creativas.

Después de todo, el derecho de autor provee del incentivo económico para crear y diseminar las ideas, y la reimplementación de una interface de usuario, permite la

creación de nuevo código fuente para lograr entrar al mercado de una manera más accesible.

Entones, dada la naturaleza incierta de la habilidad de Sun Microsystems para competir en el mercado de Android, la fuente de sus ingresos perdidos, y el riesgo de la creatividad relacionado con el daño público, tomando todos ellos en conjunto la Corte Suprema expresa que el cuatro factor se inclina favor del uso justo.

VOTO DE LA MINORIA

En el VOTO DE LA MINORÍA disintiendo, el Juez THOMAS dijo, y adhiriendo el Juez ALITO:

Se menciona que el voto de la mayoría arribó a este resultado improbable en parte porque eludió la primera pregunta que Google solicitó responder a la Corte Suprema en su Petición II de *Writ of Certiorari.*

La primera pregunta fue: *¿Se encuentra protegido por las leyes de derechos de autor de EE.UU. el código en cuestión?*

El voto de la mayoría *asumió que sí, sin decidir, que el código estaba protegido*.

Pero su análisis, en palabras de Thomas, de uso justo es inconsistente con la protección que la ley de derechos de autor de EE.UU. otorga al código de computadora.

Al eludirse la pregunta de protección de derechos de autor, el voto de la mayoría descartó la mitad del texto relevante de la ley de derechos de autor, y distorsionó su propio análisis de uso justo regulado por el artículo 107, Título 17, USC.

El voto de la mayoría concluyó que cada uno de los cuatro factores estaba a favor de Google, confiando en gran parte en la distinción entre "código fuente declarado y código fuente implementado", una distinción que la ley de derechos de autor de EE.UU. no efectúa.

Así el Congreso de los EE.UU. cuando decidió establecer la protección al código de computadora eligió hacerlo mediante el sistema de derechos de autor, y en esa elección de protección se encuentra incluido el "código fuente declarado".

Así, la ley define a programa de computación como *un conjunto de declaraciones o instrucciones para ser usadas directa o indirectamente en una computadora con el propósito de producir un resultado.*

Por lo tanto, el Congreso de los EE.UU. cuando decidió la protección bajo derechos de autor y definió al código de computadora como lo hizo, rechazó cualquier distinción o categorización entre código declarado y código implementado.

El código implementado ordena a una computadora una operación de forma directa, y el código declarado lo hace de manera indirecta.

Entonces, sin duda que la definición legal claramente incluyó al código fuente declarado *como conjunto de declaraciones que indirectamente ejecutan funciones de cómputo mediante el código implementado.*

Incluso sin un lenguaje expreso en la ley, el código declarado cumple con los estándares de protección exigidos por la ley de derechos de autor.

Las líneas de código declarado en la plataforma Java SE satisfacen estos requisitos.

En primer lugar, está expresada en palabras, números, u otros símbolos verbales o numéricos y es original por cuanto fue creada en forma independiente por Sun Microsystems.

Por otro lado, son originales porque Oracle podría haberlas creadas o escrito, y así expresarlas de diferentes formas.

Google ha sostenido que el código declarado no puede quedar protegido por las leyes de derechos de autor porque constituye un "método de operación" conforme a lo estipulado en el artículo 102 inciso (b) del Título 17 USC.

Este argumento, sostiene el Juez Thomas no es correcto.

Así del voto de la mayoría surge correctamente que el código declarado y el código implementado están *inextricablemente unidos*.

Ello por cuanto el código declarado define el ámbito de un conjunto de código implementado y provee a los programadores una manera de usarlos mediante un atajo o acceso directo (en inglés *"shortcut"*).

El código declarado incluye al código implementado, pero el código declarado no funciona por sí mismo.

Lo mismo le pasa al código implementado.

En ausencia del código declarado, los desarrolladores tendrían que escribir cada programa desde cero (en inglés "*from scratch"*), y posiblemente haría que el desarrollo de programas complejos sea prohibitivo de desarrollo a razón del consumo de tiempo que ello llevaría.

En definitiva, está claro que Oracle no puede proteger mediante derechos de autor la idea de usar código declarado, pero si puede reclamar protección bajo el sistema de derechos de

autor respecto de la expresión especifica que ha hecho de esa idea y que se la encuentra en la librería estándar de Java

Google también ha sostenido que el código declarado no es protegible bajo el sistema de derechos de autor basándose en la doctrina de la fusión (en inglés "*merge doctrine"*).

Oracle tuvo múltiples maneras de expresar su idea.

A lo sumo, existió una sola manera en la cual Google pudo copiar el código declarado expresado por Oracle, pero existieron innumerables maneras para Oracle de escribir su código declarado.

Por lo tanto, este argumento de Google no puede más que ser rechazado.

El voto de la mayoría de forma muy particular decidió evaluar los factores del uso justo no por su orden secuencial o por su orden de importancia (según los precedentes de la Corte Suprema existen dos factores más importantes [115] que los restantes), sino por el contrario, decidió empezar por el segundo factor que trata sobre la naturaleza de la obra protegida.

Ahora, según el Juez Thomas el voto de la mayoría procedió de esta forma para crear una distinción entre código declarado y código implementado, dándole menor protección al código declarado, y de ahí construir su teoría.

Por ello, dice el Juez Thomas que teniendo en cuenta que el equivocado análisis del voto de la mayoría confió tanto en este

[115] El cuarto factor -el efecto de lo copiado por Google en el mercado potencial para la obra de Oracle, es sin duda el elemento individual más importante del uso justo. *Véase Harper & Row, Publishers, Inc. v. Nation Enterprises (1985).* Por su parte, el primer factor -propósito y carácter del uso, incluyendo si el uso es comercial o no, constituye el segundo factor más relevante.

factor, él también decidió comenzar por el mismo segundo factor.

SEGUNDO FACTOR. NATURALEZA DE LA OBRA PROTEGIDA

Este factor requiere una evaluación respecto del nivel de "creatividad o funcionabilidad" en la obra original.

De modo general se suele decir que este factor favorece el uso justo cuando la obra protegida posee carácter "informativo o funcional", en lugar de *creativo*.

Y también, que teniendo en cuenta que el código de computadora es predominantemente funcional este factor también suele favorecer la copia cuando la obra original trata de código de computadora.

Ahora bien, teniendo en cuenta que el sistema de derechos de autor promulga la protección tanto del código declarado como el código implementado, este factor por sí sólo no puede soportar el hallazgo de uso justo.

El voto de la mayoría utilizó este factor para crear una diferencia entre el código declarado y el código implementado, y así remover de protección al código declarado.

Así el voto de la mayoría concluye, que a diferencia del código implementado el código declarado se encuentra más alejado, menos próximo al "núcleo de protección de los derechos de autor", porque se convierte en valioso sólo cuando terceras partes (programadores) lo valoran y porque está inherentemente unido junto con ideas que no son protegibles bajo el derecho de autor.

Comenta nuevamente el Juez Thomas, que el Congreso de los EE.UU. rechaza esta distinción que haría que el código declarado tuviese una protección precaria o menor.

La ley de derechos de autor protege al código que opera en una computadora con el propósito de obtener un cierto resultado, sea directo (código implementado) sea indirecto (código declarado).

Y en todo caso, si hubiese algo que estaría más cerca del "núcleo del derecho de autor" sería el código declarado.

Así se ha dicho durante el juicio que el código declarado es orientado al usuario y que los desarrolladores ni siquiera pueden ver el código implementado.

El código declarado debe estar diseñado y organizado de manera que sea intuitivo y entendible para los programadores de tal forma que puedan invocarlo.

Aún, dejando de lado todo esto, menciona el Juez Thomas que el voto de la mayoría es insostenible.

El voto de la mayoría dice que el código declarado está inherentemente unido con ideas que no son protegibles bajo el derecho de autor.

Ahora, se pregunta Thomas, ¿acaso un libro no está inherentemente unido a ideas que no están protegidas?

Así en el uso de, capítulos, diálogos, pie de notas.

Esto no hace que los libros estén lejos del núcleo del derecho de autor.

De hecho, el código implementado, el cual en el voto de la mayoría se concede que es protegible, está inherentemente unido "con la división de las tareas de cómputo", que no es protegible.

En el voto de la mayoría se desprecia al código declarado al sugerirse que éste simplemente es una forma de organizar el código implementado.

No es así.

El código declarado define los subprogramas del código implementado, incluyendo mediante el control de los *inputs* que ellos pueden procesar.

El código declarado "crea" los métodos de llamadas (en inglés "*method calls"*).

En definitiva, no se puede descartar a una obra como de protegible por la sola mera circunstancia de que se encuentra asociada o ligada a ideas que no son protegibles.

Mientras las ideas no pueden ser protegidas por el derecho de autor, la expresión de esas ideas si se protegen.

El código declarado es como los programadores acceden al código implementado pre-escrito.

Por ello el valor del código implementado es directamente proporcional a como los programadores valoran el código declarado asociado.

CUARTO FACTOR. EL EFECTO DEL USO SOBRE EL MERCADO POTENCIAL

Sin lugar a dudas, el cuarto factor constituye el elemento individual más importante en el análisis de uso justo.

Expresa el Juez Thomas que la evidencia en el juicio que surge del daño actual y potencial a causa de lo copiado por Google, es abrumadora.

Al copiar el código de Oracle y luego desarrollar y lanzar Android, Google arruinó el mercado potencial de Oracle en menos de dos años.

En primer lugar, Google eliminó las razones por las cuales los fabricantes podrían querer pagar por instalar Java en sus dispositivos.

Mientras que los ingresos de Oracle se generan por cobrarles a los fabricantes de dispositivos por instalar Java en sus aparatos fabricados, Google obtiene sus ingresos principalmente a través de sus ventas de publicidad.

Su estrategia al lanzar Android fue para ofrecerlo sin costo a los fabricantes de dispositivos y así utilizar a Android como vehículo para recolectar datos de consumidores y así entregar publicidad basada en comportamiento.

En definitiva, teniendo presente que había disponible un producto sin costo, *"gratis"*, que incluía el código de Oracle, los fabricantes de dispositivos no vieron muchas razones para seguir pagándole a Oracle por embeber Java SE en sus dispositivos.

Por ejemplo, antes de que Google hiciera disponible Android, Amazon pagaba a Oracle por una licencia por incluir Java SE en sus dispositivos Kindle.

Luego de que Google liberará Android al mercado, Amazon utilizó el argumento de que Android era gratuito para obtener un descuento de Oracle del 97, 5%.

De igual forma, de la prueba rendida en el juicio surge que Samsung tenía un contrato con Oracle de USD 40 millones de dólares. Después de Android ese mismo contrato fue de USD 1 millón de dólares.

Google menciona que Amazon usaba una plataforma distinta de la plataforma Java SE, la cual era la plataforma Java Micro Edition (ME). Esta diferencia es irrelevante, ya que Java Micro Edition es un conjunto de Java SE.

Google copió código de ambas plataformas.

El voto de la mayoría no lo disputa, ni siquiera lo menciona.

Thomas, menciona que sin duda el daño fue enorme.

Por otro lado, Google interfirió con las oportunidades de Oracle de licenciar la plataforma Java para los desarrolladores de sistemas operativos para dispositivos móviles.

Antes de que Google copiara el código de Oracle, en casi todos los teléfonos móviles del mercado se encontraba Java SE.

El código de Oracle sin duda fue extraordinariamente valioso para cualquiera que quisiera desarrollar teléfonos móviles inteligentes, y eso explica porque Google trato unas cuatro veces de obtener una licencia de Oracle.

El voto de la mayoría recalca sobre esto, que Google busco obtener una licencia de Oracle, pero de otro tipo.

Pero eso no cambia las cosas.

Ambas partes estuvieron de acuerdo que Oracle podía entrar al mercado actual de Google licenciando su código declarado.

Entonces, al copiar el código de Oracle y lanzar Android, Google eliminó la oportunidad de Oracle de licenciar su código para tal uso.

El voto de la mayoría descarta este daño diciendo que se podría haber concluido que Oracle no hubiera sido capaz de entrar al mercado moderno de los teléfonos móviles inteligentes de una manera exitosa.

Ahora, que Oracle por sí mismo pudiese haber entrado o no a tal mercado, constituye la mitad de la película.

Ello porque cuando se analiza el mercado potencial que los creadores de una obra original probablemente desarrollarían, no sólo se mira lo que ellos podrían desarrollar por sí mismos, sino también aquello que podría desarrollarse a través del modelo de licenciamiento a terceros para que lo realicen.

Es decir, desarrollarlo de manera indirecta con una red de distribuidores autorizados por Oracle.

Es indiscutible que Oracle podría haber licenciado a terceros para que estos desarrollasen la tecnología de Oracle en el nuevo mercado de teléfonos móviles inteligentes.

Nuevamente, que Oracle no lo hubiese podido hacer por sí sólo, en forma directa, no significa que no hubiese podido hacerlo de manera indirecta, licenciando o autorizando a otros para hacerlo.

Dice Thomas a su vez, que el voto de la mayoría incapaz de discutir seriamente que las acciones de Google causaron un efecto desastroso en el mercado potencial de Oracle, cambia el curso y afirma que promover la protección del derecho de autor podría dañar al público por darle a Oracle el poder de limitar el futuro de la creatividad de los programas de Android.

Ahora, por un lado, este caso sólo involucra las *versiones de Android hasta noviembre de 2014.*

Desde esa fecha Google ha publicado seis versiones.

Y sólo el 7,7% de los dispositivos activos de Android contienen las versiones anteriores al 2014.

La preocupación del voto de la mayoría acerca del posible *vendor lock-in* de parte de Oracle tendría más sentido se si

estuviera discutiendo acerca de las versiones actuales en uso de Android o de las que estarían en uso.

No hace demasiado sentido cuando se trata de versiones que están a punto de ser obsoletas.

A su vez, es especulativo.

En primer lugar, Oracle nunca tuvo en sus manos el *vendor lock-in.*

El voto de la mayoría se olvida que Apple y Microsoft crearon su propio sistema operativo para teléfonos móviles sin usar el código declarado de Oracle.

En segundo lugar, Oracle siempre hizo disponible el código declarado disponible bajo una licencia de software libre y código abierto (GPLv2).

El voto de la mayoría expresa preocupación en relación a que Oracle pueda abusar de sus derechos de autor (en las versiones obsoletas de Android) y realizar un intento de monopolizar el mercado.

Ahora, menciona el Juez Thomas, que fue Google quién recientemente fue condenado a una multa de USD 5 billones por cometer prácticas ilegales en relación con Android y violar leyes antimonopólicas.

Entonces, es Google quién controla el sistema operativo para teléfonos móviles más usado en el mundo.

Si el voto de la mayoría está preocupado en relación a las conductas monopólicas, debería considerar si Google es la gran amenaza.

Si ahora las compañías pueden libremente copiar código declarado cuando sea más conveniente que escribir el suyo

propio, dudarán lógicamente de gastar los recursos que Oracle gastó.

Al copiar el código de Oracle, Google diezmo el mercado de Oracle y creó un sistema operativo para dispositivos móviles con 2.5 billones de dispositivos, obteniendo ganancias por más de decenas de billones de dólares cada año.

Si estos efectos sobre el potencial mercado de Oracle, favorecen a Google, dice Thomas algo está muy mal en nuestro análisis de uso justo.

El juez Thomas, entiende que este factor también favorece a Oracle.

PRIMER FACTOR. EL PROPÓSITO Y EL CARÁCTER DEL USO

Este factor constituye el segundo factor más importante.

Como se mencionó anteriormente este factor incluye si el uso efectuado es de naturaleza comercial o es con fines educativos sin ánimo de lucro.

Este factor también requiere analizar si el uso es "transformativo".

Sólo en el año 2015, explica Thomas, el año anterior al juicio de uso justo, Google obtuvo una ganancia de USD 18 Billones (o USD 18.000 millones lo que es lo mismo) en relación a Android.

Sin duda que el uso de Google del código declarado de propiedad de Oracle, si no es decisivo por lo menos pesa muy fuerte en contra del uso justo.

El voto de la mayoría intentar minimizar este abrumador uso comercial sosteniendo que el uso comercial no necesariamente juega en contra del uso justo.

Es cierto, expresa el Juez Thomas, el uso comercial a veces se ve superado por un uso que es lo suficientemente transformativo. Ahora, no se puede ignorar el propósito de Google de suplantar la valiosa plataforma comercial de Oracle con la suya propia.

Generalmente no podría encontrarse uso justo si el uso del copista no es transformativo.

Una obra es transformativa si agrega algo nuevo, con un propósito diferente, alterando la primera expresión, significado o mensaje.

Esta pregunta es guiada por los ejemplos provistos en el artículo 107, Titulo 17 USC, cuando incluye como ejemplos, y de manera simplemente enunciativa, a las críticas, enseñanza, educación, becas, investigación, etc.

Google, en su reutilización del código de Oracle no efectúo ninguno de ellos, como tampoco uso el código de Oracle para enseñar, o efectuar ingeniería inversa o para asegurar la compatibilidad entre sistemas.

Por el contrario, Google utilizó el código declarado para los mismos exactos propósitos que Oracle.

Por estas razones, se entendió que este factor se inclina por la inexistencia de uso justo.

TERCER FACTOR. CANTIDAD Y SUSTANCIALIDAD DE LA PARTE UTILIZADA, EN RELACIÓN A LA OBRA PROTEGIDA POR DERECHOS DE AUTOR EN SU CONJUNTO

En general a mayor cantidad en uso o copiada, menor la probabilidad de uso justo.

Pero aún, así, si el copista toma una pequeña parte, copiando el "corazón" o "puntos centrales" de una obra, ello pesa en contra de un uso justo, a menos que no se haya tomado más de lo necesario por el copista para lograr su uso transformativo.

Google no ha disputado la conclusión del Tribunal de Apelaciones del Circuito Federal cuando concluyó que Google había copiado el corazón o la parte central de la obra protegida de Oracle.

El código declarado es lo que atraída a los programadores a la plataforma Java y por ello Google estaba tan interesado en ello.

Que Google haya copiado exacto (en inglés "*verbatim"*) el código de Oracle, pesa en contra del uso justo.

El voto de la mayoría expresa que Google tomo no más que lo necesario para crear nuevos productos.

Esta conclusión es errónea, porque el uso de Google no es transformativo.

A su vez, aunque el uso de Google fuera transformativo, el voto de la mayoría concluye que Google sólo copió una pequeña porción de la obra original. Señala así el voto de la mayoría que las 11.500 líneas de código declarado copiado por Google, constituye solamente "una fracción" del código de la plataforma Java.

El problema con esta visión es que el denominador correcto no es “todo el código” de la plataforma Java SE, sino “sólo” el código declarado.

Por ello, este factor pesa en contra del uso justo.

Concluye el Juez Thomas que tres de los cuatro factores pesan decididamente en contra de Google.

Y que la naturaleza de la obra protegida, el único factor que tal vez podría favorecer a Google, por sí sólo no puede lograr una determinación de uso justo, porque sostener ello sería contrariar los principios que el Congreso de los EE.UU. tomó en cuenta al determinar la protección bajo derechos de autor del código declarado.

Finaliza diciendo Thomas, que el voto de la mayoría propone dejar “*para otro día”* la cuestión de si el código declarado se encuentra protegido o no por el derecho de autor.

La única razón aparente para ello, es que no pueden cuadrar su análisis de uso justo con el hecho de que el código declarado es protegible bajo los derechos de autor.

CAPITULO XIV

LISTADO DE PREGUNTAS NO EXAUSTIVAS, OBJETIVAS, COMO DE MINIMA Y OBLIGATORIAS

Como se ha mencionado al inicio de este trabajo el objetivo del mismo no es brindar opiniones personales.

Muy por el contrario, el objetivo que se intenta con este trabajo es poner a disposición de cada lector, independientemente de su jurisdicción, la posibilidad de qué a través de un listado de preguntas con las características de *"no exhaustivas, objetivas, como de mínima y obligatorias"* puedan los lectores obtener una conclusión propia acerca del caso judicial relatado.

Para obtener una *conclusión* válida la *hipótesis* de partida debe ser válida, y para ello, se deben brindar todos los elementos para que dicho punto de inicio sea válido.

En este caso, tal elemento de partida lo constituiría *"lo expresado"* por cada una de las partes litigantes a través de sus escritos judiciales, y *"lo expresado"* por los jueces a través de sus sentencias.

Para lograr dicho objetivo, parte del trabajo minucioso que se ha tratado realizar ha sido recolectar y extraer de todas las presentaciones judiciales -lógicamente de las que se ha podido tener acceso- tales "elementos".

Y en esa recolección y extracción también se ha tratado como el mayor de los cuidados la objetividad en la *selección del material*, es decir, que cada presentación realizada por una de las

partes posee como contrapartida la presentación de la contraparte.

Entonces, por citar un ejemplo de lo constituye un hecho objetivo: quedo probado en el juicio que Google basó Android en la implementación independiente de la especificación de las API de Java realizado por la Fundación de Software Apache[116], es decir, Google no tomó el código fuente declarado "directamente" de Java SE propiedad de Sun/Oracle, ni siquiera lo hizo de la versión *open-source* (GPLv2) de Java, llamada Open JDK, sino como surge de la causa judicial, Google lo hizo del proyecto Harmony de la Fundación Apache.

A su vez, también constituye un hecho objetivo probado en el juicio que la implementación de la especificación de las API de Java de Sun/Oracle por parte de la Fundación Apache nunca obtuvieron la aprobación del test de compatibilidad de Java o TCK de Sun/Oracle.

Como se expresa tales hechos fueron probados en el juicio, y como tales son objetivos cuando se lo expone en el presente trabajo.

Así entonces, todo lo narrado en los capítulos anteriores no es más que la "extracción real de lo dicho por las partes y jueces en

[116] *April 28, 2016. Opinion or Order Filing 1748 ORDER IN LIMINE RE ORACLE'S MOTION IN LIMINE NO. 2 TO EXCLUDE EVIDENCE OF APACHE HARMONY AND GNU CLASSPATH by Judge William Alsup [denying #1552 Motion in Limine]. The Court reserves its ruling on evidence regarding GNU Classpath for the reasons stated in this order. (whasec, COURT STAFF) (Filed on 4/28/2016).* *https://www.plainsite.org/dockets/download.html?id=235606491&z=2d4c55f9*

el juicio", las partes han hablado a través de sus presentaciones judiciales y argumentos orales, y los jueces lo han hecho a través de sus sentencias.

A su vez, la terminología de *"no exhaustivas"* posee el significado que otras preguntas podrían agregarse al listado realizado No hay duda de ello.

Por su parte, el requisito de *"no exhaustivas"* se relaciona con el requisito de *"de mínima y obligatorias"*, con el cual se quiere significar que como mínimo ciertas preguntas de base deben existir, pero también porque de los hechos narrados es técnicamente imposible no efectuarse ciertas preguntas, por ello lo de *"obligatorias"*.

Finalmente, el requisito de *"objetivas"* la misma palabra lo indica, las preguntas son abstractas, agnósticas, independientes a la posición que cada una de las partes ha tomado en el pleito en relación a lo que cada una de ellas considera como protegible por el sistema de derechos de autor de EE.UU. y en su caso las limitaciones que corresponde o no aplicar.

Este juicio ha sido muy particular, porque su disputa no sólo ha sido vertida por las partes en los estrados judiciales, sino también que se ha dado por terceros en los medios de comunicación, sobre todo en la red internet.

Así existe una gran cantidad de información relacionado al caso, no sólo desde lo judicial sino también desde lo técnico y comercial, comentado por distintas partes con distintos intereses y motivaciones.

En algún punto esto fue tan llamativo, que hizo que el Juez William Alsup emitiera el 7 de agosto de 2012 una orden judicial *"Order regarding Disclosure of Financial Relationship with Commentators on Issues in this Case"*[117] "muy particular".

El Juez ordenó a Oracle y Google que informaran al respecto de todos los autores, periodistas, comentaristas o *bloggers* los cuales hayan reportado o comentado al respecto del caso en cuestión y hayan recibido algún pago -distinto a lo que sería un pago por una suscripción normal de un servicio- por alguna de las partes o sus abogados mientras se encuentre pendiente el trámite de lo que iba del juicio.

Entonces, la *"objetividad"* se relaciona a que lo narrado en los distintos capítulos surge conforme a la información expuesta en las presentaciones judiciales, y no de otra fuente, como puede ser un blog de noticias. Y de nuevo, ello es así, porque es parte del objetivo del trabajo, y no se relaciona a emitir juicio de valor sobre lo expuesto en diferentes sitios periodísticos acerca del caso judicial.

---

[117] La orden del Juez Alsup expresa: "The Court is concerned that the parties and/or counsel herein may have retained or paid print or internet authors, journalists, commentators or bloggers who have and/or may publish comments on the issues in this case. Although proceedings in this matter are almost over, they are not fully over yet and, in any event, the disclosure required by this order would be of use on appeal or on any remand to make clear whether any treatise, article, commentary or analysis on the issues posed by this case are possibly influenced by financial relationships to the parties or counsel. Therefore, each side and its counsel shall file a statement herein clear identifying all authors, journalists, commentators or bloggers who have reported or commented on any issues in this case and who have received money (other than normal subscription fees) from the party or its counsel during the pendency of this action. Disponible en http://www.groklaw.net/pdf3/OraGoogle-1229.pdf

A su vez, las preguntas *"no exhaustivas, objetivas, como de mínima y obligatorias"* están formuladas de forma *agnóstica* o con *independencia* a las distintas posibles jurisdicciones en materia de derechos de autor. Por lo cual sin importar la jurisdicción y ley aplicable del lector las hipótesis de las preguntas son válidas para colaborar en la elaboración de la propia conclusión de cada lector.

Por último, las preguntas se ofrecen en un orden determinado que no es al azar, sino que, por el contrario, se intentó hacerlo con la finalidad de ayudar al lector en su razonamiento, y así se comienza desde la pregunta más básica hacia las más complejas.

La idea es la misma, colaborar en que cada lector pueda construir su propia conclusión.

¿Por qué esta metodología?

Simplemente porque ese fue el objetivo al iniciar este trabajo de investigación.

La idea principal ha sido intentar mostrar a través de una narración simple lo expresado por las partes litigantes y los jueces intervinientes, basándose para ello el autor en las presentaciones legales y resoluciones judiciales de lo debatido y resuelto en el litigio. La idea es simple, pero el desafío ha sido enorme.

Las preguntas "*no exhaustivas, objetivas, como de mínima y obligatorias"* son las siguientes:

1. ¿Cómo se define por las leyes de derechos de autor de su jurisdicción a los “programas de computación”?

2. ¿Qué tan semejante es dicha definición dada en su jurisdicción a la brindada por el artículo 101 del Título 17 USC? *“conjunto de declaraciones o instrucciones para ser usadas directa o indirectamente en una computadora con el propósito de llevar a cabo un resultado”*

3. En las leyes de su jurisdicción que regulan y otorgan protección bajo el sistema de derechos de autor, ¿existe alguna diferencia en la exigencia para la protección de los “programas de computación” de las demás obras protegidas?

4. ¿Cuáles son los requisitos exigidos en las leyes de derechos de autor de su jurisdicción para la protección de un “programa de computación”?

5. Cuándo en las leyes de derechos de autor de su jurisdicción se otorga protección a los “programas de computación”, ¿se regula dicha protección tanto en su versión de código objeto como de código fuente?

6. Si es así, es decir que se protege el código fuente ¿dicha ley efectúa alguna diferencia para la protección del “código fuente declarado” y “código fuente implementado”?

7. En las leyes de derechos de autor de su jurisdicción ¿se hace referencia a “código fuente” en general?, o ¿se

regula acerca de “funciones de prototipos” “interfaces de software”, “encabezados” o “declaraciones de código”?

8. Conforme a las leyes de derechos de autor de su jurisdicción, ¿es protegible bajo derechos de autor la implementación de un lenguaje de programación como programa de computación?

9. Dicha de otra manera, ¿Todas las implementaciones que conforman programas de computación son protegibles en su jurisdicción bajo las leyes de derechos de autor? o ¿existe algún tipo de programa de computación que no es protegible en su jurisdicción?

10. Conforme a las leyes de derechos de autor de su jurisdicción, y a la interpretación de las mismas por los jueces de su jurisdicción ¿se encuentran protegidos, los elementos calificados por el Tribunal de Apelaciones del Circuito Federal (Circuito Noveno) como “no literales” de un programa de computación”, es decir, entre ellos la estructura, secuencia y organización de un programa de computación? Si ello fuera así, ¿en qué circunstancias y que elementos deberían ser ponderados?

11. Conforme a las leyes de derechos de autor de su jurisdicción y a la interpretación de dichas leyes por los tribunales, ¿existe en su jurisdicción alguna *interpretación judicial* que haya determinado que cierto tipo de obra protegida, específicamente cierto tipo de código fuente (código fuente declarado) se encuentre más alejado al núcleo de protección conforme al sistema

de derechos de autor que otro tipo de código fuente (código implementado)?

12. Independientemente del tipo de lenguaje de programación, y para el caso que exista una implementación, cree Ud. que una librería de software estándar ¿forma parte del lenguaje de programación implementado? ¿o por el contrario es independiente de dicho lenguaje implementado?

13. ¿En qué casos las leyes de derechos de autor de su jurisdicción justifican la *descompilación o procesos de ingeniería inversa* o similares de código de terceros sin autorización o licencia del titular o propietario del código (como un todo) o programa de computación en cuestión?

14. Conforme a las leyes de derechos de autor de su jurisdicción, y para el caso que se lo haga, ¿se regula o se interpreta por los tribunales a la "interoperabilidad" entre programas de computación como una excepción o limitación necesaria a los derechos de autor?

15. Por otro lado, conforme a su legislación, ¿en qué momento debe ser analizada la interoperabilidad, al momento de creación o al momento de la utilización o copia? Para el caso ¿qué factores externos se toman en cuenta para su análisis?

16. ¿En qué norma de las leyes de derechos de autor de su jurisdicción se codifica el principio que establece la "división entre las ideas y su expresión" es decir donde la protección bajo derechos de autor se extiende a la expresión de las ideas, pero no a las ideas en sí mismas, semejante a lo dispuesto en el artículo 102 inciso b) del Título 17 del USC, que dice: *"en ningún caso el derecho de autor para la autoría de una obra original se extiende a las ideas, procedimientos, procesos, sistemas, métodos de operación, conceptos, principios o descubrimientos, sin importar la forma en las cuales estos sean descriptos, explicados, ilustrados, o incorporados en la obra en cuestión."*?

17. Conforme a las leyes de derechos de autor de su jurisdicción, ¿la protección acordada a la expresión de una idea, se extingue por la circunstancia que dicha expresión se haya incorporado en un método de operación?

18. Es decir, el derecho de autor de su jurisdicción ¿protege la "expresión de un método de operación"?

19. En definitiva, conforme al régimen de derechos de autor de su jurisdicción una obra original ¿se encuentra sujeta a protección bajo el derecho de autor siempre que el autor posee múltiples formas de expresar la idea? o más allá del carácter expresivo que pueda tener la obra original se niega dicha protección por constituir un método de operación?

20. Conforme a las leyes de derechos de autor de su jurisdicción, ¿cómo se interpreta los términos “originalidad” y “expresión” en programas de computación?

21. Es decir, ¿Cuándo las líneas de código que conforman un programa de computación son originales y expresivas?

22. En las leyes de derechos de autor de su jurisdicción ¿se regula a las interfaces de software?

23. Si es ello así, ¿se las excluye de protección de derechos de autor?, y si es así, ¿es excluida de protección por derechos de autor en todos los casos?, ¿o sólo en algunas situaciones reguladas?

24. Conforme a las leyes de derechos de autor de su jurisdicción, y a la interpretación de las mismas por los tribunales ¿en qué momento se debe analizar la “expresión creativa”, es decir, las varias formas u opciones disponibles de expresar una idea ¿al momento de su misma creación por el autor al escribir el código y representar una determinada funcionalidad? o ¿al momento que dichas líneas de código son copiadas por un tercero sin autorización de su autor?

25. Conforme a las leyes de derechos de autor de su jurisdicción, ¿existe en la misma regulación alguna limitación o excepción que se asemeje a la limitación a los derechos de autor conforme lo establecido en el artículo 106 del Título 17 USC de uso justo o legítimo (en inglés "*fair use")*?

26. Conforme a las leyes de derechos de autor de su jurisdicción, y suponiendo que la utilización de cierto tipo de código definido como el del litigio en cuestión llamado código declarado o funciones de prototipos o interfaces de software o API, no fuera protegible en su jurisdicción ¿cómo influiría ello en la interpretación de las licencias de software libre y código abierto (FOSS) GNU GPL, AGPL y LGPL, -y de cualquier otra similar al respecto que pudiera contener similares obligaciones- en relación a la técnica de combinación de código mediante vínculos estáticos y dinámicos y a la posibilidad que ello pueda o no generar la creación de una obra derivada de software?

27. Conforme a las leyes de derechos de autor de su jurisdicción, y a la interpretación de las mismas por los tribunales, ¿se reconoce, y más allá de su denominación, a la "copia literal" y a la "copia no literal" de elementos literales (código fuente y código objeto) y no literales de un programa de computación (entre otros la estructura, secuencia y organización)?

28. Suponiendo que conforme a las leyes de derechos de autor de su jurisdicción el código fuente declarado no fuese protegible por derechos de autor en sí mismo, ¿podría considerarse protegible su estructura, secuencia y organización?

29. Si ello fuese así, ¿en qué casos se podría proteger la estructura secuencia y organización del código declarado de un programa de computación?

30. Y conforme a su legislación de derechos de autor, ¿se podría considerar que el código fuente implementado en forma independiente por un tercero constituye una obra derivada cuando este contenga en forma literal o no literal la estructura, secuencia y organización del código fuente declarado?

31. Una implementación independiente ¿puede ser considerada como una obra derivada de la especificación del software implementado?

MAPA DE RESOLUCIONES JUDICIALES

El cuadro de más abajo es una suerte de mapa para guiar al lector al respecto de resoluciones judiciales definitivas recaídas en el litigio.

Resolución Judicial	Fecha de emisión	Asunto	A favor de
-1- Veredicto del Jurado	7 de mayo de 2012	Google infringió los derechos de autor de Oracle en relación a los paquetes de las API de Java	Oracle
-2- Veredicto del Jurado	23 de mayo de 2012	Google no infringió los derechos de patentes de invención de Oracle (reivindicaciones	Google

		11, 27, 29,39,40 y 41 patente RE 38104) y reivindicaciones 1 y 20 patente 6.061.520	
-3- Sentencia de la Corte de Distrito del Norte de California (Juez William Alsup)	31 de mayo y 20 de junio de 2012	Reversa el veredicto del Jurado del 7 de mayo de 2012, expresando que las API de Java y su SSO no son protegibles bajo el sistema de derechos de autor de EE.UU.	Google
-4- Sentencia I del Tribunal de Apelaciones del Circuito Federal de EE.UU. (jueces O´ Malley,	9 de mayo de 2014	Reversó la sentencia de la Corte de Distrito del Norte de California del 31 de mayo y 20 de junio de 2012, expresando que las API de Java y SSO son protegibles bajo	Oracle

Plager y Taranto)		el sistema de derechos de autor de EE.UU. Nota: no se expide sobre el "uso justo o legítimo" y manda a llevarse a cabo un *segundo juicio* al respecto.	
-5- Denegación de Petición I *Writ of Certiorari* (recurso de apelación ante la Corte Suprema de EE.UU.) presentado por Google contra la sentencia del Tribunal de Apelaciones del Circuito Federal del 9 de mayo de	26 de mayo de 2015.	La Corte Suprema de EE.UU. solicita la opinión del Procurador General (Donald B. Verrili, Jr.), y éste expresó estar de acuerdo con la sentencia dictada por el Tribunal de Apelaciones del Circuito Federal del 9 de mayo de 2014 y por ello sugirió rechazar el recurso de apelación	Oracle

2014 sentencia I (6 de octubre de 2014)		interpuesto por Google. Posteriormente la Corte Suprema decidió denegar el *Writ of Certiorari* I.	
-6- *Segundo juicio* relacionado a la determinación del "Uso justo o legítimo" Veredicto del Jurado y Sentencia de la Corte de Distrito del Norte de California	8 de junio de 2016	Se determinó que el uso realizado por Google era un uso justo o legítimo	Google
-7-	27 de marzo de 2018	Revierte la sentencia de la	Oracle

Sentencia II del Tribunal de Apelaciones del Circuito Federal de EE.UU. (jueces O´ Malley, Plager y Taranto)		Corte de Distrito y Decisión del Jurado de fecha 8 de junio de 2016, determinándose que el uso de las API de Java por parte de Google no constituía uso justo o legítimo bajo el sistema de derechos de autor.	
-8- *Writ of Certiorari* II. Recurso de Apelación ante la Corte Suprema de EEUU presentado por Google con fecha 24 de enero de 2019 (respondido por Oracle el 27 de marzo	15 de noviembre de 2019	La Corte Suprema de EE.UU. solicitó opinión al procurador General, Noel J. Francisco y con fecha 27 de septiembre de 2019 expresó que en su opinión el recurso presentado por Google debía ser denegado. Por su	Google

de 2019, y nuevamente respondido por Google el 10 de abril de 2019)		parte la Corte Suprema, con fecha 15 de noviembre de 2019 decidió aceptar el recurso *Writ of Certiorari* II interpuesto por Google. Google abrió el debate con fecha 6 de enero de 2020 presentando sus fundamentos a su apelación. Oracle los contestó el 12 de febrero de 2020. Se dio lugar para que los *Amicis Curiae* de las partes presenten sus opiniones y con fecha octubre de 2020 se escucharon los argumentos orales de ambas partes.	

-9- Sentencia de la Corte Suprema de los EE.UU.	5 de abril de 2021	En una votación dividida (6 votos contra 2) la Corte Suprema de EE.UU. no trató la cuestión de si las API de Java y SSO eran protegibles bajo el sistema de derechos de autor de los EE.UU., sino que ***asumió*** que la sentencia I dictada por la Corte Federal de Apelaciones, ***no era necesaria modificarla.*** Por lo tanto, la sentencia de SCOTUS ***no modifica*** el fallo de la Corte Federal de Apelaciones en cuanto a la protección de las	Google

		API de Java y de SSO bajo derechos de autor, **y *sólo modifica*** la sentencia II dictada por el Tribunal de Apelaciones del Circuito Federal, ***reversándose la decisión allí establecida,*** y expresándose por SCOTUS que el uso de las API de Java y SSO por parte de Google había sido "justo o legítima"	

ANOTACIONES LECTORES