LIGAÇÕES PERIGOSAS? – REFLEXÕES SOBRE *APIS* E DIREITO DE AUTOR A PARTIR DO ACÓRDÃO *GOOGLE V. ORACLE* DO SUPREMO TRIBUNAL DOS EUA

NUNO SOUSA E SILVA*

> "Netscape inflicted irreparable damage on Microsoft's stronghold on the computing industry: our work moved developers from Win32 API, Microsoft's proprietary platform, to the Internet. Someone writing new functionality for computers no longer wrote for Microsoft's proprietary platform. Instead, they wrote to the Internet and World Wide Web's standard interfaces".
>
> Ben Horowitz, *The Hard Thing about Hard Things,* Harper Collins, 2014, p. 15.

Abstract: Application programming interfaces (APIs) are an essential part of the operation of contemporary computer systems. Their treatment by copyright law is still unclear. This text analyses the US Supreme Court's Google v. Oracle decision and critically reflects on APIs and their status under European Copyright Law.

Keywords: Software; APIs; Copyright; Google v. Oracle; Computer science.

Summary: 1. Introduction. 2. What are APIs? 3. *Google v. Oracle*: context. 4. The Supreme Court decision (and the dissenting opinion). 4.1. The decision: fair use. 4.2. The dissenting opinion: protection of APIs. 5. What if it were in Europe? 6. Conclusion.

Resumo: As interfaces de programação de aplicações (*application programming interfaces* – APIs) são uma parte essencial do funcionamento dos sistemas informáticos contemporâneos. O seu tratamento pelo Direito de Autor ainda não é claro. Este texto parte da análise da decisão *Google v. Oracle* do Supremo Tribunal dos EUA para refletir criticamente sobre as APIs e o seu enquadramento no Direito de Autor Europeu.

Palavras-chave: *Software*; APIs; Direito de Autor; *Google v. Oracle*; Informática.

Sumário: 1. Introdução. 2. O que são APIs? 3. *Google v. Oracle*: contexto. 4. A decisão do S*upreme Court* (e a *dissenting opinion*). 4.1. A decisão: *fair use*. 4.2. A *dissenting opinion:* proteção de APIs. 5. E se fosse na Europa? 6. Conclusão.

* Doutor em Direito. LLM em Propriedade Intelectual e Direito da Concorrência (MIPLC). Advogado e Prof. Auxiliar da Faculdade de Direito da Universidade Católica Portuguesa (Porto). E-mail: nsousaesilva@gmail.com. Site: www.nsousaesilva.pt. Este texto beneficiou da generosidade e amizade de Sérgio Vasconcelos, que se regista e publicamente agradece. Qualquer erro ou imprecisão é imputável ao Autor.

1. Introdução

As Interfaces de Programação de Aplicações (*Application programming interfaces – APIs*) são uma parte essencial do funcionamento dos sistemas informáticos contemporâneos, assegurando a fácil comunicação e interoperabilidade entre diferentes programas de computador. Aquele que desenvolve um programa ou função informática frequentemente disponibiliza uma ou várias APIs, gratuitamente ou em troca de um pagamento, permitindo a terceiros utilizar esse programa ou função, muitas vezes como parte de outro programa ou aplicação[1]. É graças a esta tecnologia que é possível criar uma conta num *site* de um terceiro com um perfil de Facebook ou uma conta Google, pagar com Paypal numa loja *online* ou integrar informação financeira de várias contas bancárias numa única aplicação[2].

Em 5 de Abril de 2021 o *Supreme Court* norte-americano decidiu a título definitivo um litígio sobre APIs que opôs a Google e a Oracle durante dez anos e que esteve no centro das preocupações da indústria do *software* (doravante a "Decisão")[3]. O acórdão relativo àquele que já foi apelidado "de litígio da

[1] O caráter oneroso ou gratuito desta disponibilização dependerá de vários fatores técnicos e jurídicos. A suscetibilidade de proteção jusautoral de APIs não é determinante para a respetiva exploração económica. Nesse sentido é possível estabelecer regras contratuais ou mecanismos técnicos de autenticação para acesso às funções ou programas que dependem de pagamento de um preço. É habitual distinguir-se entre APIs livres, abertas ou públicas (livremente acessíveis, tipicamente de forma gratuita) e APIs privadas (em que há algum método de autenticação, que pode estar dependente de pagamento). Em termos de arquitetura existem também diferentes tipos de APIs, sendo as mais comuns REST (*representational state transfer*), SOAP (*Simple Object Access Protocol*) e RPC (*Remote Procedural Call*).

[2] A Diretiva (UE) 2015/2366 do Parlamento Europeu e do Conselho, de 25 de novembro de 2015 relativa aos serviços de pagamento no mercado interno (conhecida como Diretiva PSD2), transposta em Portugal pelo DL 91/2018, de 12 de novembro, impôs o desenvolvimento de normas abertas, garantindo a interoperabilidade de diferentes soluções tecnológicas de comunicação (cfr. considerando 93). Em geral, esta referência foi entendida como dizendo respeito a interfaces (APIs), o que veio a ser confirmado no Regulamento Delegado (UE) 2018/389, de 13 de março. Sobre o tema cfr. Francisco Mendes Correia, "DSP 2 e Normas Abertas de Comunicação Comuns e Seguras", in: AAVV, *Fintech: Novos Estudos Sobre Tecnologia Financeira,* Almedina, Coimbra, 2019, pp. 157-169. As APIs são cada vez mais objeto de regulação, como no caso do acesso, partilha e portabilidade de dados pessoais (cfr. Orientação sobre o direito à portabilidade dos dados do WP29 (WP 242 rev.01), de Dezembro de 2016) e mesmo para efeito de controlo da atividade de plataformas digitais (veja-se o art. 30º da Proposta Digital Services Act (COM (2020) 825 final), impondo a certas plataformas a disponibilização, por meio de APIs, de dados sobre publicidade).

[3] *Google LLC v. Oracle America*, Inc., 593 U.S. ___, 140 S. Ct. 520 (2021). Tendo em conta que ainda não existe publicação com paginação oficial irei referir-me às páginas da Decisão e às páginas da *dissentig opinion* (doravante "Voto de Vencido") tal como numeradas pelo Tribunal. A audiência e decisão, inicialmente previstas para 2020, foram adiadas devido à COVID-19. Para uma visão completa e detalhada do contexto em que este contencioso surge (até 2017) veja-se Peter S. Menell, "Rise of the API Copyright Dead?:

década"[4] teve como efeito imediato absolver a Google de um pedido indemnizatório que ascendia a 9 mil milhões de dólares. Além disso, tranquilizou a generalidade dos programadores e empresas produtoras de *software*, tendo legitimado aquela que tem sido a prática habitual – a livre utilização de APIs[5].

Este comentário visa apresentar breves reflexões sobre a Decisão, começando por explicar sucintamente o que são APIs (2), apresentar o contexto do litígio (3), para então analisar a Decisão do Supremo Tribunal norte-americano (4). Antes da conclusão (6), equaciona-se como é que a questão seria decidida na União Europeia (5).

2. O que são *APIs*?

Uma interface é uma simplificação (mediante abstração). Da mesma forma que o botão de um elevador serve para o chamar, uma interface informática permite-nos desencadear e controlar uma ação de um sistema sem ser necessário intervir ao seu nível fundamental de funcionamento[6]. Neste domínio estamos habituados a interfaces gráficas de utilização (*Graphic User Interfaces – GUIs*). Na generalidade dos programas de computador do nosso dia a dia as aplicações têm ícones, *dashboards* e menus a partir dos quais, através de periféricos de entrada (como um rato, teclado ou ecrã táctil) comunicamos informação para processamento e obtemos resultados.

Mas também no *backend* ("atrás das cortinas") há abstrações, que permitem aos programadores interagir com outros programas, dados ou funcionalidades, sem ter que dominar todos os detalhes de funcionamento desses programas ou funcionalidades. São as chamadas *application programming interfaces* (*APIs*) [7].

Hoje em dia a maior parte das APIs são disponibilizadas através da *Internet* (assentes em HTTP/HTTPS – dois dos protocolos que subjazem à navegação

An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software", *Harvard Journal of Law & Technology*, 31 (2018) pp. 305-490.

[4] Ieva Vaitkunaite/Mindaugas Beniusis/Paulius Jurcys, "The Lawsuit of the Decade – Google LLC v. Oracle America, Inc.: a Victory for Interoperability and the Future of Innovation", *JIPLP,* (2021), pp. 463-465.

[5] Assim, a propósito da decisão em primeira instância Peter S. Menell, *op. cit.*, p. 386: "the ruling validated what was considered a best practice". Sobre a decisão do Supremo Ieva Vaitkunaite/Mindaugas Beniusis/Paulius Jurcys, *op. cit.*, p. 463: "a clear pro-innovation, pro-consumer message to the software industry"; Mark Lemley/Pamela Samuelson, "Interfaces and Interoperability After Google v. Oracle", *Texas Law Review*, 100 (2021), p. 2: "...important victory for software developers and for an open internet".

[6] A Decisão refere o acelerador num automóvel e as teclas de um teclado como exemplos de interfaces.

[7] Por via de regra estas interfaces são usadas para chamar bibliotecas de funções (*libraries*).

na *web*), com excelentes ganhos para a Humanidade. É em parte graças a esta tecnologia – pela simplicidade que as APIs trazem como meio de comunicação entre sistemas informáticos – que conseguimos beneficiar, nos nossos portáteis ou telefones de serviços com funcionalidades cada vez mais completas e adequadas às exigências dos utilizadores. A interoperabilidade via APIs possibilita a comunicação desses serviços, independentemente da sua localização física[8].

Numa API existem dois pólos, o programa recetor, que fará o processamento dessas instruções, e o programa emissor das instruções (aquele que faz uma chamada à API)[9]. Esta chamada pode ocorrer dentro do mesmo computador, entre aplicações, ou, como é mais comum, entre diferentes sistemas informáticos, através da *Internet*.

Recorrendo a APIs não é necessário aceder ou correr localmente o código de serviços como *Spotify*, *Skyscanner*, ou *Google Maps* para conseguir interagir com esses programas, obtendo dados e/ou o desempenho de uma dada função para integrar numa aplicação ou *website*.

No entanto, para que isto aconteça é preciso que os programas sejam capazes de comunicar entre si, o que só é possível graças à adoção de um padrão – aquilo a que os informáticos chamam o "contrato API", isto é, a documentação onde se declara como é que uma dada API se irá comportar (p.e., OpenAPI specs). Para que alguém use a API terá que "aderir" a esse "contrato", ou seja, usar aquele protocolo de comunicação. Nesse sentido, é necessário que o programa do emissor de pedidos ("chamadas") à API inclua parte das especificações técnicas no seu lado. Está em causa um léxico comum para que o pedido possa ser emitido na "língua" do recetor, aquele que expõe a API.

Esta necessidade de adotar um léxico traduz-se na obrigação de o emissor de um pedido definir o formato em que a comunicação é estabelecida com o recetor. O *Supreme Court* chamou ao código usado para fazer essa definição "*declaring code*" – código declarativo[10]. Este contrapõe-se ao chamado *implementing code* – código de implementação, aquele conjunto de instruções ao abrigo das quais a máquina que recebe a chamada irá desempenhar as funções que forem chamadas pelo

[8] Christopher Millard (ed), *Cloud Computing Law*, Oxford University Press, Oxford, 2021, pp. 165-167.

[9] Normalmente o programa emissor também será recetor de uma resposta (p. e., um resultado ou apenas um código de estado HTTP, tal como 201 "criado" ou 404 "não encontrado"), mas não necessariamente – a API pode ser utilizada apenas num sentido. Em geral, as operações feitas por meio de uma API reconduzem-se a CRUD (*create, read, update and delete*) – que correspondem a 5 "verbos"/funções HTTP: POST, GET, PUT, PATCH e DELETE (existindo outros).

[10] Veja-se pp. 21-22 da decisão. Para um exemplo cfr. pp. 2-3 do Voto de Vencido (em que se compara o *declaring code* a uma definição legal – permitindo evitar repetições do termo definido).

declaring code[11]. As palavras e/ou símbolos utilizados para invocar o código declarativo são designadas por método de chamada ("*method call*")[12].

3. *Google v. Oracle*: contexto

A linguagem Java foi desenvolvida por uma equipa liderada por James Gosling enquanto este trabalhava na empresa Sun Microsystems, a qual veio a ser adquirida pela Oracle em 2010.

No desenvolvimento do sistema operativo Android, lançado em 2007, a Google, tendo em conta o conhecimento disseminado da linguagem de programação Java, copiou parte do *declaring code*, isto é, o método de chamamento de funções dessa linguagem[13]. Mais concretamente, copiou 11500 linhas de código, correspondendo a 37 pacotes Java[14].

Em 2010, a Oracle deu entrada de uma ação judicial contra a Google alegando que o uso que esta fazia da API Java violava direitos de autor e patentes suas[15]. Este pedido foi julgado improcedente – quanto à patente por se entender que não ocorria violação (o uso não estaria abrangido pelas reivindicações), e quanto ao direito de autor porque os elementos copiados (*declaring code* e a estrutura, orga-

[11] Alguns comentadores apontaram que estas designações não correspondem à gíria dos programadores e que a generalidade dos informáticos consideraria que a API consiste apenas no *declaring code* (veja-se Jeffrey Robert Kaufman, "What Google v. Oracle means for open source", https://opensource.com/article/21/5/google-v-oracle).

[12] Existem noções próximas de APIs. Os chamados *webhooks* – que são formas de receber mensagens enviadas automaticamente por um programa quando ocorre um determinado acontecimento – distinguem-se das APIs porque as mensagens não têm caráter reativo, não é necessária uma chamada. Os *webhooks* (também chamados *reverse APIs* ou *web callbacks*) limitam-se a receber informação e não estabelecem uma comunicação bilateral entre os sistemas. Por outro lado, os SDKs (*Software Development Kits*) constituem um conjunto de ferramentas, muitas vezes incluindo APIs, tipicamente disponibilizadas por uma empresa para permitir a terceiros integrar ou usar o seu serviço (p. e., *software* com função de validação de documentos de identificação para ser integrado numa *app* de um banco ou seguradora).

[13] De acordo com o PopularitY of Programming Language Index (PYPL), gerado a partir da frequência com que se pesquisam tutoriais no Google, em 2021 a linguagem Java foi a segunda linguagem mais popular (https://pypl.github.io/PYPL.html).

[14] Pacote (*packages*) são conjuntos de APIs, tipicamente agrupadas por funcionalidade. Peter S. Menell, *op. cit.*, p. 488 apresenta a lista desses componentes e as respetivas funções. Sublinhe-se que a Google não copiou "the task-implementing programas or implementing code" (pp. 8 e 22 da Decisão), ou seja, estão apenas em causa as linhas de código que chamam um método ou função e não aquelas que desempenham esse método ou função.

[15] Com muitos detalhes, incluindo sobre as peças processuais, despachos e audiência de julgamento, veja-se Peter S. Menell, *op. cit.*, pp. 376 e ss.

nização e sequência das APIs) não gozariam de proteção jusautoral em virtude da sua natureza funcional[16].

Em recurso, o Tribunal do Circuito Federal considerou que os elementos copiados tinham sido desenvolvidos com criatividade e eram protegidos por direito de autor[17]. No entanto, admitiu que o uso em causa pudesse ser configurado como *fair use*. Por isso, o litígio baixou novamente aos tribunais de primeira instância para que a questão da livre utilização (*fair use*) fosse julgada.

Entretanto, em 2015, a Google tentou, sem sucesso, um recurso (*petition for a writ of certiorari*) para o Supremo relativo à admissibilidade proteção jusautoral dos elementos copiados[18].

No julgamento de 2016, o júri considerou que a utilização da Google podia ser enquadrada como *fair use*, logo lícita[19]. A Oracle voltou a recorrer para o Tribunal Federal, que considerou que a utilização em causa não preenchia os requisitos do §107 da lei norte-americana de direito de autor[20]. A Google recorreu então para o Supremo norte-americano, que aceitou julgar duas questões: 1) a suscetibilidade de proteção jusautoral dos materiais copiados e 2) a admissibilidade desta utilização ao abrigo da exceção de *fair use*.

4. A decisão do *Supreme Court* (e a *dissenting opinion*)

O problema fundamental da proteção do *software* por direito de autor é a natureza essencialmente funcional do objeto de proteção[21]. Em geral, contrapõe-se que a proteção é de banda estreita, impedindo apenas a cópia literal ("mesmo código") e não abrangendo os aspetos funcionais do programa (excluídos da proteção jusautoral por constituírem ideias, processos sistemas, métodos opera-

[16] *Oracle Am., Inc. v. Google Inc.*, 872 F. Supp.2d 974 (N.D. Cal. 2012).

[17] *Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339 (Fed. Cir. 2014). Sobre a decisão cfr. Peter S. Menell, *op. cit.*, pp. 386-391.

[18] Peter S. Menell, *op. cit.*, p. 391.

[19] Apresentando minuciosamente os argumentos, provas e julgamento *vide* Peter S. Menell, *op. cit.*, pp. 391-413 (prevendo já o recurso desta decisão e o seu provável sucesso).

[20] *Oracle Am., Inc. v. Google Inc.*, 886 F.3d 1179 (Fed. Cir. 2018).

[21] Oliveira Ascensão, "Direito de Autor e Informática Jurídica", in: Oliveira Ascensão, *Estudos sobre Direito da Internet e da Sociedade da Informação*, Almedina, Coimbra, 2001, pp. 9-10 e Richard Arnold, "Copyright in software: functionality", in: *Research Handbook on Intellectual Property and Digital Technologies*, (ed. Tanya Aplin), Edward Elgar, Cheltenham, 2020, p. 26. Sobre o tema, entre nós, a obra incontornável é José Alberto Vieira, *A Protecção dos Programas de Computador pelo Direito de Autor*, Lex, Lisboa, 2005, que sublinha precisamente que a expressão de um programa de computador é uma expressão funcional (pp. 273 e ss.).

cionais ou conceitos)[22]. Por outras palavras, o "mesmo" programa (com as mesmas funcionalidades) escrito numa linguagem de programação diferente não constituirá violação de direito de autor sobre o código do programa original[23].

No caso *Oracle v. Google*, o assunto é mais complexo porque se copiou mesmo parte da expressão literária – parte do código das APIs. É verdade que, em geral, no *declaring code* está em causa uma expressão de natureza funcional, em que a ideia se funde com a expressão – depois de se definir um método de chamada, a única forma de usar aquela interface é com aquelas palavras.

No entanto, neste caso, ao desenvolver as suas APIs para Android a Google podia ter escolhido outro código declarativo, visto que não queria invocar os métodos Java existentes, isto é, usar a interface que já existia[24]. Queria antes criar novas interfaces e, para o efeito, utilizou o léxico conhecido desenvolvido pelos criadores da linguagem Java. Esta utilização tinha a vantagem de utilizar um padrão de comunicação de conhecimento generalizado, tornando a sua utilização mais fácil para os (muitos) programadores fluentes em Java[25].

O Supremo Tribunal norte-americano, num acórdão relatado pelo juiz Stephen Breyer, escolheu não abordar a questão da proteção do *declaring code*, assumindo "for argument's sake" que o material copiado gozaria de direito de autor e concentrou a sua análise na utilização livre (*fair use*)[26]. O tema da proteção

[22] Esta afirmação tem que ser entendida habilmente – em 1983 uma empresa norte-americana que tinha copiado o sistema operativo Apple defendeu-se dizendo que o sistema operativo (*BIOS – basic input/output system*) não era passível de tutela jusautoral atento o seu caráter funcional e o facto de apenas estar disponível em código-máquina. O tribunal do Terceiro Circuito rejeitou esses argumentos – *Apple Computer, Inc. v. Franklin Computer Corp.*, 714 F.2d 1240 (3d Cir. 1983). Como é evidente a cópia literal da expressão de um programa de computador (o código) inclui a sua funcionalidade. Mas, o que que se proíbe diretamente é apenas a cópia da expressão e não da funcionalidade.

[23] C-406/10, *SAS* (EU:C:2012:259), §41. O que não quer dizer que esta prática seja necessariamente lícita. Poderão existir outros fatores, como contratos, outros direitos de propriedade intelectual, concorrência desleal ou segredos de negócio, que qualificam essa "reimplementação".

[24] Em termos técnicos só seriam estritamente necessárias 170 linhas de código para garantir a compatibilidade entre Android e Java (pp. 29-30 da Decisão).

[25] Na p. 29 da Decisão o Tribunal afirma mesmo que a única razão pela qual a Google copiou o código declarativo foi para poder beneficiar do conhecimento já existente em matéria de Java. Como sublinha Noam Shemtov, "Software and Graphical User Interfaces", in: *Research...cit.*, pp. 2-3, este é um problema comum em relação a programas de computador – os utilizadores habituam-se aos interfaces, comandos e modos de utilização dos programas que conhecem e, por isso, é natural que esses elementos sejam imitados por concorrentes.

[26] Pp. 1 e 15 da Decisão. O Tribunal afirma mesmo que, tendo em conta o ambiente de acelerada mudança tecnológica e económica, considera mais avisado não se pronunciar sobre mais do que o necessário para dirimir o litígio. Ora, como referido no Voto de Vencido (p. 7), essa mudança é uma constante no que diz respeito a computadores. Além disso, a discussão em torno da proteção de elementos funcionais de programas de computador não é nova. No caso *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 516 U.S. 233 (1996) o

jusautoral foi discutido apenas na *dissentig opinion* do juiz Clarence Thomas, à qual Samuel Alito aderiu.

4.1. A decisão: *fair use*

No §107 do Copyright Act de 1976, o legislador norte-americano estabeleceu uma cláusula geral através do qual se consideram lícitas determinadas utilizações não autorizadas de obras protegidas por direito de autor, nomeadamente para fins de crítica, comentário, relato de notícias, ensino, estudo ou investigação. A cláusula, que estabelece uma defesa, manda atender a quatro fatores: 1) finalidade e caráter do uso (favorecendo-se uso para fins não lucrativos ou não comerciais), 2) natureza da obra (grau de originalidade e caráter público), 3) porção da obra utilizada (numa perspetiva quantitativa e qualitativa) e 4) o efeito do uso no mercado potencial ou valor da obra utilizada (dando-se especial ênfase ao caráter transformativo do uso)[27]. Trata-se de uma abordagem flexível, que visa evitar que o direito de autor se revele contraproducente, restringindo a criatividade que procura promover[28]. A jurisprudência considera que estes fatores não são exclusivos e sublinha a necessidade de proceder a uma análise casuística[29].

Começando por sublinhar as reconhecidas dificuldades de aplicação do direito de autor a *software*, o Tribunal aponta que nem tudo são desvantagens, dando nota que a cláusula de *fair use* pode desempenhar um papel importante na correta delimitação do âmbito de utilização lícita de direito de autor sobre *software*[30].

Supremo norte-americano confrontou-se com a cópia de parte de um interface de utilizador. O tribunal federal do primeiro circuito tinha rejeitado a tutela jusautoral da interface, mas o supremo empatou 4-4 (devido ao impedimento de um juiz) e a decisão do tribunal inferior manteve-se, mas não formou precedente.

[27] Sobre o tema do *fair use* e a sua possível adoção na legislação europeia vide Tito Rendas, "*Fair Use* na União Europeia (ou os estereótipos das *Copyright Wars*)", *Propriedades Intelectuais*, 3 (2015), pp. 26-39. A bibliografia sobre *fair use* é inabarcável, mas recomenda-se a leitura de Pierre N. Leval, "Toward a Fair Use Standard", *Harvard Law Review*, 103 (1990), pp. 1105-1136 (artigo particularmente influente, que em grande medida definiu os termos da moderna aplicação da cláusula geral); David Nimmer, ""Fairest of them All" and Other Fairy Tales of Fair Use", *Law and Contemporary Problems*, 66 (2003), pp. 263-288 (com uma abordagem crítica da jurisprudência); Neil Weinstock Netanel,"Making Sense of Fair Use", *Lewis & Clark Law Review*, 15 (2011), pp. 715-771 (demonstrando o progressivo alargamento do âmbito do *fair use*); Achim Förster, *Fair Use: Ein Systemvergleich der Schrankengeneralklausel des US-amerikanischen Copyright Act mit dem Schrankenkatalog des deutschen Urheberrechtsgesetzes*, Mohr Siebeck, Tübingen, 2008 (apresentando uma perspetiva comparativa face ao Direito alemão).

[28] *Stewart v. Abend*, 495 U.S. 207 (1990): "fair use (...) permits courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster".

[29] Veja-se pp. 13-14 da Decisão.

[30] Pp. 16-18 da Decisão, com vários exemplos dessa aplicação e rejeitando uma abordagem de tudo ou nada.

Tal como o risco de confusão no direito de marcas, também a questão do *fair use* levanta discussões sobre saber se estamos perante matéria de facto ou matéria de direito. A resposta nos EUA não difere do entendimento que tem valido entre nós: trata-se de um juízo de Direito[31]. Por isso, o Tribunal rejeitou a alegada violação do direito a julgamento por júri consagrado na Sétima Emenda da Constituição norte-americana[32].

Seguiu-se então a análise dos quatro fatores principais usados para a determinação da existência de *fair use*.

Começando pelo segundo fator – a natureza da obra – o Tribunal distinguiu os três componentes de uma API: i) os métodos de chamada (simples palavras utilizadas para invocar um método ou função), em relação aos quais ninguém invoca direito de autor; ii) o código declarativo (aquele que garante a interoperabilidade ao localizar e "traduzir" o método de chamada, invocando uma função) e iii) o código de implementação (aquele que efetivamente desempenha a função). Na visão do Tribunal há dois tipos muito diferentes de criatividade envolvidos. No código declarativo está sobretudo em causa uma classificação e organização de tarefas e funções de forma que seja fácil e intuitivo a um programador invocar certos métodos[33]. O seu valor reside sobretudo no investimento que os programadores façam para aprender essa classificação[34]. Em contrapartida, no código de implementação estamos a lidar com a real operação dos programas, tendo o Tribunal apontado para as exigências de código eficiente, adaptado a dispositivos móveis, que funcionam com base em baterias, que a Google teve que desenvolver[35]. No fundo, o Tribunal desvalorizou a criatividade do código declarativo e sublinhou o facto de a Oracle ter promovido a sua adoção generalizada a título gratuito.

Em relação ao propósito e caráter da utilização, relativizando a importância da natureza comercial[36], o Tribunal sublinhou o caráter transformativo desta utilização, consistente com a noção de progresso criativo estabelecida na Constituição norte-americana[37]. Essa natureza transformativa é evidenciada pelo facto

[31] P. 19 da Decisão. Nesta linha, a propósito do Direito de Marcas, veja-se Acórdão do STJ, de 2 de Outubro de 2003 (p. 03B2236): "é matéria de facto saber se existe ou não semelhança e é matéria de direito apurar quer da existência ou não de imitação em face das semelhanças ou dissemelhanças fixadas pelas instâncias, quer se a imitação assenta numa semelhança capaz de determinar erro ou confusão".

[32] Pp. 18 a 21 da Decisão.

[33] Pp. 22-23 da Decisão.

[34] P. 24 da Decisão.

[35] P. 23 da Decisão.

[36] P. 27 da Decisão.

[37] P. 25 da Decisão.

de o código, concebido para computadores de secretária e portáteis, estar a ser reutilizado num ambiente computacional diferente (*smartphones*) [38].

Quanto à porção utilizada, o Tribunal reconheceu que a contagem pode ser feita de duas formas: limitada ao código declarativo, caso em que será substancial, ou na globalidade da API, correspondendo apenas a 0,4 %[39]. Em qualquer caso, considerou que o propósito da Google foi apenas o de aproveitar o conhecimento existente em Java, pelo que também este fator favorece a existência de *fair use*[40].

Por último, em relação aos efeitos de mercado, o Tribunal entendeu que Android e Java estavam em mercados diferentes e que o falhanço da Sun/Oracle no mercado dos *smartphones* não se devia à Google, o que contribuía favoravelmente para a existência de *fair use*[41].

Em face desta análise, o Tribunal decidiu no sentido da licitude da conduta da Google.

4.2. A *dissenting opinion*: proteção de APIs

Começando por sublinhar aquilo que parece uma injustiça – a cópia literal de onze mil e quinhentas linhas de código, os lucros astronómicos da Google e a destruição do mercado da Oracle – o voto de vencido de Clarence Thomas considera que a opinião da maioria não tratou a questão central e, por isso, fez uma análise incorreta em matéria de *fair use*[42].

Segundo o juiz, a decisão do Tribunal na prática equivale a excluir o código declarativo da tutela por direito de autor[43]. Ora, visto que existe liberdade criativa na forma como este código é escrito[44] e não há qualquer indicação de que o legislador quisesse diferenciar o código declarativo do código de implementação[45] o juiz Clarence Thomas não encontra motivos para a exclusão de proteção.

Nessa linha, rejeita a perspetiva de que o código declarativo corresponda a uma ideia, método ou tenha caráter estritamente funcional, até porque não parece possível separá-lo do código de implementação[46]. De igual forma, defende que a "*merger doctrine*", que afasta a tutela de direito de autor quando uma determi-

[38] P. 26 da Decisão.
[39] P. 28 da Decisão.
[40] Pp. 29-30 da Decisão.
[41] Pp.30 a 35 da Decisão.
[42] Pp. 1-2 do Voto de Vencido.
[43] Pp. 7-8 do Voto de Vencido.
[44] Pp. 4-5 do Voto de Vencido.
[45] Pp. 7 e 9 do Voto de Vencido.
[46] P. 6 do Voto de Vencido.

nada ideia só tenha uma expressão possível, não pode valer aqui, até porque a Apple e a Microsoft desenvolveram o seu próprio *declaring code*, sem copiar o da Oracle/Sun[47].

O Voto de Vencido prossegue com uma crítica violenta à análise feita em sede de *fair use*. Começa por assinalar que entre os dois tipos de código, o código de implementação nem sequer é visto pelos programadores que usam a API, portanto se há algum que tem natureza expressiva é o código declarativo[48]. Sublinha que todas as obras estão inerentemente ligadas às ideias que expressam[49] e que o valor de muitas peças musicais ou teatrais também resulta do facto de os artistas-intérpretes as conhecerem bem e não é por isso que essas obras se podem utilizar livremente[50]. Aponta que os diferentes modelos de negócio (a Oracle cobrava licenças para incluir o código em dispositivos, enquanto a Google distribuiu Android gratuitamente) levaram a grandes prejuízos e destaca o comportamento incorreto da Google (nomeadamente as recentes coimas por violação do Direito da Concorrência)[51]. Clarence Thomas rejeita igualmente a natureza transformativa da utilização feita pela Google, dizendo que para haver um uso transformativo não basta a criação de um novo produto[52], e entende que a porção copiada foi substancial tanto do ponto de vista quantitativo como qualitativo[53].

5. E se fosse na Europa?

Como é sabido, desde 1991 que os Estados-Membros da União Europeia estão obrigados a proteger o *software* como obra literária[54]. O art. 1º/2 da Diretiva 2009/24 estabelece que "...a protecção abrange a expressão, sob qualquer forma, de um programa de computador. As ideias e princípios subjacentes a qualquer elemento de um programa de computador, incluindo os que estão na base das

[47] P. 7 do Voto de Vencido.

[48] P. 10 do Voto de Vencido.

[49] *Idem*.

[50] P. 11 do Voto de Vencido.

[51] Pp. 11-14 do Voto de Vencido.

[52] P. 17 do Voto de Vencido: "Surely the majority would not say that an author can pirate the next version of Microsoft Word simply because he can use it to create new manuscripts".

[53] P. 18 do Voto de Vencido.

[54] Tal resultou da Diretiva 91/250/CEE, de 14 de maio, a qual veio a ser substituída pela Diretiva 2009/24/CE, de 23 de abril. Estas solução encontra-se igualmente prevista no art. 10º do Acordo TRIPS e no art. 4º do *WIPO Copyrigh Treaty* de 1996, sendo hoje um padrão mundial. Para o contexto histórico da Diretiva veja-se, por todos, Michel Walter/Silke Von Lewinsky, *European Copyright Law: A Commentary*, Oxford Unviersity Press, Oxford, 2010, pp. 89 e ss. e, sobre os tratados internacionais, José Alberto Vieira, *op. cit.*, pp. 191-210.

respectivas interfaces, não são protegidos pelos direitos de autor ao abrigo da presente directiva"[55]. A jurisprudência europeia já esclareceu que essa proteção abrange apenas o código-fonte e o código-objeto, únicas formas de expressão de um programa de computador, não se estendendo a outros elementos[56].

Em 2011, o TJUE foi confrontado com a questão de saber se uma interface gráfica poderia ser protegida por via de direitos de autor[57]. Tendo negado a proteção por via do direito de autor "especial" para programas de computador (visto que a interface gráfica não constitui a expressão de um programa de computador) o Tribunal afirmou que a proteção por via do direito de autor "comum" poderia ser concedida desde que essa obra fosse original[58]. Referindo-se então à determinação da originalidade afirmou que "este critério não pode ser preenchido pelos componentes da interface gráfica do utilizador que se caracterizam unicamente pela sua função técnica"[59] visto que "quando a expressão dos referidos componentes resulta da sua função técnica, o critério da originalidade não se encontra preenchido, porque as diferentes formas de executar uma ideia são tão limitadas que a ideia e a expressão se confundem"[60] e assim "...os componentes da interface gráfica do utilizador não permitem ao autor exprimir o seu espírito criador de modo original e chegar a um resultado que constitua uma criação intelectual desse autor"[61]. Por outras palavras, a interface gráfica não constitui, para este efeito, *software* e apenas poderá ser protegida quando a sua conceção não seja determinada apenas por fatores técnicos ou funcionais[62].

[55] *Vide* também os considerandos 10 e 11. Curiosamente a transposição portuguesa não refere expressamente interfaces, mas antes "... a liberdade das ideias e dos princípios que estão na base de qualquer elemento do programa ou da sua interoperabilidade" (art. 2º/2 do DL 252/94).

[56] Com exceção do material de conceção referido no art. 1º/1 da Diretiva. Sobre este veja-se José Alberto Vieira, *op. cit.*, pp.44-46 e 301-324.

[57] No contexto de um litígio despoletado por uma tentativa infrutífera de estabelecer uma sociedade de gestão coletiva de direito de autor sobre *software* na República Checa. Sobre as interfaces gráficas em detalhe cfr. José Alberto Vieira, *op. cit.*, pp. 497 e ss. e Noam Shemtov, *op. cit.*, pp 2-25.

[58] C-393/09, BSA (EU:C:2010:816), §42-46.

[59] C-393/09, *BSA*, §48

[60] C-393/09, *BSA*, §49. Isto expressa a já mencionada *merger doctrine*. Na mesma linha veja-se C-833/18, *Brompton Bicycle* (EU:C:2020:461), §27.

[61] C-393/09, *BSA*, §50.

[62] Esta conclusão parece ser congruente com a jurisprudência mais recente, como o acórdão C-833/18, *Brompton Bicycle*, §38: "... a proteção (...) ao abrigo do direito de autor aplica[-se] a um produto cuja forma é, pelo menos em parte, necessária à obtenção de um resultado técnico quando esse produto constitua uma obra original resultante de uma criação intelectual, na medida em que, através dessa forma, o seu autor exprime a sua capacidade criativa de modo original, efetuando escolhas livres e criativas que refletem na referida forma a sua personalidade, o que cabe ao órgão jurisdicional nacional verificar tendo em conta o conjunto dos elementos pertinentes do litígio no processo principal". Também assim Noam Shemtov, *op. cit.*, p. 8 "...the fact that an item is designed primarily in order to achieve a functional utilitarian objective should not render it non-eligible to copyright protection".

Na mesma linha, no acórdão de 2012, C-406/10, *SAS,* o Tribunal de Justiça concluiu[63]: "no que respeita aos elementos de um programa de computador (...) nem a funcionalidade de um programa nem a linguagem de programação e o formato de ficheiros de dados utilizados no âmbito de um programa de computador para explorar algumas das suas funções constituem uma forma de expressão desse programa"[64].

A jurisprudência portuguesa, no acórdão do Tribunal da Relação de Lisboa, de 6 de Abril de 2021[65], já decidiu também nesse sentido, concluindo: "o algoritmo não é um «bem» tutelável no domínio dos direitos de autor, estando excluído da proteção conferida aos programas de computador" visto que "o direito de autor não protege funcionalidades".

No direito europeu, a exclusão das interfaces do direito de autor para *software* parece resultar de norma expressa[66]. A proteção por direito de autor comum estará dependente da originalidade das interfaces, a qual tenderá a ser afastada em virtude da respetiva funcionalidade[67]. No entanto, como sublinha Richard Arnold[68], a questão da tutela jusautoral das APIs no Direito Europeu está longe de estar resolvida.

[63] C-406/10, *SAS* (EU:C:2012:259), §39.

[64] Admite-se, porém, a tutela jusautoral pelo direito de autor comum de algumas destas criações, como a linguagem de programação e o formato de ficheiros (C-406/10 *SAS*, §45), o manual de instruções (C-406/10, *SAS*, §64) e os elementos gráficos e sonoros de um videojogo (C-355/12, *Nintendo* (EU:C:2014:25), §23) desde que sejam originais. Sobre o conceito de obra veja-se Oehen Mendes, "A obra enquanto objecto do direito de autor", *RDI*, 1 (2021), pp. 41-73 e, especificamente sobre o formato de ficheiro, Pedro Dias Venâncio, *A Tutela Jurídica do Formato de Ficheiro Eletrónico*, Almedina, Coimbra, 2016 (propondo a criação de um regime *sui generis*. Na p. 337, depois de aludir a um "princípio geral de livre interoperabilidade", o Autor exclui o tratamento das interfaces).

[65] Proc. 55/19.4YHLSB.L1-PICRS, rel. Isoleta Costa.

[66] Christopher Millard, *op. cit.*, p. 165. Com uma posição intermédia Pamela Samuelson, "The Past, Present and Future of Software Copyright Interoperability Rules in the European Union and United States", *EIPR*, (2012), p. 232: "The Software Directive does not categorically exclude interfaces from the scope of copyright's protection, although it anticipates that interfaces may be among the unprotectable elements of programs as ideas or principles". Em sentido contrário, cfr. Opinião do AG Yves Bot no caso C-406/10, *SAS Institute* (EU:C:2011:787), §81: "Parece-me que a Diretiva 91/250 não exclui as interfaces da proteção pelos direitos de autor".

[67] Nese sentido Bently/Yin-Harn, in: Thomas Dreier/Bernt Hugenholtz, *Concise European Copyright Law*, Wolters Kluwer, Alphen aan den Rijn, 2016, p. 245. Foi também esse o entendimento dos tribunais ingleses nas decisões *SAS Institute Inc v World Programming Ltd* [2010] EWHC 1829 (Ch) de 23 de julho de 2010 e *SAS Institute Inc v World Programming Ltd* [2013] EWHC 69 (Ch), de 25 de janeiro de 2013. Para uma análise detalhada dessas decisões *vide* Richard Arnold, "Copyright in software: functionality", in: *Research...cit.*, pp. 33-43.

[68] *Op. cit.*, p. 41. Na verdade, como sublinha José Alberto Vieira, *op. cit.*, p. 374 este é um dos problemas mais debatidos de toda a temática da protecção dos programas de computador.

Assim, caso se considere que uma determinada API (ou a porção copiada dessa API) goza de direito de autor, e na ausência de uma cláusula geral como a de *fair use* não é líquido que exista fundamento de licitude para uma utilização não autorizada[69]. As previsões de estudo ("*black-box" analysis*) (art. 6º/1/b) do DL 252/94) e descompilação (art. 7º DL 252/94) têm condições bastante restritivas e não parecem aplicáveis a uma situação como a do litígio julgado pelo Supremo Norte-Americano[70].

Em contrapartida, a remissão para o art. 75º do CDADC feita no art. 10º do DL 252/94 permite defender que esta utilização constitui "a inclusão episódica de uma obra ou outro material protegido noutro material" (art. 75º/2/r) CDADC). E, nesse caso, a aplicação do "teste dos três passos" (art. 75º/4 CDADC) não seria muito diferente da análise feita em sede de *fair use*[71].

Poder-se-ia ainda, dependendo das circunstâncias do caso, ponderar a existência de uma licença implícita[72].

[69] Podemos também ter o resultado paradoxal de, na ausência de proteção jusautoral, não existirem limites à autonomia privada e os contratos de licença poderem ser mais restritivos do que se estivéssemos perante obras protegidas. Em Portugal, as normas que estabelecem utilizações livres são imperativas (art. 75º/5 CDADC), mas pressupõem que estejamos perante uma obra. Não é evidente se pode valer um argumento *a fortiori* para o domínio público e se devemos distinguir o domínio público originário (criações que nunca foram protegidas) do domínio público superveniente (obras cuja proteção caducou).

[70] Richard Arnold, *op. cit.*, pp. 31-32. Sobre a descompilação e interoperabilidade veja-se Alexandre Dias Pereira, *Informática, Direito de Autor e Propriedade Tecnodigital*, Coimbra Editora, Coimbra, 2001, pp. 641 e ss. e José Alberto Vieira, *op. cit.*, pp. 126-176. Deve sublinhar-se que, de um modo geral, a descompilação de programas modernos é uma quimera visto que envolveria tanto esforço que não faria sentido economicamente – Sally Weston, "Software interfaces – Stuck in the Middle: The Relationship Between the Law and Software Interfaces in Regulating and Encouraging Interoperability", *IIC*, 43 (2012), pp. 428-430.

[71] Assim Richard Arnold, *op. cit.*, p. 32. Tito Rendas, "As exceções no Direito de Autor da UE: Em busca do equilíbrio entre flexibilidade e certeza jurídica", *RDI*, 1 (2021), pp. 7-37 (e, com maior desenvolvimento, *Exceptions in EU Copyright Law: In Search of a Balance Between Flexibility and Certainty*, Wolters Kluwer, Alphen aan den Rijn, 2021) sublinhando a proximidade entre o teste dos três passos (reconfigurado na Diretiva Infosoc) e a previsão do §107 do Copyright Act e, também por isso, avançando com uma proposta de reforma do sistema de exceções a nível europeu, que passa pela consagração expressa de uma cláusula geral subsidiária.

[72] Sobre a figura *vide* Poorna Mysoor, *Implied Licences in Copyright Law*, Oxford University Press, Oxford, 2021, *passim*, esp. pp. 163-164 (aludindo a licenças implícitas de origem consuetudinária na indústria do *software*). Não trato aqui da questão da licença obrigatória e das obrigações de partilha de interfaces, especialmente quando impostas pelo Direito da Concorrência. Sobre isso veja-se Ashwin van Rooijen, *The Software Interface Between Copyright and Competition Law*, Wolters Kluwer, Alphen aan den Rijn, 2010.

6. Conclusão

Tanto o acórdão como o voto de vencido se revelam convincentes, o que, só por si, demonstra a dificuldade do tema. O Supremo Tribunal norte-americano preocupou-se essencialmente em salvaguardar a liberdade de reutilização daquele código sem, no entanto, se querer comprometer com uma posição definitiva. Uma análise em sede de *fair use* dá maior liberdade aos tribunais, mas, em contrapartida, gera maiores incertezas.

Poderia temer-se que esta decisão desincentivasse o investimento na criação e organização de APIs. No entanto, nas últimas décadas, apesar da convicção generalizada sobre a sua livre utilização, tem havido considerável atividade no desenvolvimento e partilha de APIs[73]. Por isso, esse receio será infundado.

A abordagem, aparentemente mais simples, de exclusão de tutela jusautoral de APIs atingiria um resultado equivalente à decisão norte-americana[74]. Porém, perante um litígio equivalente, um tribunal europeu terá que explicar em que é consiste uma API[75], considerar se esta integra a noção de *software* e de obra e em que medida é que a respetiva funcionalidade afastará ou não a sua proteção. Se a questão se puser perante o Tribunal de Justiça da União Europeia espera-se que este seja mais categórico que o *Supreme Court*, dissipando dúvidas e prevenindo futuros litígios[76]. Não parece desejável prolongar a incerteza, limitar a concorrência e dificultar a interoperabilidade entre sistemas.

[73] Nesse sentido Charles Duan, "A tale of two interoperabilities; or, how *Google v. Oracle* could become social media legislation", *Cardozo Law Review*, (2021), pp. 267-268.

[74] Sendo que o ónus da prova desequilibra a posição das partes – no sistema europeu será o autor que terá que estabelecer que a interface está protegida (o que parece difícil), no sistema americano é o réu que tem que provar os pressupostos do *fair use*.

[75] Veja-se *supra* nota 11 assinalando que a noção de API utilizada pelo tribunal norte-americano poderá ser demasiado lata.

[76] Nessa linha, elogiando a abordagem europeia, que coloca o debate ao nível da proteção jusautoral *vide* Jonathand Band, "The Global API Copyright Conflict", *Harvard Journal of Law & Technology*, 31 (2018), pp. 636-637. Segundo José Alberto Vieira, *op. cit.*, p. 380: "Tornar livre a expressão das interfaces significaria esvaziar a proteção outorgada aos programadas de computador, deixando à mercê de terceiros o trabalho valioso do(s) programador(es)". No entanto, este Autor entende que as especificações das interfaces e os protocolos de acesso não devem ser protegidos, sendo livremente apropriáveis (p. 444). Se compreendemos bem o seu pensamento, José Alberto Vieira admitiria a proteção do *implementing code*, mas não do *declaring code*, uma vez que define as especificações das interfaces como "a descrição de como dois dispositivos que têm funções diferentes podem comunicar entre si" e os protocolos de acesso como "a série de regras sintáticas e semânticas que determinam o comportamento de unidades funcionais na obtenção de comunicação" (p. 440).