Clase 8 — Contratos informáticos II. Licencias, recursos, SaaS, IaaS, PaaS y técnica profesional del programador.

1. Objetivos de aprendizaje

Al finalizar la clase, cada estudiante debería poder:

- Separar titularidad del software, licencia de uso, acceso a un servicio y entrega de código fuente.
- Identificar qué se contrata en SaaS, PaaS e IaaS y qué responsabilidades conserva el cliente.
- Convertir necesidades tecnológicas en condiciones contractuales verificables.
- Reconocer permisos y obligaciones básicos de MIT, Apache-2.0 y GPLv3.
- Detectar problemas de compatibilidad sin afirmar que cualquier componente abierto obliga a publicar todo el proyecto.
- Aplicar diligencia profesional, confidencialidad y prevención del daño a decisiones concretas.
- Elaborar una matriz de debida diligencia y justificar una decisión de despliegue.

2. Apertura — Que el código funcione no significa que pueda usarse de cualquier manera

Guion de inicio

«En la clase anterior vimos qué prometemos cuando desarrollamos o mantenemos software, cómo se mide el cumplimiento y cuándo una falla puede generar responsabilidad.

Hoy vamos a sumar otra pregunta. Imaginemos que el sistema funciona perfectamente: supera las pruebas, responde rápido y no se cae. ¿Eso alcanza para entregarlo?

No necesariamente. Podemos haber incluido una biblioteca sin cumplir su licencia. Podemos haber contratado un servicio que no permite el uso que le vamos a dar. Podemos haber guardado los datos del cliente en una cuenta personal. O podemos haber prometido entregar la propiedad exclusiva de un código que contiene componentes de terceros.

En todos esos casos, el problema no empieza con un bug. Empieza con una decisión técnica que no tuvo revisión jurídica».

Caso disparador

Un equipo recibe el encargo de desarrollar una plataforma comercial. Para acelerar:

- copia código de un repositorio público sin revisar la licencia;
- incorpora una biblioteca GPLv3;
- contrata una máquina virtual y supone que el proveedor hará todos los backups;
- usa una cuenta personal para administrar el servicio;

- promete al cliente «propiedad exclusiva sobre absolutamente todo»;
- sube una copia de la base real a un servicio de IA para resolver un error.

Pregunta al curso: ¿cuáles de estas decisiones pueden causar problemas aunque la aplicación funcione?

Respuesta orientadora: todas requieren revisión, pero por motivos diferentes. Hay que separar derechos sobre el código, obligaciones contractuales, seguridad, control de cuentas y tratamiento de datos. Tampoco puede concluirse, solo por mencionar GPL, que necesariamente deba publicarse todo el producto: faltan arquitectura, versión de licencia y modalidad de entrega.

Idea central

La calidad profesional incluye funcionar correctamente, tener derecho a utilizar los componentes, cumplir lo prometido, proteger información y poder demostrar las decisiones.

---

3. Recursos informáticos: no contratamos solamente computadoras

Un recurso informático es un elemento material, lógico, de conectividad o de servicio necesario para construir, ejecutar, proteger o mantener una solución.

Puede tratarse de hardware, almacenamiento, redes, licencias, capacidad de cómputo, bases de datos administradas, APIs, dominios, certificados, herramientas de monitoreo o servicios de respaldo. Las horas de soporte y los conocimientos especializados también forman parte de los recursos que un proyecto necesita contratar u organizar.

La palabra «recurso» es útil para planificar, pero no define por sí sola un tipo de contrato. Comprar un servidor, licenciar una herramienta y contratar procesamiento en la nube son negocios distintos.

Antes de avanzar: qué es cada recurso y cómo se utiliza

Hardware

Es el conjunto de componentes físicos de un sistema informático: computadoras, servidores, discos, dispositivos de red y otros equipos.

Ejemplo: un consultorio compra una computadora para recepción y un dispositivo para almacenar copias de seguridad. Contrata bienes físicos; eso no incluye necesariamente el programa de turnos, su licencia ni el mantenimiento.

Para explicar oralmente: «El hardware es la parte material. Pero comprar la computadora no significa comprar todos los programas que después vamos a instalar».

Software

Es el conjunto de programas e instrucciones que permiten realizar tareas. Incluye sistemas operativos, aplicaciones y otras herramientas.

Ejemplo: el programa que registra pacientes, asigna horarios y envía recordatorios es software. Puede utilizarse mediante una licencia instalada localmente o como un servicio online, según la solución contratada.

Servidor

Es un sistema que proporciona recursos o servicios a otros sistemas, llamados clientes. Puede referirse al equipo que los ejecuta o al programa que atiende solicitudes.

Ejemplo: cuando un paciente consulta los horarios disponibles desde su teléfono, un servidor recibe la solicitud y devuelve los turnos.

Punto jurídico: hay que establecer quién administra ese servicio, quién controla los accesos y qué disponibilidad se prometió.

Almacenamiento y capacidad de cómputo

El almacenamiento conserva información. La capacidad de cómputo permite ejecutar operaciones mediante recursos como procesador y memoria.

Ejemplo: guardar diez mil estudios requiere espacio; procesarlos para generar un informe requiere capacidad de cómputo. Contratar mucho almacenamiento no garantiza que el sistema responda rápido.

Redes y conectividad

Son los medios y reglas que permiten comunicar dispositivos y servicios.

Ejemplo: la computadora de recepción se conecta al sistema remoto mediante internet. Si esa conexión falla, la aplicación puede quedar inaccesible aunque su servidor funcione.

La consecuencia contractual depende de qué parte debía asegurar cada conexión y de las alternativas comprometidas.

Base de datos y base de datos administrada

Una base de datos es una colección organizada de información. Un sistema gestor permite almacenarla, consultarla y modificarla.

En una base de datos administrada, un proveedor asume determinadas tareas de operación del gestor. Su alcance depende del servicio: no debe suponerse que incluye cualquier respaldo, recuperación o protección.

Ejemplo: una base guarda pacientes, profesionales y turnos. El proveedor puede actualizar el motor, pero el equipo que desarrolla la aplicación sigue debiendo impedir que un paciente consulte los datos de otro.

API — Interfaz de programación de aplicaciones

Es un conjunto de reglas e interfaces mediante las cuales un programa utiliza funciones o datos de otro.

Ejemplo: una aplicación de turnos llama a una API de mensajería para enviar un recordatorio. La aplicación no construye toda la infraestructura de mensajería: utiliza una prestación externa.

Hay que revisar precio por solicitud, límites, permisos, disponibilidad y datos que se transmiten. Tener una clave de API no habilita cualquier finalidad.

Dominio

Es un nombre dentro del sistema de nombres de dominio que facilita localizar un servicio.

Ejemplo ficticio: turnos.ejemplo identifica el acceso de una aplicación. Registrar el dominio no equivale a obtener una marca ni a contratar automáticamente alojamiento.

Conviene definir quién figura como registrante, quién puede renovarlo y quién conserva el control al terminar el proyecto.

Certificado digital para TLS

Es un documento electrónico que vincula una identidad, como un nombre de dominio, con una clave pública bajo un sistema de confianza. En una conexión HTTPS participa en la autenticación del servidor y el establecimiento de una comunicación cifrada.

Ejemplo: el navegador comprueba el certificado del sitio antes de establecer la conexión segura.

No demuestra que la empresa sea honesta ni que toda la aplicación sea segura. Tampoco debe confundirse con el certificado utilizado para firmar digitalmente un documento conforme a un régimen legal específico.

Monitoreo y logs

El monitoreo observa indicadores del funcionamiento del sistema. Los logs son registros de eventos.

Ejemplo: el monitoreo detecta que aumentaron los errores al reservar; los logs permiten investigar qué solicitudes fallaron y cuándo.

No son lo mismo: una alerta informa un problema; un registro ayuda a reconstruirlo. Deben protegerse los registros y evitar contraseñas u otros datos innecesarios.

Backup y restauración

Un backup es una copia de información destinada a permitir su recuperación. Restaurar es utilizarla para recuperar datos o funcionamiento.

Ejemplo: cada noche se copia la base. Una mañana se elimina información por error y debe recuperarse la copia correcta.

Tener un archivo de respaldo no demuestra que pueda restaurarse: hay que probarlo. Una sincronización que replica de inmediato un borrado tampoco equivale necesariamente a un respaldo recuperable.

Máquina virtual

Es un entorno informático virtualizado que permite ejecutar un sistema operativo y aplicaciones utilizando recursos de una infraestructura física.

Ejemplo: se contrata una máquina virtual con memoria, procesador y disco, y el equipo instala allí el sistema de turnos.

El proveedor aporta recursos; las tareas de administración incluidas deben verificarse. No equivale a contratar una aplicación terminada.

Contenedor e imagen de contenedor

Un contenedor permite ejecutar una aplicación en un entorno aislado a nivel del sistema operativo. La imagen es el paquete utilizado para crear ese entorno, con la aplicación y los elementos definidos para ejecutarla.

Ejemplo: el equipo entrega una imagen con el programa y sus dependencias para que el cliente lo ejecute.

Entregar una imagen también puede entregar copias de componentes de terceros: empaquetarlos juntos no elimina sus licencias.

Soporte y mantenimiento

El soporte asiste en consultas o incidentes. El mantenimiento corrige, adapta o evoluciona la solución según lo contratado.

Ejemplo: explicar cómo dar de alta un profesional es soporte; corregir un error que duplica turnos es mantenimiento correctivo; agregar una función de videollamada puede ser evolución.

El nombre del abono no resuelve qué se incluye: deben definirse prestaciones, horarios y tiempos.

3.1. Capacidad técnica y habilitación jurídica

Una API puede permitir técnicamente hacer miles de solicitudes. Su contrato puede limitar su frecuencia, su finalidad o la posibilidad de almacenar resultados.

Una imagen de contenedor puede descargarse sin pagar. Eso no demuestra que todos los componentes incluidos permitan redistribución sin condiciones.

Un servicio puede admitir datos personales, pero imponer restricciones sobre datos especialmente protegidos o requerir condiciones adicionales.

Debemos formular dos preguntas distintas:

- ¿El recurso puede hacer lo que necesitamos?
- ¿Tenemos derecho a usarlo así y bajo qué condiciones?

Ejemplo oral: «Que una llave abra una puerta no demuestra que tengamos autorización para entrar. En software, una capacidad técnica tampoco demuestra por sí sola un permiso jurídico».

3.2. El costo real del recurso

El precio publicado no siempre representa el costo total. Deben revisarse almacenamiento, solicitudes, transferencia de salida, soporte, copias de respaldo, licencias asociadas y migración.

Ejemplo: un servicio resulta barato mientras se guardan pocos archivos. El proyecto crece y la transferencia hacia otro proveedor se vuelve costosa. No basta comparar el abono: hay que estudiar el ciclo de vida y la salida.

También hay que identificar quién contrata, quién paga y quién controla. Si la cuenta está a nombre de una persona del equipo, el cliente puede carecer de facultades suficientes para administrar el servicio cuando termine la relación. Debe pactarse la titularidad o el control operativo adecuado y la transición, respetando las condiciones del proveedor.

3.3. El pliego de condiciones técnicas

Un pliego técnico es el documento que describe qué se requiere, bajo qué condiciones y cómo se verificará. En contratación pública integra un procedimiento sometido a reglas específicas. En contratación privada puede utilizarse un documento equivalente como anexo técnico o solicitud de propuesta.

No debe confundirse:

- Necesidad: «necesitamos mantener disponible la información».
- Requisito: «debe existir un respaldo con determinada frecuencia y retención».
- Criterio de aceptación: «se realizará una restauración de prueba que permita verificar los datos y el tiempo comprometido».
- Obligación contractual: quién debe hacerlo, desde cuándo y con qué consecuencias ante incumplimiento.

Una cifra sin método de comprobación puede generar una apariencia de precisión.

3.4. Contenido mínimo de un pliego

Debe identificar objeto, alcance, entorno, capacidad, interoperabilidad, seguridad, niveles de servicio, soporte, derechos de uso, precio, aceptación y finalización.

Ejemplo de requisito insuficiente: «El servicio tendrá máxima seguridad».

Versión operativa: «Las cuentas administrativas deberán utilizar autenticación multifactor; los accesos se asignarán por rol; se registrarán altas, bajas y operaciones privilegiadas; y la verificación se realizará mediante los casos de prueba del anexo».

Esos controles son un ejemplo: no constituyen por sí solos una certificación de cumplimiento de todas las obligaciones legales.

También conviene definir:

- quién actualiza el sistema operativo y las dependencias;
- quién gestiona claves y certificados;
- qué datos pueden alojarse y en qué condiciones;
- cómo se notifica un incidente;
- cómo se exporta la información al terminar;
- qué evidencia se entrega para verificar cumplimiento.

Actividad oral breve

«El proveedor ofrece almacenamiento ilimitado». ¿Qué preguntarían?

Guía: límites de uso razonable, tipos de archivo admitidos, velocidad, recuperación, disponibilidad, costo de exportación, suspensión, retención y eliminación. «Ilimitado» es una expresión comercial que necesita lectura contractual.

---

4. SaaS, PaaS e IaaS: qué cambia al contratar cada capa

Estas categorías describen modelos de servicio. No son, por sí solas, tipos legales argentinos con un régimen contractual completo y autónomo.

La referencia técnica utilizada es NIST SP 800-145, de Peter Mell y Timothy Grance, publicada en septiembre de 2011. Su clasificación ayuda a comprender el reparto técnico de control; no sustituye la lectura del contrato ni tiene, por sí sola, fuerza de ley argentina.

Un mismo proyecto permite entender los tres modelos

La nube no es un lugar abstracto: se utilizan recursos de equipos e infraestructuras accesibles por red. El servicio contratado determina qué puede usar o administrar el cliente.

Imaginemos que una clínica necesita una aplicación de turnos. Puede resolverlo de tres maneras:

Opción A: utilizar una aplicación terminada — SaaS

La clínica contrata un sistema online existente, crea sus profesionales, configura horarios y comienza a cargar turnos.

No desarrolla la aplicación ni administra normalmente su sistema operativo. Utiliza las funciones que ofrece el proveedor.

Ejemplo de contratación: «Acceso al sistema de turnos para cinco usuarios, con las funciones del plan acordado».

En palabras simples: «Contrato un programa listo para usar».

No debe confundirse personalización con control total: poder cambiar un logo o un horario no convierte a la clínica en dueña del código.

Opción B: desplegar una aplicación propia en una plataforma — PaaS

La clínica contrata a un equipo para programar una solución específica. El equipo despliega ese código en una plataforma que proporciona el entorno de ejecución y gestiona determinadas capas técnicas.

Ejemplo de contratación: «Plataforma para ejecutar la aplicación desarrollada por el equipo, bajo las versiones y límites admitidos».

En palabras simples: «Llevo mi programa a un entorno preparado para ejecutarlo».

El equipo mantiene responsabilidad sobre su código y su configuración. La plataforma no corrige automáticamente un error que permite a un paciente ver turnos ajenos.

Opción C: contratar recursos y administrar el entorno — IaaS

El equipo contrata una máquina virtual, instala el sistema operativo, configura el gestor de base de datos y pone en marcha la aplicación.

Ejemplo de contratación: «Recursos virtuales de procesamiento, memoria, almacenamiento y red».

En palabras simples: «Contrato capacidad informática y organizo gran parte de lo que corre sobre ella».

Ese mayor control suele exigir más tareas de administración. No debe suponerse que el precio incluye actualizaciones o recuperación de la base.

Analogía para explicarlo oralmente

SaaS se parece a utilizar un local ya equipado y operado para una actividad concreta. PaaS, a recibir un espacio con instalaciones preparadas donde desarrollar nuestra actividad. IaaS, a contratar infraestructura sobre la cual debemos organizar más elementos.

La analogía ayuda a recordar el grado de gestión, pero no reemplaza el contrato. En los tres modelos pueden existir configuraciones, servicios adicionales y repartos diferentes.

4.1. SaaS — Software as a Service

En SaaS se utiliza una aplicación proporcionada como servicio, normalmente mediante navegador o cliente. El proveedor administra la solución y su infraestructura, mientras que el cliente utiliza las funciones y configuraciones habilitadas.

Ejemplos: un sistema online de facturación, una herramienta colaborativa o una aplicación de gestión de turnos.

Guion oral: «No estoy comprando necesariamente una copia del programa para instalarla donde quiera. Estoy contratando la posibilidad de usar determinadas funciones, con ciertos límites, durante una relación de servicio».

Por eso debe analizarse:

- cantidad y tipo de usuarios;
- funciones y límites del plan;
- disponibilidad y soporte;
- cambios de características o de precio;
- administración de accesos;
- uso permitido de los datos;
- integraciones;
- suspensión y cancelación;
- exportación y eliminación de información.

Pagar una suscripción no transfiere automáticamente el código fuente ni los derechos patrimoniales del programa.

Tampoco elimina responsabilidades del cliente. Una aplicación segura puede quedar expuesta si el cliente comparte una cuenta administrativa, asigna permisos excesivos o publica un enlace que debía permanecer restringido.

4.2. PaaS — Platform as a Service

En PaaS se contrata una plataforma donde desplegar aplicaciones propias o adquiridas. El proveedor administra infraestructura y capas de plataforma; el cliente controla su aplicación y determinadas configuraciones.

Ejemplo: un entorno administrado que permite desplegar una aplicación mediante un repositorio o una imagen.

La ventaja es reducir tareas de infraestructura. El riesgo es suponer que, por ese motivo, también se delegó la seguridad del código.

El proveedor puede mantener el entorno de ejecución, pero el equipo sigue debiendo analizar autenticación de la aplicación, validación de entradas, permisos, dependencias y uso de secretos.

Preguntas contractuales: ¿qué versiones se admiten?, ¿cómo se anuncian retiros de soporte?, ¿qué límites de ejecución existen?, ¿cómo se accede a logs?, ¿qué sucede con tareas en segundo plano?, ¿cómo se migra?

4.3. IaaS — Infrastructure as a Service

En IaaS se contratan recursos de procesamiento, almacenamiento y red. El cliente administra normalmente el sistema operativo, sus aplicaciones y numerosas configuraciones.

Ejemplo: una máquina virtual donde el equipo instala Linux, una base de datos y su aplicación.

Contratar esa máquina no equivale a contratar automáticamente actualización del sistema operativo, endurecimiento de seguridad, backups, restauración o administración de la base. Pueden existir servicios adicionales que modifiquen ese reparto, pero deben identificarse.

Guion oral: «Alquilar capacidad de cómputo no implica contratar a alguien para que administre todo lo que ponemos adentro».

4.4. Responsabilidad compartida: punto de partida, no eximente automática

En términos generales, al pasar de IaaS a SaaS el cliente gestiona menos capas técnicas. Sin embargo, los límites exactos dependen del servicio y del contrato.

Cuestión	Pregunta que debe resolverse
Infraestructura	¿Quién mantiene hardware, red y virtualización?
Plataforma	¿Quién actualiza sistema operativo y entorno de ejecución?
Aplicación	¿Quién corrige el código y sus dependencias?
Identidades	¿Quién da de alta, revisa y revoca permisos?
Datos	¿Quién define finalidades, conservación y accesos?
Recuperación	¿Quién respalda, prueba y restaura?

Una matriz de responsabilidades describe tareas; no puede, por sí sola, eliminar derechos de terceros ni validar cláusulas abusivas.

Si intervienen datos personales, también deben analizarse las funciones efectivas de cada participante, la Ley 25.326 y las condiciones del tratamiento por cuenta de terceros. Si hay transferencias internacionales, no alcanza con

saber el país donde dice estar el servidor: hay que examinar accesos, subproveedores y régimen aplicable.

4.5. Portabilidad, reversibilidad y salida

Portabilidad es la posibilidad de extraer y reutilizar información. Reversibilidad es la capacidad de trasladar o recuperar la operación al terminar la relación.

Un archivo descargable no demuestra que la migración sea viable. Pueden faltar adjuntos, relaciones, metadatos o documentación del formato.

Debe definirse plazo de exportación, formato, alcance, costo, asistencia, período de acceso posterior y tratamiento de respaldos. La eliminación debe coordinarse con deberes de conservación y obligaciones legales: no se promete borrar todo inmediatamente si existe una obligación válida de preservar determinada evidencia.

Pregunta para el curso: si una aplicación permite exportar contactos, pero no historial ni archivos, ¿el cliente puede continuar realmente su actividad con otro proveedor?

---

5. Licencia, titularidad y entrega: tres cuestiones diferentes

La licencia es una autorización para ejercer determinados usos sobre el software, dentro de ciertas condiciones. La titularidad identifica a quien corresponde el derecho. La entrega de archivos es un hecho material o contractual que no resuelve, por sí solo, las otras dos cuestiones.

Una persona puede tener:

- permiso para usar un programa sin poder redistribuirlo;

- acceso al código sin permiso para comercializarlo;
- una copia del ejecutable sin derecho a modificarlo;
- una licencia abierta que permite uso comercial, pero exige conservar avisos;
- una cesión de determinados derechos sobre código propio, con exclusión de componentes de terceros.

Ejemplos para distinguir los conceptos

Titularidad

Es la pertenencia de los derechos sobre una obra o de determinados derechos sobre ella. No siempre coincide con quien tiene una copia del archivo.

Ejemplo: una empresa puede ser titular de derechos patrimoniales sobre un programa y permitir que muchos clientes lo usen. Los clientes no se convierten por eso en titulares de esos mismos derechos.

Licencia de uso

Es el permiso para utilizar el software en las condiciones concedidas.

Ejemplo hipotético: una clínica puede usar el programa en diez puestos, pero no revender copias. Tiene una autorización delimitada, no necesariamente todos los derechos de explotación.

Cesión de derechos

Es la transferencia de determinados derechos, con el alcance y las condiciones legalmente admisibles.

Ejemplo: el contrato puede prever la transferencia de derechos patrimoniales sobre módulos creados específicamente para el cliente. Debe individualizarse lo

cedido y excluirse lo que el proveedor no puede transferir, como derechos ajenos sobre bibliotecas.

Código fuente

Es la forma del programa que se utiliza para realizar modificaciones, normalmente escrita en un lenguaje de programación.

Ejemplo: los archivos donde se define cómo validar y guardar un turno.

Código objeto y ejecutable

El código objeto es una forma resultante de la transformación del código fuente; un ejecutable es un archivo o programa preparado para ejecutarse en determinado entorno. No todo código objeto es, por sí solo, un ejecutable completo, y no todo programa requiere una compilación tradicional.

Ejemplo: el cliente recibe una aplicación compilada que puede ejecutar, pero no necesariamente recibe los archivos fuente ni autorización para modificarlos.

Repositorio y dependencia

Un repositorio organiza archivos y, habitualmente, su historial de versiones. Una dependencia es un componente que el proyecto necesita para construir o ejecutar determinadas funciones.

Ejemplo: el equipo guarda su aplicación en un repositorio y utiliza una biblioteca externa para generar PDFs.

Que el repositorio sea público no elimina los derechos sobre su contenido. Que la biblioteca se instale automáticamente tampoco elimina sus condiciones de licencia.

5.1. Base normativa argentina

La Ley 11.723, modificada por la Ley 25.036, incluye programas fuente y objeto dentro de las obras protegidas. El art. 1 distingue expresión protegida de ideas, procedimientos, métodos de operación y conceptos matemáticos en sí mismos.

El art. 55 bis contempla los contratos de licencia para uso o reproducción entre las formas de explotación del software.

El art. 4, inc. d, contiene una regla específica para programas producidos por dependientes contratados para elaborarlos, en el desempeño de sus funciones laborales, salvo estipulación en contrario. No corresponde trasladarla automáticamente a cualquier contratación independiente.

El art. 9 contiene una autorización específica para una copia de salvaguardia bajo sus condiciones. No debe transformarse esa regla en una autorización general para copiar, compartir o explotar comercialmente.

5.2. «Pagamos el desarrollo, entonces todo es nuestro»

La frase es insuficiente. Debe analizarse el contrato, el régimen aplicable y la cadena de derechos.

Ejemplo: el equipo desarrolla módulos propios, reutiliza una herramienta preexistente y agrega bibliotecas abiertas. Puede pactar ciertos derechos sobre sus aportes, pero no atribuir exclusividad sobre derechos que pertenecen a terceros.

Por eso un contrato debería distinguir:

- desarrollos específicos;
- materiales preexistentes;
- componentes de terceros;
- documentación y diseños;
- derechos concedidos y reservados;

- acceso y entrega de código;
- facultades para modificar, mantener y migrar.

Una lista de componentes no es un formalismo: permite evitar promesas incompatibles con derechos ajenos.

---

6. Software privativo, abierto y comercial

Qué significa cada categoría

Privativo o propietario

En esta clasificación docente, describe software cuyo titular reserva usos que no concede libremente al usuario, por ejemplo modificación o redistribución.

Ejemplo hipotético: un programa de diseño permite instalarse en dos equipos, pero no compartir su instalador y licencia con toda una empresa.

La restricción exacta surge de sus términos: «privativo» no significa que todas las licencias sean idénticas.

Software libre y de código abierto

Son enfoques relacionados con las libertades y permisos sobre el software, como uso, estudio, modificación y redistribución. Tienen fundamentos y criterios propios, aunque numerosas licencias pertenecen a ambas categorías.

Ejemplo: un programa se distribuye con una licencia que permite estudiarlo, adaptarlo y compartirlo si se cumplen sus condiciones.

No significa «sin autor», «sin copyright» ni «obligatoriamente gratis».

Licencia permisiva

Concede amplias facultades con condiciones relativamente limitadas y sin exigir, como regla general, que toda obra derivada se distribuya bajo la misma licencia. MIT y Apache-2.0 son los ejemplos trabajados aquí.

Ejemplo: una empresa integra una biblioteca permisiva a su producto cerrado, conserva los avisos exigidos y cumple las demás condiciones aplicables.

Copyleft

Es una técnica de licenciamiento que exige mantener determinadas libertades y condiciones en la distribución de la obra cubierta y de las modificaciones u obras alcanzadas por sus términos.

Ejemplo: se modifica un programa GPLv3 y se entrega esa versión. Deben examinarse las obligaciones de licenciamiento y fuente correspondiente de la obra cubierta.

No significa que cualquier programa que conviva con él deba abrirse automáticamente.

Comercial y gratuito

«Comercial» describe una explotación económica; «gratuito» describe ausencia de precio en una prestación concreta.

Ejemplo: puede cobrarse por distribuir software GPL y dar soporte. También puede ofrecerse gratis una aplicación privativa que no permite modificar ni redistribuir.

Por eso «abierto o comercial» no es una oposición correcta.

6.1. Licencias privativas

Restringen los usos a lo autorizado por su titular. Pueden limitar usuarios, dispositivos, instalaciones, entornos, territorios, duración o finalidades.

No todas son iguales. Algunas permiten ciertos desarrollos o integraciones; otras requieren licencias adicionales para redistribuir componentes.

Una licencia perpetua no garantiza necesariamente actualizaciones, soporte indefinido ni continuidad de servicios remotos. Debe leerse qué derecho es permanente y qué prestaciones tienen duración limitada.

Ejemplo: una licencia académica puede no permitir el uso en producción comercial. Que funcione fuera del aula no amplía el permiso.

6.2. Software abierto

El software de código abierto no es software sin derechos. Su titular utiliza una licencia que concede permisos bajo determinadas condiciones.

Hay que distinguir:

- gratuito: referencia al precio;
- abierto: referencia a derechos y condiciones de la licencia;
- comercial: referencia a una actividad económica;
- código visible: posibilidad de lectura que no garantiza permisos de modificación o redistribución.

Un producto abierto puede venderse. Un programa gratuito puede ser privativo. Un repositorio público puede carecer de una licencia suficiente para el uso pretendido.

6.3. Sin licencia identificable

No debe presumirse libertad de uso por ausencia de un archivo LICENSE.

La respuesta profesional es identificar titularidad, términos y autorización. Puede ser necesario reemplazar el componente, obtener permiso o restringir su utilización hasta resolverlo.

Ni la cantidad de estrellas ni la popularidad de un repositorio sustituyen esa verificación.

---

7. MIT: permisiva, pero no carente de condiciones

MIT es el nombre de una licencia, no una clase de aplicación ni un servicio de nube. Sus siglas se asocian al Massachusetts Institute of Technology. Lo que concede permisos es el texto de licencia aplicado al componente.

Ejemplo paso a paso: un equipo desarrolla un programa de ventas y utiliza una biblioteca ficticia «Calendario Simple», licenciada bajo MIT, para mostrar fechas. El equipo vende su programa y no publica su código propio. Al entregar el producto, incluye el aviso de copyright y el permiso de la biblioteca como exige MIT.

La conclusión no es «el producto entero pasa a MIT», sino «debo respetar MIT respecto del componente y los usos alcanzados».

La licencia MIT concede amplios permisos de uso, copia, modificación, distribución, sublicencia y venta de copias.

Su condición central exige incluir el aviso de copyright y el aviso de permiso en todas las copias o porciones sustanciales del software.

Ejemplo aplicado

Un equipo incorpora una biblioteca MIT a una aplicación comercial. En principio, puede conservar cerrado su código propio, pero debe cumplir los avisos exigidos respecto del componente.

No basta escribir «usamos open source» en una presentación. Debe diseñarse la forma de distribuir los avisos, por ejemplo mediante documentación y archivos de licencias de terceros, según el paquete entregado.

Qué no debe enseñarse

No es correcto afirmar que MIT exige publicar todo el proyecto ni que no exige absolutamente nada.

Su cláusula de entrega «tal cual» expresa una exclusión de garantías y responsabilidad en la licencia. Su alcance debe examinarse conforme al derecho aplicable. Tampoco libera automáticamente al integrador frente al cliente de obligaciones que el propio integrador asumió.

Pregunta al curso: si el autor de la biblioteca no prometió soporte, pero nuestra empresa vendió mantenimiento, ¿podemos responder al cliente que nadie tiene obligación de corregir? No: son relaciones y prestaciones diferentes.

---

8. Apache-2.0: permisos, avisos y patentes

Apache-2.0 es el nombre y la versión de una licencia de software. No debe confundirse con un programa concreto ni suponerse que solo se utiliza en servidores.

Ejemplo paso a paso: un equipo incorpora un generador ficticio de informes bajo Apache-2.0, modifica dos archivos y lo entrega dentro de su sistema comercial. Debe acompañar la licencia, identificar los cambios y conservar los

avisos pertinentes; si corresponde por el NOTICE del componente, reproducir sus atribuciones de la forma admitida.

Un archivo NOTICE reúne determinados avisos de atribución. No es lo mismo que LICENSE, que contiene el texto de la licencia. Conservar uno no reemplaza automáticamente las obligaciones relativas al otro.

Apache-2.0 también es una licencia permisiva y admite utilización en productos comerciales.

Al redistribuir deben examinarse especialmente sus condiciones de la sección 4:

- entregar una copia de la licencia;
- identificar de forma destacada los archivos modificados;
- conservar los avisos pertinentes en los términos previstos;
- reproducir los avisos de atribución del archivo NOTICE si la obra lo incluye y corresponde.

No se exige inventar un NOTICE para todo componente que carece de él. Sí debe revisarse y cumplirse el que efectivamente integra la distribución.

8.1. Patentes

La sección 3 incorpora una licencia expresa de determinadas reivindicaciones de patentes de los contribuidores, con el alcance y la terminación previstos en su texto.

No significa una garantía de que el producto completo no infringe ninguna patente del mundo. La concesión tiene sujetos, contribuciones y límites definidos.

8.2. Marcas y garantías

La sección 6 no otorga una autorización general para usar las marcas del licenciante como identidad del producto.

Las secciones 7 a 9 tratan garantías, límites de responsabilidad y aceptación de obligaciones adicionales. Ofrecer soporte propio no convierte a los autores originales en responsables por lo que prometimos.

Ejemplo didáctico

Se modifica un archivo Apache-2.0 y se distribuye dentro de una solución cerrada. Deben revisarse avisos, cambios y eventual NOTICE. No corresponde concluir que toda la solución deba publicarse solo por ese componente, ni eliminar las referencias de autoría para presentar todo como creación exclusiva.

---

9. GPLv3: copyleft y condiciones de entrega

GPL significa General Public License —Licencia Pública General—. La «v3» identifica su versión 3. Es una licencia, no un programa: muchas obras diferentes pueden adoptarla.

Ejemplo paso a paso: un equipo toma un programa ficticio de reportes GPLv3, modifica su cálculo y entrega la versión modificada a un cliente. No puede tratar esa versión cubierta como si fuese exclusivamente privativa y prohibir todas las facultades que GPL exige conceder. Debe cumplir las condiciones aplicables sobre licencia, avisos y fuente correspondiente.

Ejemplo diferente: una persona ejecuta ese programa para su trabajo interno y no entrega copias a terceros. No surge, solo por ese uso, una obligación general de publicar todo su código.

Ambos ejemplos simplifican deliberadamente los hechos para mostrar por qué importa distinguir ejecutar, modificar y distribuir. En una integración real hay que determinar qué obra está alcanzada.

La GPLv3 es una licencia de software libre con copyleft. Concede permisos amplios, incluido el uso comercial, y busca mantener las libertades de los destinatarios respecto de las obras cubiertas.

9.1. Ejecutar no es lo mismo que entregar copias

La sección 0 diferencia la entrega que permite a terceros recibir copias de la mera interacción por red sin transferencia de copia.

La sección 2 permite ejecutar y realizar modificaciones privadas en las condiciones de la licencia. No existe una regla general que obligue a publicar toda modificación interna por el solo hecho de usar un programa GPLv3.

Pero si se entrega una copia a un cliente, no debe descartarse el análisis porque exista un único destinatario. También pueden existir entregas de JavaScript al navegador o de imágenes a terceros: no todo lo que se llama «SaaS» carece de distribución de componentes.

9.2. Código fuente correspondiente

Las secciones 5 y 6 regulan entrega de versiones modificadas y de formas no fuente. Según el supuesto, deben preservarse avisos, licenciarse la obra cubierta conforme a GPL y facilitarse el código fuente correspondiente mediante una modalidad admitida.

El código fuente correspondiente no se reduce necesariamente a los archivos editados. La sección 1 comprende elementos necesarios para generar, instalar, ejecutar y modificar la obra, con exclusiones determinadas.

No basta entregar un archivo incompleto y afirmar «acá está el fuente». Tampoco existe una exigencia universal de subir todo a una plataforma pública determinada: las obligaciones y modalidades deben leerse según el caso.

9.3. El error de la «contaminación automática»

No es riguroso afirmar que cualquier contacto con GPL obliga a abrir toda la empresa o todos sus repositorios.

La propia sección 5 contempla agregados de obras separadas e independientes que no quedan sometidas por esa mera inclusión a GPL.

El análisis requiere determinar:

- qué obra está cubierta;
- si hubo modificación o incorporación de expresión protegida;
- cómo interactúan los componentes;
- si forman una obra combinada o conservan independencia;
- qué copias se entregan;
- qué versión y excepciones rigen.

El enlazado estático, dinámico, la comunicación por procesos o por API pueden ser datos relevantes, pero no deben convertirse en fórmulas universales sin estudiar arquitectura, licencia y derecho aplicable.

9.4. GPL no es sinónimo de todas las licencias de la familia

GPLv2 y GPLv3 no deben tratarse como textos idénticos. También importan las designaciones «solo esta versión», «o posterior» y las excepciones adicionales.

LGPL y AGPL requieren lectura propia. En particular, no debe trasladarse automáticamente a GPLv3 una regla atribuida a AGPL sobre interacción por

red. Esta clase trabaja el texto verificado de GPLv3; cualquier decisión basada en otra licencia debe consultar su versión exacta.

9.5. Un negocio legítimo puede basarse en GPL

Puede cobrarse por copias y por soporte. El problema no es que el producto sea comercial: es que el modelo contractual pretenda quitar libertades que la licencia exige mantener para la obra cubierta.

Si el cliente quiere una distribución exclusivamente cerrada, puede ser necesario sustituir un componente, conseguir una licencia alternativa del titular o cambiar la propuesta. Rediseñar arquitectura no garantiza por sí solo que se eviten todas las obligaciones: debe verificarse el resultado.

---

10. Compatibilidad: una conclusión documentada, no una etiqueta

Compatibilidad significa que las condiciones aplicables pueden cumplirse simultáneamente en la operación concreta.

No basta preguntar «¿MIT combina con GPL?». Hay que precisar qué se integra, cómo se entrega y qué términos se pretenden imponer.

Un componente permisivo puede incorporarse en determinados proyectos copyleft si se cumplen sus condiciones y el régimen de la obra resultante. Eso no autoriza a retirar una obra cubierta por GPL y relicenciarla unilateralmente como privativa.

Debida diligencia y matriz: qué son y para qué sirven

La debida diligencia es una revisión sistemática y proporcionada para identificar información, obligaciones y riesgos antes de adoptar una decisión y durante su ejecución.

Ejemplo: antes de prometer un producto cerrado, el equipo identifica sus bibliotecas, revisa sus licencias y verifica qué copias entregará.

No es una garantía de ausencia de riesgos: es un proceso que permite detectarlos, evaluarlos y tratarlos.

Una matriz es una forma de organizar esa revisión por filas y columnas. No es un contrato ni una resolución judicial.

Ejemplo de una fila: «Componente sin licencia identificada; falta comprobar autorización; riesgo de uso no permitido; medida: obtener permiso suficiente o reemplazar; responsable: desarrollo con revisión jurídica; evidencia: autorización archivada o reemplazo verificado».

Un checklist es una lista de controles que deben verificarse.

Ejemplo: «¿Se revocaron las claves expuestas? ¿Se probó restauración? ¿Se incluyeron los avisos de terceros?».

La matriz explica el problema y su tratamiento. El checklist ayuda a comprobar que no se omitió un control. Marcar una casilla sin realizar la verificación no demuestra diligencia.

Método de revisión

1. Identificar el componente y su versión.
2. Guardar origen y texto de licencia de esa versión.
3. Revisar dependencias directas y transitivas.
4. Precisar si se modifica, copia, enlaza o ejecuta de manera independiente.
5. Identificar entregas a clientes, usuarios, dispositivos o navegadores.

6. Confrontar condiciones con el contrato comercial.
7. Registrar avisos, fuente u otras medidas exigibles.
8. Documentar incertidumbres y escalar lo que no pueda resolverse.

Una SBOM —lista de materiales de software— permite inventariar componentes. No constituye una opinión jurídica automática ni asegura que todos los datos de licencia sean correctos.

Ejemplo oral

«El escáner informa GPLv3. Eso es una alerta para investigar, no una sentencia. Pero ignorar la alerta porque estamos apurados tampoco es una respuesta profesional».

---

11. Técnica profesional: el estándar de diligencia

La técnica profesional comprende conocimientos, procedimientos y cuidados razonablemente exigibles para realizar la prestación en su contexto.

No consiste en garantizar perfección absoluta, usar siempre la tecnología más cara ni justificar cualquier error diciendo que el software es complejo.

Qué son diligencia, prudencia, pericia y previsión

Diligencia es actuar con el cuidado exigible para la tarea y sus circunstancias. Ejemplo: antes de migrar una base, realizar las verificaciones y respaldos razonables, en vez de ejecutar directamente sobre producción.

Prudencia es evaluar riesgos antes de actuar. Ejemplo: no habilitar una función de pagos que tiene un defecto crítico conocido sin resolver su tratamiento.

Pericia es contar con conocimientos y habilidades adecuados para la actividad. Ejemplo: quien administra accesos debe comprender cómo configurar permisos y verificar que las restricciones se cumplan.

Previsión es considerar consecuencias razonablemente anticipables. Ejemplo: si un servicio externo anuncia que retirará una versión de su API, debe evaluarse el impacto antes de que deje de funcionar.

Estos conceptos no son garantías de infalibilidad. Permiten preguntar qué habría sido razonable conocer, hacer, advertir o verificar en ese contexto.

Ejemplo integrado

Una clínica necesita trasladar turnos a un sistema nuevo. El equipo detecta que algunos identificadores están duplicados.

Actuar diligentemente no es borrar duplicados sin preguntar ni continuar en silencio. Es advertir el problema, analizar sus efectos, conservar respaldos, acordar un criterio de corrección y validar el resultado.

La medida concreta depende del encargo y del riesgo. El art. 1725 ayuda a valorar la conducta; no determina por sí solo una indemnización.

11.1. Artículo 1725 CCCN

Su primer párrafo establece:

> "Cuanto mayor sea el deber de obrar con prudencia y pleno conocimiento de las cosas, mayor es la diligencia exigible al agente y la valoración de la previsibilidad de las consecuencias."

El artículo también contempla la confianza especial y las condiciones particulares de las partes. Debe leerse completo: no permite atribuir

omnisciencia ni responsabilidad automática a cualquier persona con conocimientos informáticos.

Aplicación didáctica: quien se presenta como especialista en seguridad para una plataforma de pagos debe prever riesgos propios de esa tarea. Su actuación se evalúa en relación con la prestación, los conocimientos razonablemente exigibles y el contexto.

Tener título o matrícula no es el único dato relevante para valorar conocimientos y diligencia. Tampoco corresponde aplicar automáticamente todo el régimen de profesionales liberales a cualquier actividad de programación.

11.2. Artículo 1256 CCCN

El inc. a exige ejecutar el contrato conforme a lo pactado y a los conocimientos razonablemente requeridos al tiempo de realización por el arte, la ciencia y la técnica de la actividad.

El inc. b impone informar al comitente los aspectos esenciales del cumplimiento.

Por eso puede ser relevante:

- advertir que una dependencia dejó de mantenerse;
- informar que el plan contratado carece de recuperación suficiente;
- explicar una incompatibilidad de licencias;
- documentar que un requisito implica un riesgo de exposición;
- no presentar un prototipo como una solución validada para producción.

La información debe ser comprensible y útil para decidir. Un mensaje «hay riesgos» puede resultar insuficiente si no identifica el problema y su impacto.

11.3. Instrucciones del cliente

Si el cliente ordena «publicalo igual», no queda resuelto todo el análisis. Hay que estudiar licitud, riesgo, obligaciones y capacidad efectiva de prevención.

Documentar una advertencia es importante, pero no convierte en válida una conducta prohibida ni elimina derechos de terceros. Ante riesgos graves puede ser necesario no ejecutar una operación hasta corregirla o adoptar una alternativa segura.

No hay que confundir desacuerdo técnico con poder para decidir por todo el proyecto. La respuesta profesional exige identificar atribuciones, responsables y canales de escalamiento.

---

12. Confidencialidad, secreto y NDA

Un NDA —Non-Disclosure Agreement— es un acuerdo de confidencialidad. Puede proteger código, documentación, estrategias, credenciales o información comercial.

No toda obligación de confidencialidad depende de firmar un NDA. Puede surgir de la ley, del contrato y de deberes de buena fe.

Diferencia entre confidencialidad, secreto profesional y NDA

Confidencialidad es el deber de no utilizar ni revelar cierta información fuera de los fines y destinatarios autorizados.

Ejemplo: el equipo conoce el listado de clientes de una empresa para migrarlo. No puede reutilizarlo para ofrecer servicios propios sin una habilitación suficiente.

Secreto profesional es una obligación vinculada al conocimiento de información en el ejercicio de determinadas funciones o actividades y a su régimen legal. No toda reserva comercial es idéntica al secreto profesional de cualquier profesión.

Ejemplo pertinente a esta clase: el art. 10 de la Ley 25.326 utiliza expresamente esa categoría para quienes intervienen en el tratamiento de datos personales, según sus presupuestos.

El NDA es el acuerdo que organiza contractualmente la confidencialidad.

Ejemplo: antes de mostrar el código de una aplicación no publicada, la empresa y el equipo delimitan qué información podrá utilizarse para evaluarla, quién podrá acceder y cómo deberá devolverse o eliminarse.

En palabras simples: «La confidencialidad es el deber; el NDA es una herramienta para pactarlo y precisarlo; y algunas normas imponen deberes de secreto aunque no exista ese documento».

12.1. Datos personales: una obligación legal expresa

El art. 10 de la Ley 25.326 obliga al secreto profesional al responsable y a quienes intervengan en cualquier fase del tratamiento de datos personales. La obligación subsiste después de terminar la relación con el titular del archivo.

Esto permite explicar por qué un integrante técnico no puede sostener que «como no firmé NDA, puedo compartir la base». Debe verificarse la intervención concreta y las normas aplicables, pero la ausencia del documento no equivale a ausencia de deber.

12.2. Qué debería definir un NDA

Información protegida, finalidad autorizada, destinatarios permitidos, cuidados exigidos, duración, exclusiones justificadas y procedimiento frente a revelaciones legalmente exigidas.

También devolución o eliminación, tratamiento de copias y respaldos, y medidas ante un incidente.

Un NDA no transfiere automáticamente propiedad intelectual, no autoriza tratamiento ilícito de datos y no crea una inmunidad frente a órdenes judiciales. La penalidad que pueda prever debe analizarse según su régimen: no toda infracción produce automáticamente cualquier suma pretendida.

12.3. Ejemplos para programadores

- Publicar un repositorio con claves: puede comprometer confidencialidad y seguridad.
- Compartir una captura con datos de usuarios: también es una divulgación.
- Utilizar datos reales del cliente en una demostración: exige justificación y autorización adecuadas.
- Enviar código o logs a un servicio de IA: requiere revisar contenido, autorización, términos, conservación y acceso de terceros.

Cambiar un nombre por un identificador no garantiza anonimización. Si puede reidentificarse a la persona, puede seguir habiendo datos personales.

12.4. Dar acceso no equivale a permitir cualquier finalidad

Las credenciales se conceden para determinadas funciones. Que una cuenta tenga privilegios amplios no demuestra autorización para copiar información a un equipo personal ni reutilizarla en otro proyecto.

Debe aplicarse necesidad de acceso, privilegio mínimo y revocación al finalizar la relación.

---

13. Prevención del daño y desarrollo seguro

El art. 1710 CCCN establece deberes de evitar daño injustificado, adoptar de buena fe medidas razonables para evitarlo o disminuirlo y no agravar el daño ya producido, en cuanto dependa de la persona.

La expresión «en cuanto de ella dependa» es esencial: se analiza la esfera real de actuación. No convierte a todo integrante del equipo en garante universal, pero impide reducir la profesión a escribir código e ignorar riesgos que se pueden gestionar.

Qué significa prevenir y qué significa desarrollar de forma segura

Prevenir el daño es actuar razonablemente para evitar que ocurra o reducir su impacto dentro de la esfera de actuación propia.

Ejemplo: se detecta que una dirección web permite descargar historias clínicas sin autorización. La prevención no consiste solo en anotar el error para el próximo mes: exige evaluar y adoptar medidas de contención, corrección y comunicación según las responsabilidades y obligaciones aplicables.

Desarrollo seguro es incorporar cuidados de seguridad al diseño, programación, pruebas, despliegue y mantenimiento, en vez de revisar todo únicamente al final.

Ejemplo: al diseñar una aplicación médica se define qué información puede ver cada rol, se implementan controles del lado del servidor, se prueban accesos indebidos y se revisa qué queda en los registros.

Autenticación significa comprobar una identidad o credencial. Autorización significa determinar qué operaciones puede realizar esa identidad.

Ejemplo: iniciar sesión demuestra que se superó un mecanismo de autenticación; no habilita a consultar los datos de cualquier paciente.

Producción es el entorno donde opera el servicio real. Despliegue es poner una versión en un entorno. Rollback es volver a una versión o estado anterior mediante un procedimiento previsto; no garantiza por sí solo recuperar datos alterados.

Ejemplo: una nueva versión falla y se revierte el programa. Si también modificó de manera incompatible la base, puede necesitarse un plan adicional de recuperación.

13.1. Seguridad proporcionada al riesgo

No se exige el mismo diseño para una maqueta sin datos reales y para un sistema sanitario. Deben considerarse finalidad, datos, usuarios, amenazas previsibles, obligaciones y consecuencias de una falla.

El art. 9 de la Ley 25.326 exige medidas técnicas y organizativas para seguridad y confidencialidad de datos personales, evitando adulteración, pérdida y tratamientos no autorizados.

Las prácticas técnicas permiten concretar esos deberes según el caso; una herramienta aislada no prueba cumplimiento integral.

13.2. Checklist previo al despliegue

- Identificar datos tratados y reducir los innecesarios.
- Revisar autenticación, roles y autorizaciones del lado del servidor.
- Proteger secretos y evitar su inclusión en repositorios.

- Analizar dependencias y vulnerabilidades relevantes.
- Validar entradas y consultas; no depender solo de controles de interfaz.
- Separar ambientes y evitar copias indiscriminadas de producción.
- Definir backups y probar restauración.
- Preservar logs útiles sin registrar contraseñas ni información excesiva.
- Preparar monitoreo, responsables y escalamiento.
- Verificar cambios, aprobación y plan de reversión.
- Revisar licencias y avisos del paquete efectivamente entregado.
- Documentar riesgos abiertos y su tratamiento.

13.3. Si ya ocurrió un incidente

Contener el riesgo, preservar evidencia pertinente, revocar o rotar accesos comprometidos, coordinar restauración y comunicar según las obligaciones aplicables.

Si una clave quedó pública, borrar el último commit no demuestra que dejó de estar expuesta. Puede subsistir en historial y copias: debe tratarse como potencialmente comprometida.

No existe aquí una regla universal de notificar en 72 horas para cualquier incidente argentino. Los plazos deben verificarse según ley, sector, contrato y eventual aplicación de otros ordenamientos.

13.4. Evidencia de diligencia

Registro de decisiones, análisis de riesgos, pruebas, revisión de dependencias, advertencias, aprobaciones y resultados de restauración.

Documentar no es fabricar una defensa después del problema. Es hacer trazable el proceso mientras ocurre. Tampoco una firma de aceptación demuestra por sí sola que todos los controles funcionaron.

14. Doctrina y jurisprudencia para fundamentar la clase

14.1. Doctrina argentina consultada

Herrera, Marisa; Caramelo, Gustavo; Picasso, Sebastián —directores—, Código Civil y Comercial de la Nación Comentado, tomo IV, Infojus, 2015. Texto disponible en la Facultad de Derecho de la UBA.

Los comentarios a los arts. 1256, 1710 y 1725 permiten trabajar:

- ejecución conforme a conocimientos técnicos razonablemente requeridos;
- información y cooperación durante la prestación;
- prevención dentro de la esfera de actuación;
- valoración contextual del estándar de conducta.

La obra distingue la agravación del estándar de culpa de una ampliación automática de la causalidad. Esta precisión evita concluir que más conocimientos equivalen a responder por cualquier consecuencia.

Los comentarios de los arts. 1708 a 1756 se identifican en la obra con Sebastián Picasso y Luis R. J. Sáenz. No se atribuye a los directores la autoría individual de todos los comentarios.

Las aplicaciones a repositorios, nube, licencias y despliegues son construcciones didácticas de esta clase, no ejemplos textuales de los comentaristas.

14.2. TJUE — SAS Institute Inc. contra World Programming Ltd

Identificación: Tribunal de Justicia de la Unión Europea, Gran Sala, sentencia del 2 de mayo de 2012, asunto C-406/10. Se consultó el texto íntegro oficial en español.

Hechos relevantes

World Programming desarrolló un sistema que reproducía funcionalidades del sistema SAS. Según el contexto expuesto por el tribunal remitente, no había accedido al código fuente ni copiado su texto o diseño estructural. También se discutían manuales y el uso de una versión licenciada.

Qué resolvió y qué debe distinguirse

Los apartados 39 a 46 distinguen la funcionalidad, el lenguaje y los formatos de archivos de las formas de expresión protegidas del programa conforme a la Directiva 91/250. El apartado 43 advierte que utilizar código fuente u objeto ajeno puede implicar reproducción parcial.

Los apartados 50 a 62 analizan la observación, estudio y verificación por el usuario legítimo dentro de las operaciones autorizadas y sin infringir derechos exclusivos.

Los apartados 63 a 70 señalan que reproducir elementos de un manual puede vulnerar derechos si se reproduce la expresión de la creación intelectual del autor; corresponde verificarlo al órgano nacional.

Aplicación docente

No es lo mismo implementar una función con código propio que copiar expresión protegida. Tampoco «copiar funcionalidad» concede una autorización universal para copiar código, documentación o secretos.

Su relación con el art. 1 de la Ley 11.723 es comparativa: ambos permiten trabajar la distinción entre ideas y expresión. El fallo interpreta derecho europeo y no es vinculante en Argentina ni traslada automáticamente las excepciones europeas a nuestro país.

No resuelve la compatibilidad de GPL con un contrato concreto ni habilita cualquier ingeniería inversa. Su valor en esta clase es orientador y delimitado.

Pregunta para el debate

Dos programas calculan el mismo resultado. ¿Eso demuestra que uno copió al otro?

No. Deben analizarse código, documentación, acceso, originalidad y evidencia. Coincidencia funcional no prueba por sí sola copia de expresión protegida; tampoco descarta que exista.

---

15. Taller integrador — Auditoría previa al lanzamiento de TurnoClaro

Caso ficticio

Un equipo desarrolla TurnoClaro, una aplicación de turnos y administración de consultorios. El contrato promete al cliente una versión para instalar en su infraestructura y una oferta posterior como servicio.

El equipo prepara esta composición:

- Módulo propio de interfaz y agenda.
- Biblioteca de fechas bajo MIT.
- Módulo de exportación bajo Apache-2.0, modificado, cuyo paquete incluye NOTICE.
- Motor de reportes GPLv3 incorporado al proceso principal. Todavía no se documentó el modo de integración ni se revisaron posibles excepciones.

- Fragmento obtenido de un repositorio público sin licencia identificada.

El cliente exige «todo cerrado, exclusivo y sin mencionar terceros». Se entregará una imagen de contenedor para la instalación. No se preparó el código fuente correspondiente ni un paquete de avisos.

El entorno de prueba está en una máquina virtual IaaS abierta con una cuenta personal. El equipo supone que el proveedor respalda la base, pero no verificó el plan. No se probó restauración.

Para probar se copió una base real con nombres y motivos de consulta. Una clave administrativa quedó en el historial del repositorio. El responsable comercial pide publicar al día siguiente porque «ya está todo funcionando».

No ocurrió una filtración confirmada ni existe una pericia. Deben distinguirse exposición, riesgo y daño comprobado.

Organización de los 23 minutos

- 3 minutos: lectura e identificación de problemas.
- 10 minutos: matriz y decisión grupal.
- 7 minutos: puesta en común.
- 3 minutos: devolución docente.

Consigna

Elaborar una matriz con estas columnas: hallazgo, información faltante, riesgo jurídico/técnico, medida, responsable propuesto y evidencia de cierre.

Responder además:

- ¿Qué autorizaciones y obligaciones deben comprobarse por componente?
- ¿Qué cambia entre instalación entregada y servicio por red?

- ¿Qué promesas al cliente deben corregirse?
- ¿Qué responsabilidades no cubre automáticamente IaaS?
- ¿Qué medidas concretas requieren las credenciales y datos de prueba?
- ¿Se aprueba el lanzamiento, se aprueba con condiciones o se posterga? Justificar sin inventar hechos.

Los responsables se proponen por rol. Los plazos pertenecen a la simulación, no son vencimientos reales ni tareas del espacio de trabajo.

Matriz orientadora

Hallazgo	Dato por verificar	Medida propuesta	Rol responsable	Evidencia de cierre
Biblioteca MIT	Versión y avisos originales	Incorporar avisos exigidos al paquete	Desarrollo y revisión de licencias	Inventario y paquete de avisos
Apache-2.0 modificado	Archivos cambiados y NOTICE pertinente	Conservar licencia, avisos y registrar modificaciones	Desarrollo	Revisión de la distribución
Motor GPLv3	Integración, obra cubierta y excepciones	compatibilidad antes de entregar; cumplir o sustituir/obtener	Arquitectura y asesoramiento jurídico	Informe y paquete autorizado

		licencia alternativa		
Fragmento sin licencia	Origen y permiso	No distribuirlo hasta obtener autorización suficiente o reemplazo	Desarrollo	Permiso verificable o sustitución
Cuenta personal	Titularidad y recuperación	Establecer control organizacional y accesos adecuados	Administración técnica	Cuenta, roles y recuperación verificados
Backup supuesto	Cobertura real y responsables	Configurar respaldo y probar restauración	Operaciones	Registro de restauración
Base real de prueba	Necesidad, autorización y exposición	Restringir acceso y sustituir por datos sintéticos cuando proceda	Datos y seguridad	Inventario, permisos y prueba segura
Clave en historial	Alcance de exposición	Revocar/rotar, revisar usos	Seguridad	Rotación y análisis

		y tratar copias		documentados

Guía de resolución

A. Licencias y exclusividad

MIT y Apache-2.0 no son, por sí mismas, incompatibles con un producto comercial cerrado, pero no permiten suprimir las condiciones ni atribuir exclusividad sobre derechos ajenos.

La imagen entregada habilita a recibir copias: no puede descartarse GPLv3 como si solo existiera interacción por red. Debe resolverse cómo se integra el motor y qué obra queda cubierta. No hay datos suficientes para concluir que todo el proyecto debe publicarse; sí para impedir una entrega sin revisión.

El fragmento sin licencia requiere permiso suficiente o sustitución. La disponibilidad pública no prueba autorización.

La promesa comercial debe distinguir código propio, terceros, derechos de uso y obligaciones. No se conserva una cláusula falsa para «arreglarla después».

B. Infraestructura

La VM no demuestra que alguien administre respaldos ni seguridad del sistema operativo. Hay que verificar servicio contratado, configuración y obligaciones del equipo.

También deben resolverse acceso institucional, recuperación de cuenta, facturación y baja de personas. El cliente no debería depender de la disponibilidad de una cuenta personal sin una transición adecuada.

C. Datos y credenciales

Los motivos de consulta pueden revelar datos de salud. No basta describir la base como «de prueba». Deben analizarse necesidad, legitimación, acceso y seguridad.

La clave expuesta exige rotación o revocación y análisis de uso. Eliminar el texto del archivo actual no basta. No debe afirmarse que ocurrió un acceso indebido si todavía no existe evidencia.

D. Decisión de lanzamiento

Con los hechos dados, no corresponde aprobar una entrega incondicionada. Existen cuestiones de permisos, secretos y recuperación sin resolver.

La solución no es necesariamente paralizar todo desarrollo. Puede continuarse con pruebas aisladas y datos sintéticos mientras se cierran los puntos impeditivos. La entrega y la producción deben esperar la resolución de los riesgos que las afectan.

Una aprobación comercial no reemplaza licencias, obligaciones legales ni controles necesarios.

E. Evidencias esperadas

Inventario versionado, textos de licencia, avisos de terceros, decisión sobre GPL, control de cuenta, rotación de secretos, restauración probada, datos de prueba adecuados y aprobación de una versión identificada.

No alcanza un mensaje «revisado». Debe poder reconstruirse qué se revisó, con qué resultado y qué quedó pendiente.

Criterios de evaluación sugeridos para el taller

- Diferenciar hecho verificado, hipótesis y dato faltante.
- Identificar norma o condición relevante.

- Evitar conclusiones automáticas sobre copyleft y responsabilidad.
- Proponer medidas concretas y evidencia de cumplimiento.
- Justificar una decisión proporcionada.

Estos criterios organizan la actividad de clase; no modifican por sí solos las condiciones de aprobación de los parciales.

---

16. Preguntas frecuentes

«¿Puedo vender software abierto?»

Sí, si cumplo la licencia. Comercial y abierto no son categorías opuestas.

«¿Puedo usar en mi trabajo cualquier código visible en internet?»

No. Debo verificar derechos, licencia y condiciones del uso pretendido.

«¿Con SaaS desaparece toda obligación de licencias?»

No. Hay que revisar componentes, distribución efectiva y términos concretos. El navegador también puede recibir código y otros archivos.

«¿La nube se hace cargo de cualquier pérdida?»

No. Depende de la prestación, las obligaciones, la causa y el régimen aplicable. Debe verificarse quién respalda y restaura.

«¿Un NDA me permite enviar datos a una herramienta externa?»

No por sí solo. Deben examinarse finalidad, autorización, condiciones del servicio y legislación de datos.

«¿Un escáner de vulnerabilidades prueba que el sistema es seguro?»

No. Es una herramienta dentro de un proceso. Su resultado tiene alcance y limitaciones.

«¿Si el cliente aceptó un riesgo ya no tengo responsabilidad?»

No automáticamente. Hay que examinar qué podía disponer, qué se informó, qué conducta se realizó y qué derechos no podían excluirse.

«¿Las cláusulas de ausencia de garantía de una biblioteca me liberan frente a mi cliente?»

No automáticamente. El integrador tiene una relación contractual propia y puede haber asumido prestaciones adicionales.

«¿Publicar todo el fuente resuelve cualquier problema GPL?»

No necesariamente. Deben cumplirse todas las condiciones pertinentes y respetarse derechos de terceros. Publicar código ajeno que no podía divulgarse puede crear otro problema.

---

17. Cierre

«Hoy vimos que una decisión técnica también distribuye permisos, costos y responsabilidades.

Elegir una biblioteca no es solamente elegir funciones. Es aceptar condiciones de uso y entrega.

Elegir SaaS, PaaS o IaaS no es solamente elegir dónde corre el programa. Es decidir qué capas administramos y qué dependencias tendremos.

Y actuar profesionalmente no es prometer que nunca habrá errores. Es prever razonablemente, informar, proteger, documentar y corregir.

Antes de entregar un sistema tenemos que poder contestar tres preguntas: ¿funciona como prometimos?, ¿tenemos derecho a entregarlo así?, ¿tomamos los cuidados exigibles para evitar daños?

Si falta una respuesta, todavía hay trabajo profesional por hacer».

Ticket de salida

Responder brevemente:

- Una diferencia entre licencia y titularidad.
- Una condición que revisarían antes de entregar un componente abierto.
- Una responsabilidad que el cliente conserva al contratar nube.
- Una evidencia concreta de diligencia profesional.

Tarea de aplicación

Elegir un proyecto ficticio o académico, sin datos ni credenciales reales, e identificar tres componentes y un servicio externo.

Para cada componente: versión, origen, licencia, uso, forma de entrega y condiciones relevantes. Para el servicio: modelo, responsabilidades, recuperación y salida.

Cerrar con una advertencia profesional de cinco líneas dirigida a un cliente no técnico. La fecha de entrega se fija según el cronograma docente; no se presume una fecha.

18. Síntesis

1. Funcionamiento técnico y habilitación jurídica son controles distintos.
2. Un pliego convierte necesidades en prestaciones y pruebas.
3. SaaS, PaaS e IaaS distribuyen capas de control, no eliminan automáticamente responsabilidades.
4. Pagar o recibir el código no equivale a adquirir todos sus derechos.
5. Abierto no significa sin condiciones ni necesariamente gratuito.
6. MIT exige conservar avisos; Apache-2.0 agrega condiciones específicas y una licencia delimitada de patentes.
7. GPLv3 permite actividad comercial y exige analizar obra cubierta, entrega y fuente correspondiente.
8. La compatibilidad depende de versión, arquitectura y operación.
9. Diligencia significa cuidados razonablemente exigibles según conocimientos, prestación y contexto.
10. La confidencialidad puede existir aunque no se haya firmado un NDA.
11. Prevenir incluye actuar sobre riesgos dentro de la esfera de control y no agravar incidentes.
12. Documentar permite probar qué se decidió y qué se verificó; no reemplaza hacer los controles.

19. Fuentes y alcance de la verificación

Normativa argentina

- Código Civil y Comercial — texto actualizado oficial: arts. 959, 961, 1256, 1710 y 1725 como núcleo de la clase; art. 1743 para límites a exclusiones de responsabilidad. Las explicaciones no convierten cualquier incumplimiento técnico en responsabilidad automática.

- Ley 25.036 — texto en WIPO Lex: modificaciones a los arts. 1, 4 y 9 e incorporación del art. 55 bis de la Ley 11.723. Se consultó el texto de la reforma, no se presenta esa ficha histórica como consolidación exhaustiva de toda la legislación de propiedad intelectual.
- Ley 25.326 — texto actualizado oficial: arts. 9, 10 y 25 para seguridad, secreto y prestación de servicios informatizados; las transferencias internacionales requieren además revisión específica del régimen aplicable.

Textos de licencias consultados

- MIT — Open Source Initiative.
- Apache License 2.0 — Apache Software Foundation, especialmente secciones 3, 4 y 6 a 9.
- GNU GPL versión 3 — Free Software Foundation, especialmente secciones 0 a 6. La clase no sustituye la revisión de otra versión, excepciones ni licencias diferentes.

Doctrina

- Herrera, Caramelo y Picasso —directores—, Código Civil y Comercial de la Nación Comentado, tomo IV, Infojus, 2015; ejemplar en la UBA. Comentarios a los arts. 1256, 1710 y 1725.

Jurisprudencia

- TJUE, Gran Sala, SAS Institute Inc. contra World Programming Ltd, C-406/10, 2/5/2012 — texto íntegro oficial en español. Se verificaron hechos, fundamentos y parte dispositiva; se utiliza como jurisprudencia comparada orientadora, no vinculante en Argentina.

Referencia técnica

- Mell, Peter; Grance, Timothy, The NIST Definition of Cloud Computing, NIST SP 800-145, septiembre de 2011. Referencia técnica para modelos cloud, no norma jurídica argentina.

Los supuestos, matrices, preguntas y cláusulas son elaboraciones didácticas. No se agregan noticias ni precedentes cuya identidad o fundamentos no hayan podido corroborarse. La matriz aplicada a un producto real exige revisar los textos vigentes, sus versiones, el contrato y la arquitectura efectiva.