

UNIVERSIDAD NACIONAL DE INGENIERÍA

Ingeniería de Software

Seguridad, privacidad, ética y accesibilidad

Modelo de amenazas · OWASP Top 10 · Ley 29733 · WCAG 2.2 · Licencias

Gerson Garrido

Semana 15

Hoy

Sesión A

1. Seguridad desde el diseño
2. OWASP Top 10:2025
3. Autenticación y autorización
4. XSS, inyección y errores
5. Privacidad y Ley 29733

Sesión B

6. Ética: quién queda afuera
7. Accesibilidad con WCAG 2.2
8. Licencias y procedencia

Lecturas

OWASP, *Top 10:2025*. Ley N.° 29733, título preliminar y título I (arts. 4 a 12).

W3C, *WCAG* (resumen en español) y referencia rápida de WCAG 2.2. SWEG, cap. 4 *Engineering for Equity*.

Complementarias: Reglamento de la Ley 29733 (D.S. 016-2024-JUS) y normas de la ANPD; OWASP, cheat sheets de *XSS Prevention* y *Authorization*; ASVS; WCAG 2.2 y *Easy Checks*; WebAIM, *Contrast Checker*; choosealicense.com y su apéndice; SPDX.

SECCIÓN 01

Seguridad desde el diseño

01

Otra forma de evaluar el software

- Hasta ahora: ¿hace lo que debe?
- Hoy: ¿qué le permite hacer a quien no debería? ¿Qué no le permite a quien sí debería?
- Un vecino que cancela la reserva de otro. Una página que muestra el celular de todos. Una persona que no puede reservar porque no usa mouse.

Regla del laboratorio

Estos ataques se prueban **solo contra su propia copia**. Contra un sistema que no es suyo, sin autorización escrita, no es un ejercicio: es un problema.

Las tres propiedades

Confidencialidad
solo lo conocen quienes deben
cualquiera lee un celular

Integridad
solo lo cambian quienes deben
cualquiera cancela una reserva

Disponibilidad
responde cuando se lo necesita
un pedido apaga el servidor

Se deciden al diseñar

“Quién puede cancelar” no se agrega al final. Si el sistema no sabe quién llama, ninguna revisión de código lo arregla. OWASP: un diseño inseguro no se corrige con una implementación perfecta, porque el control nunca se creó.

Modelo de amenazas: cuatro preguntas

1. **¿Qué estamos construyendo?** Activos, actores, puntos de entrada, fronteras.
2. **¿Qué puede salir mal?** Por cada entrada: ¿qué pasa si quien llama miente?
3. **¿Qué vamos a hacer al respecto?** Corregir, reducir, aceptar con una razón escrita o eliminar.
4. **¿Lo hicimos bien?** Evidencia: una prueba, un `curl` que antes daba 200 y ahora da 403.

- Sin herramienta ni experto. Temprano, y se actualiza si cambia algo importante.
- STRIDE, de Microsoft, como ayuda de memoria: *Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege*.
- Un dato que no se guarda no se puede filtrar.

Threat Modeling Manifesto; OWASP, *Threat Modeling*

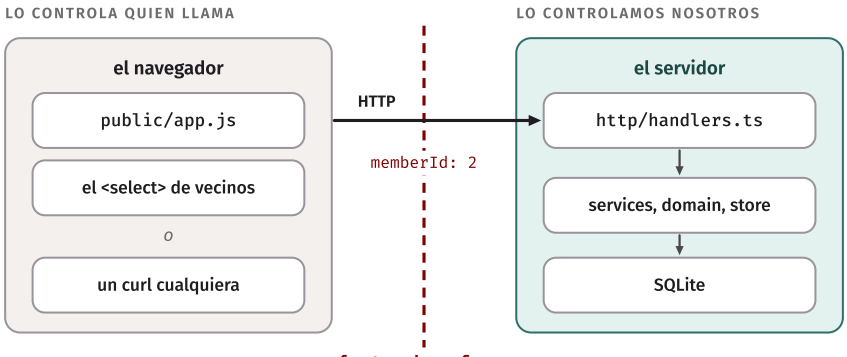
¿Qué estamos construyendo? Los puntos de entrada

Ruta	Devuelve o modifica	Debería poder	Puede hoy
`GET /api/courts`	Las canchas	Cualquiera	Cualquiera
`GET /api/courts/:id/availability`	Horas y nombres de equipos	Según el cliente	Cualquiera
`GET /api/members`	**Nombre, celular, plan**	Administración	**Cualquiera**
`GET /api/members/:id/bookings`	Reservas de esa persona	Ella y adm.	**Cualquiera**
`POST /api/bookings`	Reserva a nombre de `memberId`	El propio vecino	**Cualquiera**
`POST /api/bookings/:id/cancel`	Cancela una reserva	Su owner y adm.	**Cualquiera**
`GET /api/reports/daily`	Ingresos del día	Administración	**Cualquiera**

La última columna tiene un solo valor

Activos: datos de los vecinos, reservas, la caja del día, el servicio funcionando. Kallpa no tiene ningún control de acceso.

La frontera de confianza



Un entero válido: no dice que quien llama sea Luis.

Lo que implica la frontera

- Que `app.js` solo mande el `memberId` elegido no significa nada: corre en la computadora de quien llama.
- Semana 11: lo de afuera es `unknown`. Ahora algo más: un dato puede tener la forma correcta y ser mentira. `memberId: 2` no dice que quien llama sea Luis.

SECCIÓN 02

OWASP Top 10:2025

02



Qué es y qué no es

OWASP Top 10

Un *standard awareness document*: crea conciencia sobre los riesgos más críticos de las aplicaciones web, con un consenso amplio.

- **No es un estándar de verificación.** Para eso está **ASVS**: requisitos concretos, con niveles.
- **No es solo estadística.** Ordenaron 12 categorías con los datos aportados y permitieron que dos se promovieran por una encuesta: los datos miran al pasado.
- **Cuidado con los números.** La página de A01 da “100 % de las aplicaciones probadas” y “3.74 %” de incidencia promedio, sin explicar cómo se relacionan. Salen de lo que aportaron quienes prueban: ordenan la atención, no estiman su riesgo.

OWASP, *Top 10:2025*, Introduction y A01

Las diez, aplicadas a Kallpa

Categoría	Estado	Evidencia
A01 Broken Access Control	**No cumple**	Ninguna ruta revisa quién llama
A02 Security Misconfiguration	Parcial	Sin cabeceras; los errores no revelan detalles
A03 Software Supply Chain Failures	Parcial	Sin dependencias de producción; sin lockfile ni revisión hasta la semana 12
A04 Cryptographic Failures	Parcial	TLS depende del despliegue; celular en claro
A05 Injection	Parcial	SQL con parámetros; XSS en el nombre del equipo
A06 Insecure Design	**No cumple**	La identidad se elige de un `<select>`
A07 Authentication Failures	**No cumple**	No hay autenticación
A08 Software or Data Integrity Failures	Parcial	Sin scripts de CDN; el pipeline llega en la semana 12
A09 Security Logging and Alerting Failures	**No cumple**	Sin alertas; el log guarda el celular
A10 Mishandling of Exceptional Conditions	**No cumple**	Un pedido apaga el servidor

- Es el formato de `docs/seguridad.md` en P5. “Cumple” sin evidencia no es una revisión.
- Las categorías se solapan: no pierdan tiempo discutiendo cuál es la exacta.

Qué cambió en 2025

- **A03 Software Supply Chain Failures.** Expande A06:2021: no solo versiones vulnerables, también lo que falla al construir, distribuir o actualizar (semana 12). Pocos datos; primera en la encuesta.
- **A10 Mishandling of Exceptional Conditions.** Nueva: errores mal manejados, sistemas que fallan abiertos, excepciones sin atrapar.
- **SSRF** pasa a A01. **A01** sigue primero.
- **El XSS está en A05 Injection**: la misma idea, y el intérprete es el navegador.

Error común: “nadie va a encontrar esa ruta”

La página no enlaza `/api/reports/daily`, y la caja sigue siendo pública. Lo que una respuesta devuelve es lo que se publica, aunque la pantalla no lo muestre.

SECCIÓN 03

Autenticación y autorización

03



Dos preguntas distintas

Autenticación

¿Quién eres? Comprobar la identidad de quien llama.

Un evento: se inicia sesión.

Autorización

¿Qué puedes hacer? Decidir si quien llama puede hacer **esta** acción sobre **este** dato.

Ocurre en cada solicitud.

Se confunden porque suelen aparecer juntas

Y confundirlas produce una de las fallas más comunes: creer que, como hay inicio de sesión, los datos están protegidos.

Control de acceso roto: leer lo ajeno

corrida real, con make seed y D=2026-11-04

```
$ curl -s localhost:3000/api/members | jq -c '.members[:2]'
[{"id":3,"name":"Carmen Flores Rojas","phone":"945112233","plan":"regular"},
 {"id":10,"name":"Diego Ramirez Soto","phone":"991234567","plan":"socio"}]

$ curl -s localhost:3000/api/members/1/bookings | jq -c '.[0]'
{"id":1,"courtId":1,"memberId":1,"date":"2026-10-30","startHour":19,"hours":2,"teamName":"Los Pumas de Comas",
 "priceCents":22500,"status":"confirmed",...}

$ curl -s "localhost:3000/api/reports/daily?date=$D" | jq -c '{date,bookings,revenueCents}'
{"date":"2026-11-04","bookings":2,"revenueCents":22125}
```

- `GET /api/members` existe para llenar un `<select>`, que solo usa `id` y `name`. El navegador recibe además los diez celulares.
- Una reserva dice dónde estuvo alguien y cuánto pagó.
- La caja del día, sin identificarse.

Control de acceso roto: actuar por otro

corrida real

```
$ curl -s -X POST localhost:3000/api/bookings -H 'content-type: application/json' \
    -d '{"courtId":1,"memberId":2,"date":"2026-11-04","startHour":8,"teamName":"Prueba"}' | jq -c .
{"booking":{"id":16,"courtId":1,"memberId":2,...,"teamName":"Prueba","priceCents":8100,"status":"confirmed",...}}

$ curl -s -X POST localhost:3000/api/bookings/16/cancel | jq -c .
{"booking":{"id":16,"courtId":1,"memberId":2,...,"teamName":"Prueba","priceCents":8100,"status":"cancelled",...}}

[whatsapp] para 912345678: Hola Luis Huamán Torres, tu reserva de Fulbito 1 el 2026-11-04 de 8:00 a 9:00 ...
```

- S/ 81.00 son S/ 90 con el descuento de **Luis**. Y a Luis le llega un aviso de una reserva que no hizo.
- **IDOR** *(insecure direct object reference)*: el servidor actúa sobre un identificador sin comprobar el derecho. Con correlativos, basta sumar uno.
- Disponibilidad: 15 reservas seguidas, 15 respuestas `200`. Esa cancha se queda sin horas libres ese día, sin ningún límite.
- UUID ayudan como defensa adicional (OWASP), pero no autorizan nada: un identificador se filtra.

Un inicio de sesión no basta

`ejemplos/acceso.ts` pone un servidor delante de las rutas de Kallpa, con tokens fijos de ejemplo. Primero, **solo autenticación**:

```
node --test acceso.test.ts

➤ solo autenticación: hay sesión, pero el problema sigue
  ✓ sin token, 401 (13.934209ms)
  ✓ con la sesión de Rosa se cancela una reserva de Luis (7.375875ms)
  ✓ con la sesión de Rosa se reserva a nombre de Luis y se usa su descuento (4.280667ms)
  ✓ con la sesión de Rosa se leen los celulares de todos (2.622208ms)
```

Hay un candado en la puerta

Y adentro, cada vecino abre todos los casilleros. El servidor sabe quién llama, pero no usa ese dato para decidir nada.

La autorización es una decisión por solicitud

ejemplos/acceso.ts

```ts
export function isAllowed(session: Session, method: string, path: string, ownerOf: BookingOwner): boolean {
  const parts = path.split('/').filter(Boolean);
  const route = `${method} /${parts.join('/')}`;
  if (session.role === 'administracion') return true;
  if (route === 'GET /api/courts') return true;
  if (method === 'GET' && /^\/api\/courts\/\d+\/availability$/.test(path)) return true;
  if (route === 'POST /api/bookings') return true;
  if (method === 'GET' && /^\/api\/members\/\d+\/bookings$/.test(path)) {
    return Number(parts[2]) === session.memberId;
  }
  if (method === 'POST' && /^\/api\/bookings\/\d+\/cancel$/.test(path)) {
    return ownerOf(Number(parts[2])) === session.memberId;
  }
  return false;
}
```

- **La última línea niega**, también las rutas de mañana. **El owner lo dice la base de datos**, no quien llama.
- **La identidad sale de la sesión**: en una reserva, el `memberId` del cuerpo se reemplaza. No se valida: no se le cree.

La regla

```
node --test acceso.test.ts

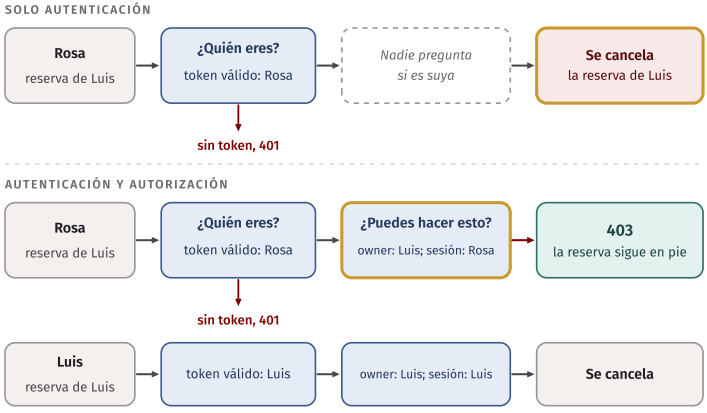
➤ autenticación y autorización
  ✓ Rosa no cancela la reserva de Luis; Luis sí (8.678709ms)
  ✓ el memberId que manda el navegador no cuenta: la reserva queda a nombre de la sesión (3.908959ms)
  ✓ Rosa ve sus reservas, no las de Luis (1.798083ms)
  ✓ la lista de vecinos y la caja del día, solo para la administración (2.975292ms)
```

Cada solicitud se autoriza

En el servidor, contra la **identidad de la sesión**, y **por defecto se niega**. Ocultar un botón no es autorización. Y hay que escribir la prueba de que Rosa recibe 403.

- 401: no sé quién eres. 403: sé quién eres, y no puedes. Con correlativos, 403 frente a 404 revela qué existe: es una decisión.
- CSRF está en A01. Hoy no aplica, sin cookies; en cuanto agreguen una, sí.

Rosa quiere cancelar una reserva de Luis



La autenticación ocurre al iniciar sesión; la autorización, en cada solicitud, y lo no listado se niega.

SECCIÓN 04

XSS: cuando el texto se vuelve código

04

XSS *(cross-site scripting)*

XSS

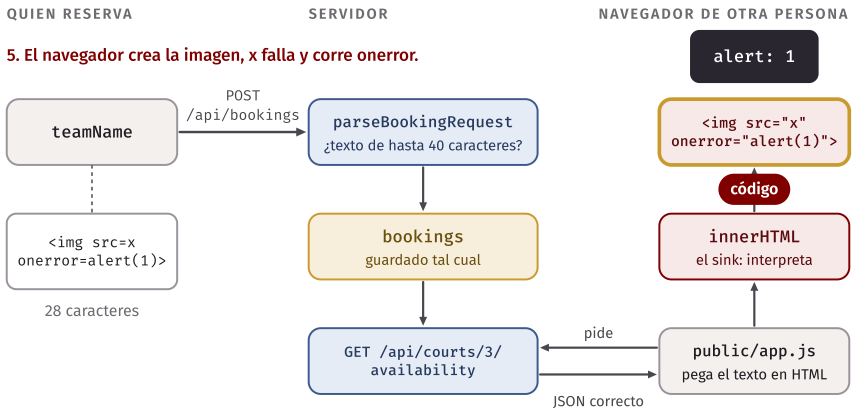
Texto controlado por una persona termina interpretado como HTML o JavaScript en la página de **otra**, con los permisos de esa página.

- **Almacenado:** se guarda en el servidor y se muestra a quien abra la página.
- **Reflejado:** vuelve en la misma respuesta; la víctima abre un enlace preparado.
- **Basado en el DOM:** un script toma texto de la URL y lo mete en la página.

El mecanismo es uno solo

Un dato cruza al lado del código. Vale para el navegador, para SQL, para un shell y, como vieron la semana pasada, para un prompt.

Cómo llega a Kallpa



El servidor no se equivocó

- El servidor y el JSON hicieron bien su trabajo. El problema aparece donde el dato se vuelve HTML: el **sink**.
- El código de quien reservó corre en la computadora de quien miró la página.

El sink, y lo que se ve

public/app.js

```
slotsBox.innerHTML = slots
  .map((slot) =>
    slot.available
      ? `<div class="slot free" data-hour="${slot.hour}">${slot.hour}:00</div>`
      : `<div class="slot taken">${slot.hour}:00<small>${slot.teamName}</small></div>`,
  ).join('');
```

el DOM en Chrome, con tres reservas

```
<div class="slot taken">8:00<small><b>prueba</b></small></div>
<div class="slot taken">9:00<small><img src="x" onerror="alert(1)"></small></div>
<div class="slot taken">10:00<small><script>alert(2)</script></small></div>
```

- `<b>`: en negrita, la prueba inofensiva. `<img ... onerror>`: se ejecutó, un solo `alert`, con el 1.
- `<script>`: no se ejecutó; insertado con `innerHTML` no corre (MDN). No concluyan que es seguro.
- Los 404 de la consola son `/x` y `/favicon.ico`: no son una violación de CSP.

No es un solo lugar

```
make sinks

/work/code/week-15/build/kallpa/public/app.js
  36:3  error  innerHTML interpreta HTML: de dónde viene el texto?  no-restricted-syntax
  41:3  error  innerHTML interpreta HTML: de dónde viene el texto?  no-restricted-syntax
  47:3  error  innerHTML interpreta HTML: de dónde viene el texto?  no-restricted-syntax
  58:3  error  innerHTML interpreta HTML: de dónde viene el texto?  no-restricted-syntax
...
```

- 47 y 58 (disponibilidad y Mis reservas): el nombre del equipo lo escribe cualquiera.
- 36 y 41 (canchas y vecinos): los escribe la administración. Menos urgente, y el mismo patrón.
- Corregir solo la disponibilidad deja el ataque vivo en Mis reservas.
- Hoy el payload casi no agrega daño: `/api/members` ya es público. Con sesiones, correría con la sesión de la víctima. También puede cambiar la página para pedir una contraseña.
- Los 40 caracteres no salvan: esta carga usa 28, y se puede traer el resto de otro lugar.

La corrección: codificar la salida (uno de los cuatro sinks)

public/app.js

```
1  // Crea un elemento con su texto. textContent nunca interpreta HTML.
2  function element(tag, className, text) {
3    const node = document.createElement(tag);
4    if (className) node.className = className;
5    if (text !== undefined) node.textContent = text;
6    return node;
7  }
8  function slotElement(slot) {
9    const label = `${slot.hour}:00`;
10   if (slot.available) {
11     const free = element('div', 'slot free', label);
12     free.dataset.hour = slot.hour;
13     return free;
14   }
15   const taken = element('div', 'slot taken', label);
16   taken.append(element('small', '', slot.teamName));
17   return taken;
18 }
19 slotsBox.replaceChildren(...slots.map(slotElement));
```

Por qué en la salida, y no en la entrada

el DOM, con los cuatro innerHTML corregidos: ningún alert

```
<div class="slot taken">9:00<small>&lt;img src=x onerror=alert(1)&gt;</small></div>
```

- `innerHTML` pide interpretar. `textContent` dice que es texto: no hay nada que escapar.
- El servidor **sigue guardando** `<img src=x ...>`, y está bien. *Los Pumas <3* y *O'Higgins* son legítimos, y no sabe dónde se va a mostrar: HTML, URL, CSV, WhatsApp.

Error común

Filtrar la entrada buscando `<script>`. Las listas negras se saltan: ya vimos que `<script>` ni siquiera era el problema. Validar sirve para las reglas del negocio; no es la defensa contra XSS.

OWASP, *Cross Site Scripting Prevention Cheat Sheet*

Segunda capa: Content Security Policy

lo que manda hoy Kallpa, y lo que puede mandar

```
$ curl -s -D - -o /dev/null localhost:3000/
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 1254
...

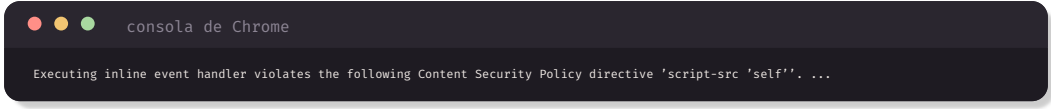
Content-Security-Policy: default-src 'none'; script-src 'self'; style-src 'self'; connect-src 'self';
  img-src 'self'; base-uri 'none'; form-action 'none'; frame-ancestors 'none'
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer
```

- Kallpa solo usa recursos propios, en archivos separados: admite una política estricta.
- Se prueba en el navegador: una CSP demasiado estricta rompe la página en silencio, y la consola dice qué se bloqueó.
- `nosniff`: el navegador respeta el tipo declarado, sin adivinarlo. La ruta 404 repite lo que recibe, pero en `application/json`: no se interpreta.

OWASP, *HTTP Headers Cheat Sheet*

La CSP es una segunda capa

Solo la cabecera, sin corregir el `innerHTML`. Ningún `alert`, y en la consola:



- Pero `<b>prueba</b>` sigue en negrita y el `<img>` sigue en la página. Se detuvo el script, no la inyección.
- Se puede agregar un aviso falso o un formulario. Y basta un `'unsafe-inline'` para perder la capa.

> **Las dos**
> El XSS se corrige en el sink. La CSP limita el daño de lo que se escape.

SECCIÓN 05

Inyección y errores

05



Inyección SQL: lo que Kallpa hace bien

corrida real

```
$ curl -s -X POST localhost:3000/api/bookings -H 'content-type: application/json' \
    -d '{"courtId":1,"memberId":1,"date":"2026-11-04","startHour":9,"teamName":"x'\'' OR 1=1; DROP TABLE bookings; --"}' \
    | jq -c .booking.teamName
"x' OR 1=1; DROP TABLE bookings; --"

$ curl -s localhost:3000/api/courts | jq -c '.courts|length'
4
```

Sentencias preparadas, siempre: el parámetro viaja aparte de la consulta. Pero una herramienta avisa:

- `make sinks` marca `src/seed.ts:18` y `src/store/database.ts:54`: “SQL armado con un template”.
- `seed.ts`: `table` solo recibe `'courts'`, `'members'` y `'bookings'`. `database.ts`: un entero del bucle.
- Son **falsos positivos**: nada viene de fuera de la frontera. Decidirlo, con la razón escrita, es parte de la auditoría.

A10: un pedido apaga el servidor

corrida real

```
$ curl -s --path-as-is -m 3 -i localhost:3000// ; echo exit=$?
exit=52
$ curl -s -m 3 localhost:3000/api/courts ; echo exit=$?
exit=7
```

src/http/server.ts

```
1  return http.createServer(async (req, res) => {
2    const url = new URL(req.url ?? '/', 'http://localhost');
3    try {
```

- `new URL` lanza `TypeError: Invalid URL` antes del `try`. Nadie la atrapa, y Node termina el proceso.
- Denegación de servicio en un comando, sin cuenta. Y sin alerta: también A09 (semana 13).
- Una ruta mal codificada falla *dentro* del `try`: 500 sin detalles, y el servidor sigue.

Con IA

- El código generado repite los patrones comunes, incluidos los inseguros: `innerHTML`, SQL con template, un token en el código.
- El nombre de equipo que aquí es XSS, en la semana 14 es una inyección de prompts indirecta. Y la salida de un modelo tampoco va a `innerHTML`.

Un estudio, con sus límites

Pearce y otros (IEEE S&P 2022): GitHub Copilot, 89 escenarios relacionados con debilidades de alto riesgo, 1 689 programas, cerca de 40 % vulnerables. **Límites:** una herramienta de 2021, en escenarios donde el error es posible. No es “el 40 % del código generado”.

La revisión no cambia

Busquen por dónde entra el texto no confiable y a dónde llega.

SECCIÓN 06

Privacidad y Ley 29733

06

Antes de empezar

Esto no es asesoría legal

Es una introducción técnica en un curso de ingeniería de software, para saber qué preguntar. Decide el titular del negocio, con su abogado o con la Autoridad Nacional de Protección de Datos Personales. Antes de usar algo en un proyecto real, lean el texto vigente.

- **Ley N.° 29733** (2011), con las modificaciones del D.L. 1353 (2017).
- **Reglamento vigente:** D.S. 016-2024-JUS, publicado el 30 de noviembre de 2024; vigencia "a partir de los 120 días calendario siguientes", con plazos más largos para algunas obligaciones.
- **Derogado:** D.S. 003-2013-JUS. Cuidado: el PDF de la ley en gob.pe trae, a continuación, ese reglamento viejo.

El vocabulario

Datos personales (art. 2.4)

"Toda información sobre una persona natural que la identifica o la hace identificable a través de medios que pueden ser razonablemente utilizados".

- El reglamento (art. III.4) incluye hábitos y localización, y la combinación de datos. Una reserva dice dónde va a estar alguien.
- **Sensibles** (art. 2.5): biométricos, salud, ingresos, origen, opiniones… El DNI y el celular no figuran en esas listas.
- **Titular de datos** (art. 2.16): el vecino. **Titular del banco** (art. 2.17): quien determina la finalidad, el contenido y las medidas de seguridad; en su proyecto, muy probablemente, el cliente. La palabra engaña.
- **Encargado** (art. 2.7): trata los datos por encargo del titular del banco.
- **Tratamiento** (art. 2.19): cualquier operación. Leer una tabla, o copiarla a su laptop, también.

Los principios rectores

Principio (art.)	Qué dice	En Kallpa, para discutir
Legalidad (4)	Sin medios fraudulentos, desleales o ilícitos	Se recogen en la administración
Consentimiento (5)	Debe mediar, con excepciones (art. 14)	¿Qué es necesario para la reserva?
Finalidad (6)	"Determinada, explícita y lícita"	Promociones por ese celular: otra finalidad
Proporcionalidad (7)	"Adecuado, relevante y no excesivo"	El celular de todos para un `<select>`
Calidad (8)	Solo por el tiempo necesario	Las reservas no se borran nunca
Seguridad (9)	Medidas técnicas, organizativas y legales	Sin control de acceso
Disposición de recurso (10)	Vías para reclamar	¿A quién le escribe un vecino?
Nivel adecuado (11)	En el flujo transfronterizo	¿En qué país está el servidor?

- La lista "es enunciativa" (art. 12). El reglamento (art. IX) agrega **transparencia** y **responsabilidad proactiva**: ser "capaz de demostrar tal cumplimiento".
- La columna derecha es lectura de ingeniería, no una conclusión legal.

Consentimiento: no todo lo necesita, pero todo se informa

- **Cuando se necesita:** previo, informado, expreso e inequívoco (art. 13.5), y libre (reglamento, art. 2). Un clic cuenta. La carga de la prueba es de quien trata los datos (art. 9).

> **Art. 14.5: no se requiere**
>
> "Cuando los datos personales sean necesarios para la preparación, celebración y ejecución de una relación contractual en la que el titular de datos personales sea parte [...]"

- Una lectura razonable: nombre, hora y un medio para confirmar la reserva. Promociones: no; "debe obtenerse el consentimiento" (reglamento, art. 1.3). El límite exacto no lo decide un programador.
- **Siempre se informa** (art. 18): finalidad, destinatarios, plazo, derechos. En línea, con una política de privacidad accesible, que no reemplaza al consentimiento (reglamento, art. 7).
- La política tiene que decir lo que el sistema hace. Eso solo lo sabe quien lo construyó.

Los derechos, y lo que piden al software

Derecho	Art.	Plazo	Qué le pide al software
Información	18	8 días	Saber decir qué se hace con los datos
Acceso	19	20 días	Sacar todo lo de una persona
Actualización, inclusión, rectificación, supresión	20	10 días	Corregir y borrar **sin romper**
Impedir el suministro	21	10 días	Saber a quién se entrega qué
Oposición	22	10 días	Dejar de tratar los datos de alguien
Tratamiento objetivo	23	10 días	Que una decisión importante no sea solo automática

- Días hábiles; plazos del reglamento, art. 69 (10 días para “los otros derechos”). “Cancelación” (reglamento) es la “supresión” de la ley: los derechos “ARCO” son una abreviatura.
- Kallpa no puede borrar a un vecino: `bookings.member_id` es una clave foránea. ¿Qué pasa con la caja? Se puede anonimizar (art. 2.14): es una decisión de esquema.

Lo que depende de condiciones

Lean la condición antes de repetir la consecuencia.

- **Control de acceso y trazas** (art. 46): registros de quién hizo qué, por dos años como mínimo. La traza de la acción, no el celular en cada línea.
- **Inscripción** del banco de datos (art. 42) y **documento de seguridad** con inventario (art. 47).
- **Incidentes**: se documenta cualquiera (art. 35). A la Autoridad, en 48 horas, si hay grandes volúmenes, muchas personas, datos sensibles o un perjuicio evidente (art. 34.1).
- **Oficial de Datos** (art. 37): en entidades públicas, grandes volúmenes o datos sensibles; progresivo por tamaño de empresa.
- **Evaluación de impacto** (art. 40): "de manera facultativa".
- **Menores** (art. 22): consentimiento de quien ejerce la patria potestad o tutela; de 14 a 17 años, de acuerdo a su capacidad.
- **Multas** (ley, art. 39): leves de 0.5 a 5 UIT, graves hasta 50, muy graves hasta 100, con un tope de 10 % de los ingresos brutos anuales.

El inventario de Kallpa, y lo que se olvida

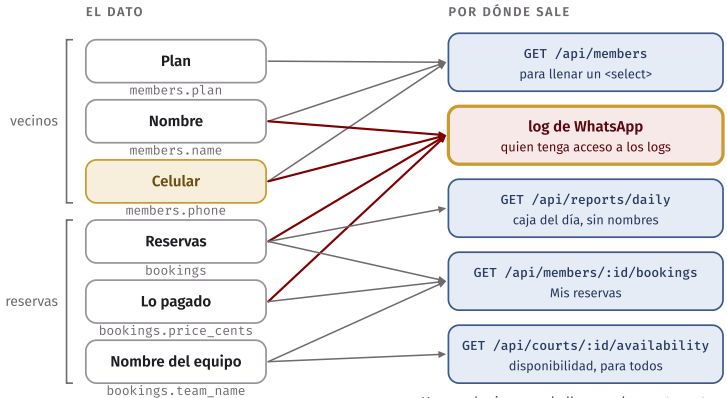
Dato	Por dónde sale	Finalidad	Quién lo necesita
Nombre	`/api/members`, `<select>`, log	Saber quién reserva	Esa persona; administración
Celular	`/api/members`, **log**	Avisarle de su reserva	Administración; quien envía el aviso
Plan	`/api/members`	Calcular el descuento	El cálculo del precio
Reservas	`/api/members/:id/bookings`	Administrar la cancha	Esa persona; administración
Equipo	Disponibilidad, para todos	Saber quién tiene la cancha	Según el cliente

la salida del servidor

```
[whatsapp] para 912345678: Hola Luis Huamán Torres, tu reserva de Fulbito 1 el 2026-11-04 de 8:00 a 9:00 ...
```

- Los logs se guardan más tiempo y con menos cuidado que la base de datos. Es A09, y también es proporcionalidad.
- Quién va a estar, dónde y a qué hora: a veces es justo lo que no se quiere que un desconocido sepa.

Por dónde sale cada dato



Hoy, cualquiera puede llamar a las cuatro rutas.

Finalidad y proporcionalidad: los cambios mínimos

1. `GET /api/members` devuelve solo `id` y `name`. Una línea, y salen los celulares.
2. El log deja de escribir el mensaje y el celular.
3. Una política de privacidad: qué datos, para qué, hasta cuándo, a quién escribirle.
4. Un plazo de conservación, acordado con el cliente.
5. Con sesiones: cada vecino ve lo suyo. La lista de vecinos deja de ser pública.

- **Otro vecino**, para reservar, necesita las horas libres. No el celular, ni el plan, ni el nombre de nadie.
- **La administración** sí, pero no hace falta mandarlo a cada navegador.

Cuatro palabras

Qué se guarda, **para qué**, **quién** lo ve y **hasta cuándo**. Y lo que más protege un dato es no recogerlo.

Quién es quién en su proyecto

- **¿Quién es el titular del banco?** Probablemente el cliente, y ustedes, encargados. Art. 30: no se usa para otro fin ni se transfiere, "ni aun para su conservación", y al terminar "deben ser suprimidos", salvo autorización. También la copia de su laptop.
- **¿Quién más los recibe?** Hosting, correo, logs (reglamento, arts. 28 y 29). En otro país, flujo transfronterizo (ley, art. 15).
- **¿Y una herramienta de IA?** Pegar un dump de la base puede ser una transferencia (art. 2.18). No hace falta resolverlo para decidir: no se pegan datos personales reales sin permiso del cliente.

Lo que no puedan responder

Anótenlo como **pregunta para el cliente**. Sobre registro u oficial de datos en una asociación o una empresa pequeña, no les toca decidir. Sí les toca decir qué datos hay, dónde están y quién los ve.

SECCIÓN 07

Ética: quién queda afuera

07

“Bias is the default”

- Quien construye decide, aunque no lo note, quién puede usar el producto y a quién perjudica. “Hace falta un smartphone” es una decisión.
- **Un caso:** 2015, Google Photos etiquetó a personas negras con un insulto racista. Los datos no representaban a la población; las pruebas no lo vieron, así que lo vieron los usuarios. En 2018, el problema de fondo seguía sin resolverse bien.
- **Otro:** un sistema iba a mostrar evaluaciones bajas a quien contrata, porque lo pedían los reclutadores. Al revisarlo, no predecían el desempeño en otro equipo.
- *“Don’t build for everyone. Build with everyone.”* Diseñen para quien más dificultades tiene.
- *“Don’t assume equity; measure equity.”*

Es el relato de una empresa sobre sí misma, no un estudio: lo valioso es el análisis de cómo pasó.

SWEG, cap. 4

Tres preguntas para su proyecto

1. **¿Quién no puede usar esto?** Sin smartphone o sin datos, con una discapacidad, que lee con dificultad, con otra lengua materna. Si la página reemplaza al teléfono de la administración, alguien quedó afuera, y nadie lo decidió.
2. **¿Qué pasa con los datos si el cliente los pierde o los filtra?** ¿Quién sufre? No el cliente: los vecinos, que no eligieron el sistema.
3. **¿Qué haría falta para que el cliente pueda seguir sin el equipo?** Si el servidor lo paga una tarjeta suya o solo ustedes saben desplegar, el cliente quedó peor que al inicio.

> **En un equipo real**
>
> Empieza con algo simple: decirlo por escrito en el design doc o en el pull request, con una alternativa.

SECCIÓN 08

Accesibilidad

08

WCAG 2.2

WCAG *(Web Content Accessibility Guidelines)*

La referencia internacional del W3C para que personas con discapacidad, permanente o temporal, puedan usar la web.

Cuatro principios

- Perceptible
- Operable
- Comprensible
- Robusto

Tres niveles

- A, AA y AAA; cada uno incluye los anteriores.
- **AA** es el que suelen pedir normas y contratos.
- El W3C no recomienda exigir AAA a un sitio entero.

Primero, **HTML semántico**: `<button>`, `<label>`, `<fieldset>`. ARIA es para lo que el HTML no puede expresar.

Los criterios que más se incumplen

Criterio	Nivel	Cómo se comprueba	¿Una herramienta?
1.1.1 Contenido no textual	A	El `alt` de cada imagen	Que falte, sí; que describa, no
1.3.1 Información y relaciones	A	Clic en la etiqueta: ¿va al campo?	En parte
1.4.1 Uso del color	A	Verla en escala de grises	Casi nunca
1.4.3 Contraste mínimo	AA	Calcular: 4.5:1, y 3:1 en texto grande	Sí, en la mayoría de los casos
2.1.1 Teclado	A	Hacer la tarea sin mouse	**No**
4.1.2 Nombre, función y valor	A	Inspector; lector de pantalla	En parte
4.1.3 Mensajes de estado	AA	Lector de pantalla; `role="status"`	Casi nunca

- 1.1.1: Kallpa no tiene imágenes. Una auditoría también dice lo que miró y no encontró.
- La referencia rápida trae cada criterio con técnicas y ejemplos.

El contraste de Kallpa

```
make contraste

body              #222222 sobre #ffffff  15.91:1  cumple
header h1         #0b6e4f sobre #ffffff   6.25:1  cumple
.hint             #b5b5b5 sobre #ffffff   2.05:1  NO cumple
.mine .cancelled  #b5b5b5 sobre #ffffff   2.05:1  NO cumple
.slot.free        #ffffff sobre #43a047   3.30:1  NO cumple
.slot.taken       #ffffff sobre #e53935   4.23:1  NO cumple
#message.ok       #43a047 sobre #ffffff   3.30:1  NO cumple
#message.error    #e53935 sobre #ffffff   4.23:1  NO cumple
```

- La razón es $(L_1 + 0.05)/(L_2 + 0.05)$, de 1:1 a 21:1. Fórmula pública en WCAG, implementada y probada en `contrast.ts`.
- 4.23:1 está cerca, y no cumple. No se redondea.
- Se comparan **pares**. El mismo verde va de fondo y de texto.
- Lo que no es texto (1.4.11, 3:1): el borde de la hora elegida da 1.40:1 contra el blanco.

Teclado: dónde se traba una reserva

Tab, varias veces, en Chrome. Cada elemento es `etiqueta#id.clase`:

recorrido del foco

```
select#court. > input#date. > input#date. > input#date. > input#date. > select#member. > input#team. >
select#hours. > button#book. > a#. > body > select#court.
```

- Ninguna hora. Son 15 en pantalla, y son `<div>` con un listener de clic: no son controles.
- Sin hora, `Reservar` dice *Primero elige una hora*. Una persona que no usa mouse **no puede reservar**. 2.1.1, nivel A.

la corrección: el elemento que ya lo hace

```
1  const free = element('button', 'slot free', label);
2  free.type = 'button';
```

Con todas las correcciones, la primera hora libre llegó tras 6 Tab, y se completó una reserva solo con el teclado. `role="button"` en un `<div>` lo anuncia, pero sigue sin responder a Enter.

Lo demás

- **Etiquetas** (1.3.1, 4.1.2, 3.3.2). Cinco campos y ninguna `<label>`. El del equipo solo tiene un `placeholder`, que desaparece al escribir.
- **Mensajes de estado** (4.1.3). “Reserva confirmada” aparece en un `<p>` sin marca. Un lector de pantalla no tiene por qué anunciarlo: el foco sigue en el botón. `role="status"`, y se comprueba con NVDA o VoiceOver.
- **Uso del color** (1.4.1). La hora elegida se distingue solo por un borde amarillo. Las libres y ocupadas son verde y rojo, aunque aquí no es solo color: las ocupadas traen el nombre del equipo. Una palabra ayuda a todos.
- **“cancelar”** es un `<a href="#">` que hace una acción. Funciona con teclado, así que no incumple 2.1.1. Un enlace lleva a otro lugar; un botón hace algo.

Cómo se evalúa, y el puntaje que engaña

Kallpa publicada

Lighthouse 12.8.2

```
score 0.81
  color-contrast [18]
  label [1]
  select-name [3]
```

Solo lo que Lighthouse dijo

Lighthouse 12.8.2

```
score 1
```

Etiquetas y colores. Sin tocar los `<div>`.

Chrome 156 en macOS, con el servidor en el contenedor; puede cambiar entre versiones

100 de 100, y no se puede reservar sin mouse

recorrido del foco en esa misma página con 100 de 100

```
select#court. > input#date. > input#date. > input#date. > input#date. > select#member. > input#team. >
select#hours. > button#book. > a#. > body > select#court.
```

Error común

Dar por accesible una página porque Lighthouse le da un puntaje alto. No puede saber si el orden del foco tiene sentido, si un `alt` describe la imagen o si un botón responde al teclado.

- El aviso `label` solo marcó la fecha: al campo del equipo lo aceptó por su `placeholder`.
- El puntaje dice lo que está mal entre lo poco que puede medir, no que la página funcione.

Cómo se evalúa, en orden

1. **Las herramientas**: Lighthouse, el inspector de accesibilidad. Encuentran contrastes y campos sin nombre.
2. **El teclado**: toda la tarea, con Tab, Shift+Tab, Enter y Espacio.
3. **Un lector de pantalla**: NVDA en Windows, VoiceOver en macOS.
4. **Personas**: *Build with everyone*, si se puede.

Con IA

Una herramienta puede leer el HTML y listar etiquetas faltantes. Eso no comprueba que una persona termine la tarea. Al corregir, revisen el diff: `role="button"` y `tabindex="0"` en un `<div>` pueden callar un aviso sin que nada funcione. Se comprueba de la misma forma: con Tab.

W3C, *Easy Checks*; WebAIM, *Contrast Checker*

SECCIÓN 09

Licencias y procedencia

09

Sin licencia, nadie tiene permiso

- El código tiene derechos de autor desde que se crea. Una **licencia** es el permiso.
- Público no es lo mismo que libre. En GitHub, los términos permiten ver y hacer fork; no usarlo en su proyecto.
- Con varios autores, “nadie” empieza a incluirlos a ustedes.
- Kallpa: su `package.json` no declara `license`, y no hay `LICENSE`.

Tipo	Ejemplos	Qué pide
Permisiva	MIT, BSD, ISC, BlueOak-1.0.0	Conservar el aviso
Con patentes	Apache-2.0	Conservar avisos, indicar cambios; concede patentes
Copyleft débil	MPL-2.0	Compartir los cambios a esos archivos
Copyleft	GPL-3.0	Si **distribuye** un derivado: misma licencia, con su código
Copyleft de red	AGPL-3.0	Además, si una versión **modificada** se ofrece por la red

choosealicense.com; un resumen, no el texto legal

Con el cliente, y las dependencias

La licencia se acuerda

- ¿Otros pueden reutilizarlo? ¿Le importaría que una empresa lo venda?
- ¿El equipo lo muestra en su portafolio?
- ¿Hay datos o claves en el repositorio? También en el historial.
- Se declara con su identificador **SPDX** en `package.json` y en `LICENSE`.
- GPL no prohíbe: pide compartir. Puede ser justo lo que quiere el cliente.

El lockfile de la semana 12

```
make licencias

74 MIT
13 Apache-2.0
 6 BSD-2-Clause
 5 ISC
 1 BlueOak-1.0.0
 1 BSD-3-Clause
```

100 paquetes, todos permisivos y de desarrollo: no llegan al producto. El campo lo declara quien publica; para algo importante, miren el `LICENSE`.

✂ Procedencia del código generado

- **¿Puede reproducir código ajeno?** Sí es posible. Carlini y otros (USENIX Security 2021) recuperaron de GPT-2 cientos de secuencias textuales de su entrenamiento, con código y datos personales públicos. **Límites:** un ataque a propósito sobre un modelo de 2019; no mide cuánto pasa en el uso normal.
- **¿De quién es lo generado?** Depende de la ley de cada país y de los términos de la herramienta. Discusión abierta: tampoco aquí hay asesoría legal.

Lo que el equipo sí controla

Registrar qué se generó y con qué (`docs/registro-ia.md`). **Entender** ese código como si fuera propio. **Sospechar** de bloques largos: comentarios ajenos, nombres de otro proyecto. **No pegar** código o datos del cliente sin su permiso.

Un caso: 80 líneas con un comentario que nombra otro proyecto. Busquen el nombre y líneas distintivas. Si aparece el origen, respeten su licencia o reescriban entendiendo. Y anoten qué se aceptó, qué se rechazó y por qué.

SECCIÓN 10

Laboratorio

10

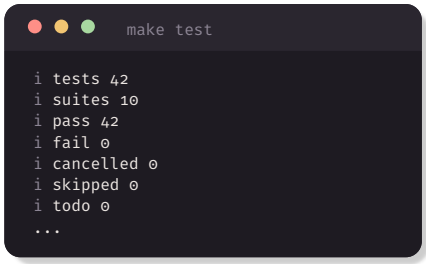
Laboratorio: auditoría de su Kallpa

En parejas, con `make seed`. Cada hallazgo en `docs/auditoria.md`, con `code/week-15/template-auditoria.md`: qué es, cómo se reproduce (**comandos exactos**), categoría, severidad y corrección.

1. **Modelo de amenazas:** activos, actores y cada ruta. Marquen la frontera.
2. **Control de acceso:** con `curl`, lo que un vecino no debería poder hacer.
3. **XSS:** `<b>prueba</b>` y luego `<img src=x onerror=alert(1)>`. Corríjanlo en una rama, con una prueba o un procedimiento escrito.
4. **Datos personales:** inventario de lo que guarda y de cada salida, con los logs. Finalidad, quién lo necesita, si es proporcional, y cambios mínimos. No es asesoría legal.
5. **Cabeceras:** una CSP que no rompa la página. Comprueben la consola.
6. **Accesibilidad:** una reserva solo con el teclado, Lighthouse, Contrast Checker y, si pueden, NVDA o VoiceOver.
7. **Licencias:** cuéntenlas en el lockfile. ¿Cuál le pondrían a Kallpa?
8. **Prioricen:** los dos más graves en pull requests separados, con revisión; el resto, issues.

El código de la semana

- `auditoria.test.ts`: cada debilidad, contra una Kallpa de verdad.
- `ejemplos/acceso.ts`: autenticación frente a autorización.
- `contrast.ts`: la fórmula de WCAG.
- `eslint.sinks.js`: `innerHTML` y templates de SQL.
- `licencias/`: un lockfile real.



Son pruebas al revés

`auditoria.test.ts` pasa **porque las debilidades están ahí**: una corrección la hace fallar. Por eso no va a su repositorio; lo que va es la prueba de que Rosa recibe 403.

Para la próxima clase

- Terminar la auditoría: `docs/auditoria.md` y dos pull requests con revisión.
- **P5**: entrega en la semana 16. Apliquen lo mismo a su proyecto: `docs/seguridad.md` con A01 a A10, los datos personales y la licencia acordada con el cliente.
- Semana 16, mantenimiento, deuda técnica y deprecación. Lecturas: SWEG, caps. 15 *Deprecation* y 22 *Large-Scale Changes*.

Pregunta

En su proyecto, ¿qué dato guardan que, si lo pensaran hoy, no recogerían? ¿Quién de su cliente lo necesita de verdad, y hasta cuándo?