



Entrega 4 Visicontrol: Accesibilidad, usabilidad y pruebas

Integrantes:

- Kevin Adrián Gil Soto (@kevingS0712)
 - Boris Guillermo Briones Dupla
 - Fernando Moreno Mora
 - Rafael Alejandro Ajila Gallegos

Carrera: Ingeniería de Software

Fecha: [03/12/2025]

1. Metodología general de accesibilidad

Para evaluar la accesibilidad de Visicontrol se utilizaron las herramientas Lighthouse (integrada en Chrome) y la extensión WAVE, aplicando las pautas WCAG 2.1.

Las pruebas se ejecutaron en el navegador Google Chrome, simulando vista móvil (herramienta “Device Toolbar”) porque la aplicación está pensada principalmente para uso en teléfonos, aunque es responsive y funciona también en escritorio.

En cada pantalla se revisaron tres aspectos principales:

- Contraste de colores entre texto y fondo.
- Navegación por teclado y foco visible en los elementos interactivos.
- Presencia de etiquetas, nombres accesibles y atributos alt para que los lectores de pantalla puedan interpretar correctamente la interfaz.

1.1 Lighthouse – Pantalla de Login

The screenshot shows the Lighthouse Accessibility audit results for a login page at <http://localhost:5173/login>. The overall score is 100. The report includes a summary section with a green circle icon, followed by sections for Navigation and Additional items to manually check.

NAVIGATION

- The page contains a heading, skip link, or landmark region

These are opportunities to improve keyboard navigation in your application.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) [Show](#)

PASSED AUDITS (18)	Hide
● <code>[aria-*</code>] attributes match their roles	▼
● <code>[aria-hidden="true"]</code> is not present on the document <code><body></code>	▼
● <code>[aria-*</code>] attributes have valid values	▼
● <code>[aria-*</code>] attributes are valid and not misspelled	▼
● Buttons have an accessible name	▼
● Image elements have <code>[alt]</code> attributes	▼
● <code>[user-scalable="no"]</code> is not used in the <code><meta name="viewport"></code> element and the <code>[maximum-scale]</code> attribute is not less than 5.	▼
● ARIA attributes are used as specified for the element's role	▼
● Elements use only permitted ARIA attributes	▼
● Background and foreground colors have a sufficient contrast ratio	▼
● Document has a <code><title></code> element	▼
● <code><html></code> element has a <code>[lang]</code> attribute	▼
● <code><html></code> element has a valid value for its <code>[lang]</code> attribute	▼
● Form elements have associated labels	▼
● Links are distinguishable without relying on color.	▼
● Links have a discernible name	▼

NOT APPLICABLE (38)	Hide
○ <code>[accesskey]</code> values are unique	▼
○ Uses ARIA roles only on compatible elements	▼
○ <code>button</code> , <code>link</code> , and <code>menuitem</code> elements have accessible names	▼
○ Deprecated ARIA roles were not used	▼
○ Elements with <code>role="dialog"</code> or <code>role="alertdialog"</code> have accessible names.	▼
○ <code>[aria-hidden="true"]</code> elements do not contain focusable descendants	▼
○ ARIA input fields have accessible names	▼
○ ARIA <code>meter</code> elements have accessible names	▼
○ ARIA <code>progressbar</code> elements have accessible names	▼
○ <code>[role]</code> s have all required <code>[aria-*</code>] attributes	▼
○ Elements with an ARIA <code>[role]</code> that require children to contain a specific <code>[role]</code> have all required children.	▼
○ <code>[role]</code> s are contained by their required parent element	▼
○ <code>[role]</code> values are valid	▼
○ Elements with the <code>role=text</code> attribute do not have focusable descendants.	▼
○ ARIA toggle fields have accessible names	▼
○ ARIA <code>tooltip</code> elements have accessible names	▼

Herramienta: Lighthouse – pestaña *Accessibility*

Modo: Mobile

URL: <http://localhost:5173/login>

Resultado de accesibilidad: 100 / 100

En esta vista se validó el formulario de inicio de sesión.

Los puntos más relevantes del reporte fueron:

- La página contiene al menos un heading / landmark, lo que permite a los lectores de pantalla y usuarios de teclado entender la estructura básica.
- Todos los botones tienen un nombre accesible, por lo que se anuncian correctamente (ej. botón de ingresar).
- Las imágenes usadas en la vista cuentan con atributo alt.
- Los campos del formulario tienen labels asociados, facilitando que un lector de pantalla relacione el nombre del campo con el input.
- El contraste entre texto y fondo es suficiente según WCAG.
- El documento define el idioma mediante lang en <html> y tiene un <title> válido.
- Los atributos ARIA presentes son válidos, están bien escritos y se utilizan en elementos compatibles.
- La meta viewport permite zoom del usuario (no hay user-scalable="no" y el maximum-scale es adecuado).

Lighthouse también muestra una lista de ítems “Not applicable” (dialogs, barras de progreso, tooltips, etc.).

Estos elementos no existen en la pantalla de login, por lo que esas pruebas simplemente no aplican para esta vista.

Conclusión login:

La pantalla de inicio de sesión cumple todos los checks automáticos de Lighthouse y obtiene 100/100 en accesibilidad, lo que indica un buen soporte para navegación por teclado y tecnologías asistivas en esta parte del sistema.

1.2 Lighthouse - pantalla dashboard

http://localhost:5173/dashboard

There were issues affecting this run of Lighthouse:

- Clearing the browser cache timed out. Try auditing this page again and file a bug if the issue persists.

86

Accessibility

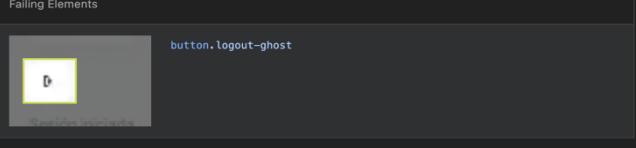
These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

NAMES AND LABELS

▲ Buttons do not have an accessible name

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. [Learn how to make buttons more accessible](#).

Failing Elements



These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

CONTRAST

▲ Background and foreground colors do not have a sufficient contrast ratio.

Low-contrast text is difficult or impossible for many users to read. [Learn how to provide sufficient color contrast](#).

Failing Elements



PASSED AUDITS (13)

Hide

- [aria- \star] attributes match their roles
- [aria-hidden="true"] is not present on the document <body>
- [aria- \star] attributes have valid values
- [aria- \star] attributes are valid and not misspelled
- [user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.
- ARIA attributes are used as specified for the element's role
- [aria-hidden="true"] elements do not contain focusable descendants
- Elements use only permitted ARIA attributes
- Document has a <title> element
- <html> element has a [lang] attribute
- <html> element has a valid value for its [lang] attribute
- Touch targets have sufficient size and spacing.
- Heading elements appear in a sequentially-descending order

http://localhost:5173/dashboard

NOT APPLICABLE (42)

Hide

- (accesskey) values are unique
- Uses ARIA roles only on compatible elements
- button, link, and menuitem elements have accessible names
- Deprecated ARIA roles were not used
- Elements with role="dialog" or role="alertdialog" have accessible names.
- ARIA input fields have accessible names
- ARIA meter elements have accessible names
- ARIA progressbar elements have accessible names
- (role)s have all required [aria- \star] attributes
- Elements with an ARIA (role) that require children to contain a specific (role) have all required children.
- (role)s are contained by their required parent element
- (role) values are valid
- Elements with the role=text attribute do not have focusable descendants.
- ARIA toggle fields have accessible names
- ARIA tooltip elements have accessible names
- ARIA treeitem elements have accessible names

Herramienta: Lighthouse – pestaña Accessibility

Modo: Mobile

URL: <http://localhost:5173/dashboard>

Resultado de accesibilidad: 86 / 100

En esta vista se validó la pantalla principal del sistema (dashboard del administrador).

Los puntos más relevantes del reporte fueron:

- Los elementos ARIA presentes tienen valores válidos y se usan en el tipo de elemento correcto.
- El documento define el idioma mediante lang en <html> y tiene un <title> válido.
- No se usa aria-hidden="true" en elementos que puedan recibir foco.
- Los elementos interactivos cumplen el tamaño mínimo recomendado para pantallas táctiles.
- Los encabezados aparecen en un orden jerárquico correcto, lo que ayuda a la navegación con lector de pantalla.

Problemas encontrados por Lighthouse:

- Botón sin nombre accesible:
 - El botón de cerrar sesión (button.logout-ghost) solo muestra un ícono y no tiene texto ni aria-label.
 - Para un lector de pantalla se anuncia como “button” sin contexto.
 - Solución propuesta: añadir aria-label="Cerrar sesión" o un texto oculto para lectores de pantalla.
- Contraste insuficiente entre texto y fondo:
 - El texto del mensaje de sesión (p.session-note, por ejemplo “Sesión iniciada como Administrador”) y algunos elementos del contenedor div.app-light tienen poco contraste con el fondo.
 - Esto dificulta la lectura para usuarios con baja visión.
 - Solución propuesta: oscurecer el color del texto o aclarar el fondo hasta cumplir con la relación de contraste recomendada por WCAG (mínimo 4.5:1).

Conclusión dashboard:

La pantalla de dashboard tiene una base de accesibilidad aceptable (86/100), pero se identificaron dos mejoras claras: poner un nombre accesible al botón de cerrar sesión y ajustar los colores para mejorar el contraste del texto. Con esos ajustes se debería acercar el puntaje a 100 y ofrecer una mejor experiencia a usuarios con lector de pantalla o baja visión.

1.3 Lighthouse – pantalla visits

The screenshot shows the Lighthouse Accessibility audit results for the URL <http://localhost:5173/visits>. The main score is 100, displayed in a large green circle. Below the score, the section title is "Accessibility". A note states: "These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged." At the bottom, there is a link to "ADDITIONAL ITEMS TO MANUALLY CHECK (10)" and a "Hide" button.

The screenshot shows the "PASSED AUDITS (13)" section of the Lighthouse audit results. It lists 13 items, each with a green circular icon and a descriptive text. The items are:

- [aria-hidden="true"] is not present on the document <body>
- Buttons have an accessible name
- [user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.
- Background and foreground colors have a sufficient contrast ratio
- Document has a <title> element
- <html> element has a [lang] attribute
- <html> element has a valid value for its [lang] attribute
- Form elements have associated labels
- Links are distinguishable without relying on color.
- Links have a discernible name
- Select elements have associated label elements.
- Touch targets have sufficient size and spacing.
- Heading elements appear in a sequentially-descending order

The screenshot shows the "NOT APPLICABLE (44)" section of the Lighthouse audit results. It lists 44 items, each with a grey circular icon and a descriptive text. The items are:

- [accesskey] values are unique
- [aria-*
- Uses ARIA roles only on compatible elements
- button, link, and menuitem elements have accessible names
- ARIA attributes are used as specified for the element's role
- Deprecated ARIA roles were not used
- Elements with role="dialog" or role="alertdialog" have accessible names.
- [aria-hidden="true"] elements do not contain focusable descendants
- ARIA input fields have accessible names
- ARIA meter elements have accessible names
- ARIA progressbar elements have accessible names
- Elements use only permitted ARIA attributes
- [role]s have all required [aria-*
- Elements with an ARIA [role] that require children to contain a specific [role] have all required children.
- [role]s are contained by their required parent element
- [role] values are valid

Herramienta: Lighthouse – pestaña Accessibility

Modo: Mobile

URL: <http://localhost:5173/visits>

Resultado de accesibilidad: 100 / 100

En esta vista se evaluó el formulario donde el usuario agenda una nueva visita (selección de interno, fecha, duración, horario y notas). Los puntos más importantes del reporte fueron:

- Todos los botones del formulario tienen un nombre accesible, por lo que un lector de pantalla puede anunciar claramente las acciones (por ejemplo, botón “Crear”).
- Los campos de formulario (inputs y selects) tienen sus labels asociados, lo que ayuda a que las personas que usan lector de pantalla entiendan qué deben llenar en cada control.
- El contraste entre el texto y el fondo cumple con los valores mínimos recomendados por WCAG, tanto en los textos como en los botones principales.
- El documento define correctamente el idioma mediante el atributo lang en la etiqueta <html> y cuenta con un <title> adecuado.
- La meta viewport permite que el usuario haga zoom en la interfaz (no se usa user-scalable="no" y el maximum-scale es suficiente).
- Los enlaces son distinguibles sin depender solo del color, y los elementos clicables tienen un tamaño de toque adecuado para uso en pantallas táctiles.
- Los encabezados de la página siguen un orden jerárquico correcto, lo que ayuda a la navegación por encabezados.

Lighthouse también marca varias comprobaciones como “Not applicable” (por ejemplo, diálogos modales, barras de progreso, tooltips, elementos meter, etc.).

Estos componentes no existen en la pantalla de “Nueva visita”, por eso esas pruebas simplemente no aplican para esta vista.

Conclusión /visits:

La pantalla de agendamiento de visitas obtiene 100/100 en accesibilidad en modo móvil. Esto indica que el formulario está bien etiquetado, tiene buen contraste y resulta usable para usuarios que navegan con teclado o con tecnologías asistivas.

1.4 Lighthouse – pantalla historial

http://localhost:5173/history

87

Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

NAMES AND LABELS

▲ Form elements do not have associated labels

Labels ensure that form controls are announced properly by assistive technologies, like screen readers. [Learn more about form element labels](#).

Failing Elements

input.input-light

dd/m/Y

▲ Select elements do not have associated label elements.

Form elements without effective labels can create frustrating experiences for screen reader users. [Learn more about the select element](#).

Failing Elements

select.input-light

Todos

PASSED AUDITS (14)

PASSED AUDITS (14)

- [aria-hidden="true"] is not present on the document <body>
- Buttons have an accessible name
- [user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.
- [aria-hidden="true"] elements do not contain focusable descendants
- Background and foreground colors have a sufficient contrast ratio
- Document has a <title> element
- <html> element has a [lang] attribute
- <html> element has a valid value for its [lang] attribute
- Links are distinguishable without relying on color.
- Links have a discernible name
- Touch targets have sufficient size and spacing.
- Cells in a <table> element that use the [headers] attribute refer to table cells within the same table.
- Heading elements appear in a sequentially-descending order
- Tables have different content in the summary attribute and <caption>.

NOT APPLICABLE (41)

- [accesskey] values are unique
- [aria-*] attributes match their roles
- Uses ARIA roles only on compatible elements
- button, link, and menutem elements have accessible names
- ARIA attributes are used as specified for the element's role
- Deprecated ARIA roles were not used
- Elements with role="dialog" or role="alertdialog" have accessible names.
- ARIA input fields have accessible names
- ARIA meter elements have accessible names
- ARIA progressbar elements have accessible names
- Elements use only permitted ARIA attributes
- [role]s have all required [aria-*] attributes
- Elements with an ARIA [role] that require children to contain a specific [role] have all required children.
- [role]s are contained by their required parent element
- [role] values are valid
- Elements with the role=text attribute do not have focusable descendants.

Herramienta: Lighthouse – pestaña Accessibility

Modo: Mobile

URL: <http://localhost:5173/history>

Resultado de accesibilidad: 87 / 100

En esta vista se evaluó la pantalla de Historial de visitas, que muestra la tabla con los registros y filtros (fecha y estado).

Puntos positivos del reporte:

- El documento tiene un <title> válido y el elemento <html> define correctamente el idioma mediante el atributo lang.
- Los botones presentes tienen nombre accesible y se anuncian correctamente a los lectores de pantalla.
- No se utiliza user-scalable="no" en la meta viewport y el maximum-scale es adecuado, por lo que el usuario puede hacer zoom.
- El contraste entre color de texto y fondo cumple con los mínimos recomendados por WCAG.
- Los enlaces son distinguibles sin depender solo del color y tienen un nombre descriptivo.
- Los objetivos táctiles (botones y controles) tienen tamaño y separación adecuados para uso en pantallas táctiles.
- La tabla de historial utiliza cabeceras (headers) que se relacionan con las celdas de datos, lo que facilita la lectura por tecnologías asistivas.
- La jerarquía de encabezados aparece en orden descendente, lo que ayuda a entender la estructura visual y lógica de la pantalla.

Problemas detectados (Names and labels):

- Faltan labels asociados al input de fecha.
El campo de filtro por fecha (input con placeholder dd/mm/aaaa) aparece como input.input-light sin un <label> adecuadamente asociado. Esto hace que los lectores de pantalla no puedan anunciar claramente para qué sirve ese campo.
- Faltan labels asociados al select de estado.
El combo que permite elegir “Todos los estados” (select.input-light) tampoco tiene un label explícito. De nuevo, esto dificulta su uso a personas que navegan solo con lector de pantalla o teclado.

Conclusión historial:

La pantalla de Historial de visitas obtiene 87/100 en accesibilidad.

En general cumple bien con contraste, estructura de la tabla y semántica básica, pero se detectan problemas en los labels de los filtros (fecha y estado).

La corrección propuesta es agregar etiquetas visibles o aria-label/aria-labelledby a estos

controles para que los lectores de pantalla puedan identificarlos claramente y mejorar la experiencia de usuarios con discapacidad visual.

2. Evaluación WCAG con WAVE

2.1 Pantalla: Login

Herramienta: WAVE Web Accessibility Evaluation Tool

URL: <http://localhost:5173/login>

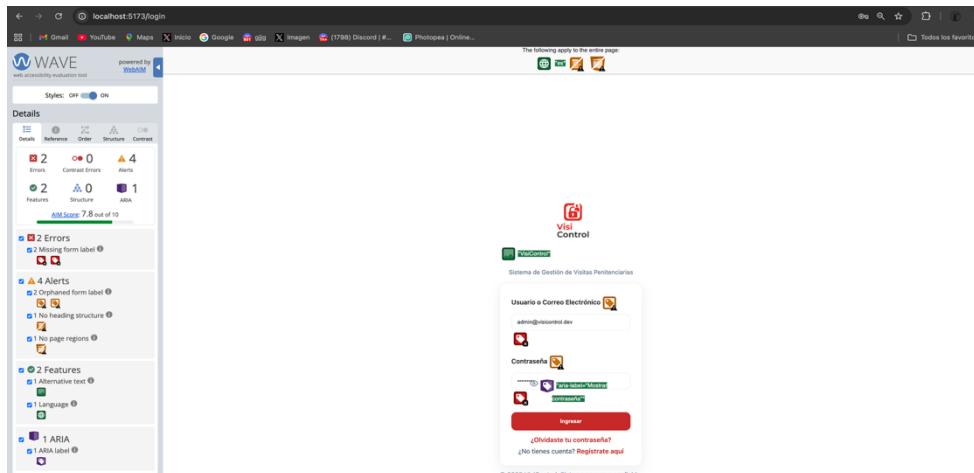
En la vista de inicio de sesión WAVE reporta:

- 2 errores de accesibilidad: ambos por *missing form label* (campos de formulario sin etiqueta explícita).
- 4 alertas:
 - 2 etiquetas huérfanas (*orphaned form label*),
 - 1 alerta por ausencia de estructura de encabezados (*no heading structure*),
 - 1 alerta por ausencia de regiones de página (*no page regions*).
- 2 “features”: detección correcta del idioma de la página y presencia de texto alternativo en imágenes.
- 1 elemento ARIA: se detecta un aria-label que ayuda a los lectores de pantalla.

Interpretación:

La pantalla de login es sencilla y ya tiene idioma declarado y texto alternativo, lo que ayuda a los lectores de pantalla. Sin embargo, WAVE indica que algunos campos no tienen etiqueta asociada correctamente y que no hay una jerarquía clara de encabezados ni regiones de página. Esto no impide usar el sistema, pero puede dificultar la navegación por teclado y el uso con tecnologías asistivas.

En el futuro se debería corregir: asociar bien todas las etiquetas <label> a sus id de input y definir al menos un encabezado principal y una región de contenido.



2.2 Pantalla: Dashboard

Herramienta: WAVE Web Accessibility Evaluation Tool

URL: <http://localhost:5173/dashboard>

Para el panel principal de la aplicación, WAVE muestra:

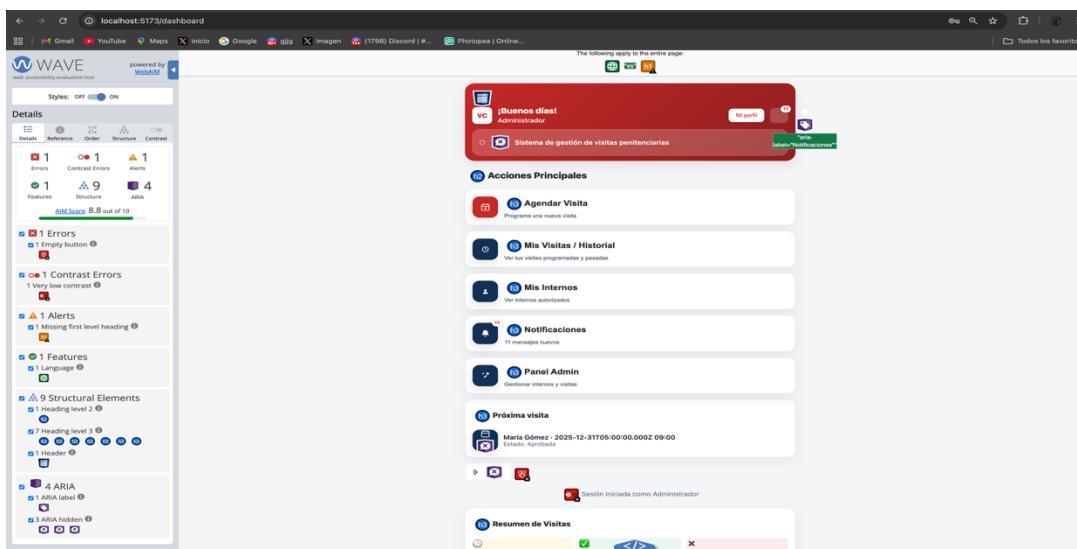
- 1 error: un botón vacío (*empty button*) sin texto ni aria-label, lo que hace que un lector de pantalla solo anuncie “button” sin contexto.
- 1 error de contraste: un texto con contraste muy bajo frente al fondo (*very low contrast*).
- 1 alerta: falta un encabezado de primer nivel (*missing first level heading*), por lo que la estructura de la página empieza directamente en niveles inferiores.
- 1 feature: idioma de la página detectado correctamente.
- 9 elementos estructurales: encabezados de nivel 2 y 3 y un <header> correctamente identificados, lo que ayuda a navegar por secciones.
- 4 elementos ARIA: se detecta al menos un aria-label y varios elementos ocultos con aria-hidden.

Interpretación:

El dashboard tiene una estructura general buena (encabezados, header y elementos ARIA), pero WAVE detecta dos puntos a mejorar para accesibilidad real:

1. Añadir un texto accesible al botón vacío (por ejemplo, el botón de cerrar sesión o de perfil) para que el lector de pantalla anuncie su función.
2. Ajustar el contraste del texto que se marcó como muy bajo para cumplir con los niveles mínimos de WCAG.

Con esos ajustes, el panel sería más usable para personas con baja visión y usuarios que navegan solo con teclado/lector de pantalla



2.3 Pantalla: Historial de visitas

Herramienta: WAVE Web Accessibility Evaluation Tool

URL: <http://localhost:5173/history>

En la tabla de historial de visitas, WAVE reporta:

- 3 errores: todos por *missing form label* (filtros o controles sin etiqueta asociada correctamente).
- 2 alertas:
 - 1 control <select> sin etiqueta clara (*select missing label*),
 - 1 alerta de ausencia de regiones de página (*no page regions*).
- 1 feature: idioma de la página configurado.
- 10 elementos estructurales:
 - 1 tabla de datos correctamente identificada,
 - 8 celdas de encabezado (<th>) marcadas como encabezados de columna,
 - 1 encabezado de nivel 1 para el título “Historial de visitas”.
- 58 elementos ARIA ocultos: uso intensivo de aria-hidden en íconos y elementos decorativos, lo que evita ruido innecesario para el lector de pantalla.

Interpretación:

La pantalla de historial está bien estructurada como tabla de datos, con encabezados de columna y un <h1> principal, lo que facilita la lectura por tecnologías asistivas. El principal problema son las etiquetas faltantes en algunos filtros y controles de la parte superior.

A nivel de mejora, bastaría con:

- Asociar correctamente las etiquetas visibles de los filtros (Filtrar por visitante, Filtrar por interno, etc.) con los controles de formulario correspondientes.
- Definir una región principal de contenido si
- se quiere mejorar aún más la navegación por atajos de lector de pantalla

Visitante	Interno	Fecha	Hora	Duración	Estado	Acciones
Maholy	Maria Gomez	2025-12-31	09:00	60 min	APPROVED	Motivo personal
Maholy Mera	Kevin Gil	2025-12-25	23:59	60 min	APPROVED	Entrega de papeles
Maholy	Maria Gomez	2025-12-26	12:30	60 min	APPROVED	Desearte Feliz Navidad
Kevin	Juan Perez	2025-12-20	08:00	60 min	PENDING	
Administrador	Matheo Gines	2025-12-05	16:56	60 min	APPROVED	
Maholy	Maria Gomez	2025-12-05	10:30	60 min	PENDING	Visita conyugal
Kevin	Juan Perez	2025-12-04	09:00	120 min	PENDING	visitas de 2 horas nueva
Administrador	Maria Gomez	2025-12-04	09:00	60 min	PENDING	

3. Pruebas usabilidad usuarios reales

Además de las pruebas técnicas y automáticas, se realizó una prueba de usabilidad con una usuaria real, que probó la aplicación VisiControl en presencia del grupo. Durante la sesión se le pidió que navevara por la app, creara una cuenta, iniciara sesión y explorara las funcionalidades básicas, incluyendo el cambio de foto de perfil.

A partir de esta prueba, la usuaria comentó dos puntos específicos de mejora:

1. Cambio de foto de perfil:

Al intentar actualizar su foto, notó que la aplicación no muestra un recuadro o marco para encuadrar la imagen antes de guardarla. Esto hace que la foto pueda quedar mal cortada o descentrada. La sugerencia fue incorporar un selector o recortador de imagen que permita visualizar y ajustar cómo quedará la foto final antes de confirmarla.

2. Validación de contraseñas al registrarse:

Al crear una cuenta, la usuaria escribió la contraseña y la repitió correctamente, pero el sistema mostraba el mensaje de que “las contraseñas no coincidían”. Durante la prueba se revisó este comportamiento junto con el desarrollador y se identificó un problema en la validación de esos campos. El error se corrigió en el momento, asegurando que ahora, cuando ambas contraseñas son iguales, el registro se complete normalmente.

Más allá de estos dos detalles puntuales, la valoración general de la usuaria fue muy positiva. Destacó que:

- La aplicación le resultó fácil de entender desde el inicio, sin necesidad de explicaciones técnicas.
- El diseño es limpio y sin ruido visual, con colores bien elegidos que ayudan a diferenciar secciones sin confundir.
- La navegación le pareció clara e intuitiva, incluso pensando en personas que no están muy familiarizadas con la tecnología.
- Consideró que la función principal de la app es realmente útil para familiares de personas privadas de libertad, ya que facilita un proceso que normalmente es largo, confuso o burocrático.

Aquí debería haber un link del video que se grabó pero se dañó el archivo, perdon miss ☺

4. Pruebas unitarias con Jest – Helpers de visitas

Para validar la lógica de formato y metadatos de las visitas en el backend de VisiControl se implementaron pruebas unitarias usando Jest sobre el módulo visits.service.ts.

- Herramienta: Jest + ts-jest
- Ubicación del código de pruebas:
src/backend/src/tests/visits-helpers.test.ts
- Comando de ejecución:
Desde la carpeta src/backend se ejecutó: npm test
- Alcance de las pruebas:

Se probaron específicamente los *helpers* que no dependen directamente de la base de datos:

1. formatDateForText(raw)
 - Verifica que una fecha Date se formatee en el formato dd/MM/yyyy.
 - Valida que, cuando el valor es null o undefined, la función devuelva una cadena vacía.
 2. formatTimeForText(raw)
 - Verifica que una hora en string como "9:5" se normalice a "09:05".
 - Comprueba el formato correcto cuando se pasa un objeto Date.
 - Valida que para valores nulos o indefinidos se devuelva una cadena vacía.
 3. formatStatusLabel(status)
 - Mapea estados internos del sistema (PENDING, APPROVED, REJECTED, CANCELLED) a etiquetas de texto en español:
Pendiente, Aprobada, Rechazada, Cancelada.
 - Verifica el comportamiento con estados desconocidos y con valores nulos, devolviendo el texto original o Pendiente por defecto.
 4. buildVisitMeta(v)
 - Comprueba que el objeto de entrada (id, visitante, interno, fecha, hora, estado, etc.) se transforme correctamente al objeto de metadatos que usa el sistema de notificaciones.
 - Valida que:
 - status_label se genere usando internamente formatStatusLabel.
 - Cuando no se envía inmate_id, el campo se complete con null.
- Resultado:

Al ejecutar npm test en la carpeta backend, Jest reporta:

- 1 test suite pasada
- 9 tests pasados

- Tiempo de ejecución aproximado: 2–3 segundos

Esto confirma que las funciones auxiliares de formato y construcción de metadatos para visitas están correctamente implementadas y cubiertas por pruebas unitarias automatizadas. Estos helpers son críticos porque se usan tanto en la validación de horarios como en los mensajes enviados al usuario a través del sistema de notificaciones.

```
[kevings0712@MacBook-Pro-de-Kevin-2 backend % npm test

> backend@1.0.0 test
> jest

PASS  src/tests/visits-helpers.test.ts
  Helpers de formato de visitas
    formatDateForText
      ✓ formatea una Date a dd/MM/yyyy (3 ms)
      ✓ devuelve cadena vacía cuando el valor es null/undefined
    formatTimeForText
      ✓ formatea una hora en string HH:mm
      ✓ formatea una Date a HH:mm
      ✓ devuelve cadena vacía cuando el valor es null/undefined
    formatStatusLabel
      ✓ mapea estados conocidos a etiquetas en español (1 ms)
      ✓ devuelve el texto original (o 'Pendiente') para estados desconocidos (1
ms)
    buildVisitMeta
      ✓ construye el objeto meta con los campos correctos (1 ms)
      ✓ rellena inmate_id con null si no viene definido

Test Suites: 1 passed, 1 total
Tests:       9 passed, 9 total
Snapshots:  0 total
Time:        2.244 s, estimated 3 s
Ran all test suites.
```

5. Pruebas de rendimiento – Inicio de sesión (10 usuarios simulados)

Herramienta: Apache JMeter

Escenario probado: carga sobre el endpoint de autenticación POST /api/auth/login del backend de VisiControl.

Objetivo: verificar cómo responde la API cuando 10 usuarios intentan iniciar sesión casi al mismo tiempo.

Configuración de la prueba:

- Thread Group con 10 usuarios (threads).
- Ramp-up: 10 segundos (aprox. 1 usuario por segundo).
- Cada usuario ejecuta 1 petición POST a /api/auth/login con credenciales válidas.
- Se agregó un HTTP Header Manager con Content-Type: application/json.
- Se utilizó el listener Summary Report para ver los tiempos de respuesta y el throughput.

Resultados principales (Summary Report):

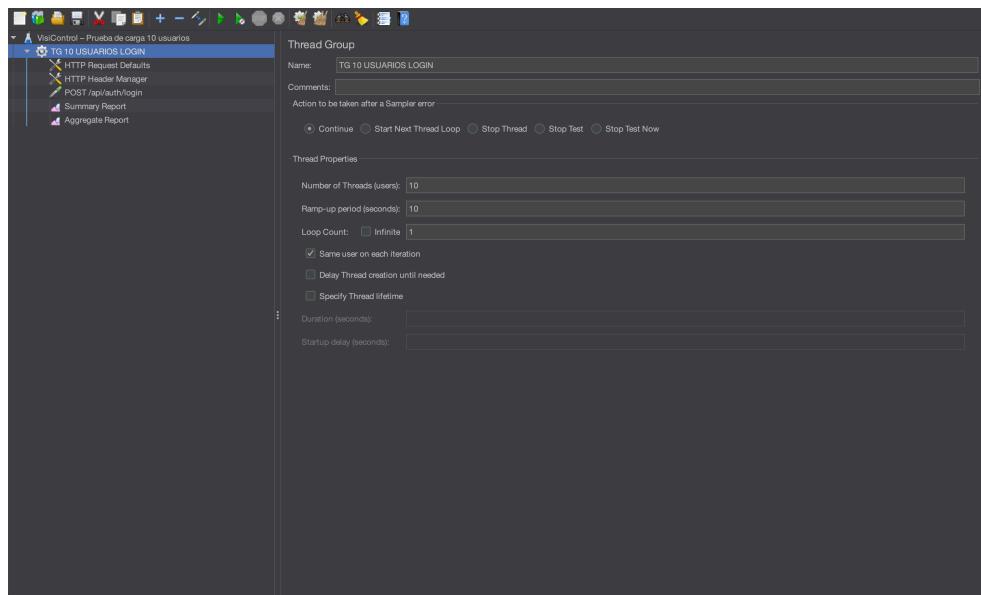
- # Samples: 10 (un login por cada usuario simulado).

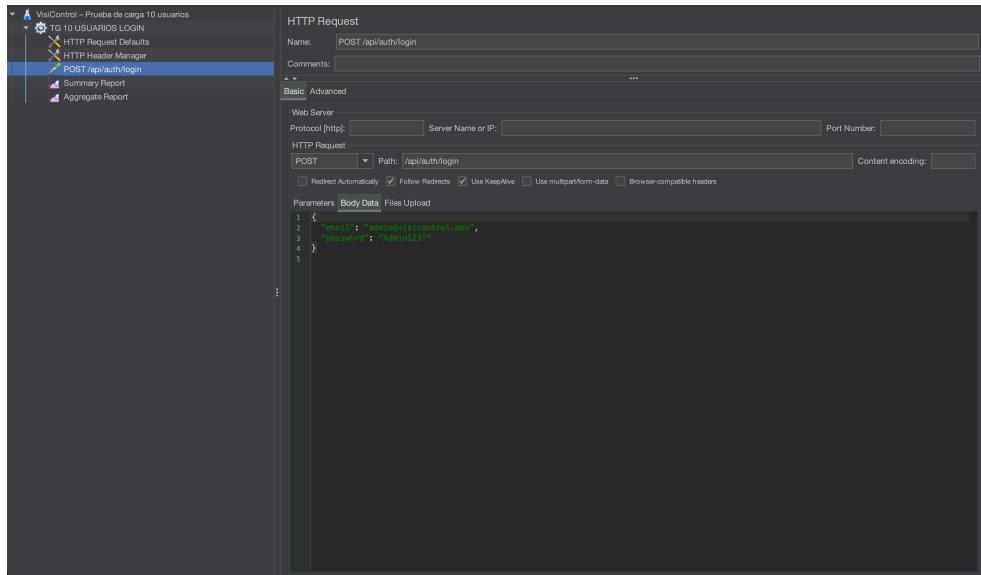
- Average: 75 ms
Tiempo promedio de respuesta del login.
- Min: 60 ms
Mejor tiempo medido.
- Max: 142 ms
Peor tiempo medido, aún por debajo de 150 ms.
- Std. Dev.: ~22 ms
La variación entre las respuestas es baja.
- Error %: 0.00 %
Ninguna de las peticiones falló (todas devolvieron código 2xx).
- Throughput: 1.1 requests/segundo
Promedio de peticiones procesadas por segundo durante la prueba.
- Received KB/sec: ~17.92 KB/s
Datos recibidos desde el servidor.
- Sent KB/sec: ~0.28 KB/s
Datos enviados al servidor (body del login y headers).

Conclusión de la prueba de rendimiento:

Con 10 usuarios simulados haciendo login casi al mismo tiempo, el backend responde en un promedio de 75 ms por petición, sin errores y con tiempos máximos inferiores a 150 ms.

Para el contexto del proyecto (sistema académico con volumen moderado), estos resultados indican que el endpoint de autenticación soporta correctamente esta carga y tiene margen suficiente para más usuarios concurrentes.





visicontrol.jmx (/Users/kevings0712/Downloads/apache-jmeter-5.6.3/bin/visicontrol.jmx) - Apache JMeter (5.6.3)

Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
POST /api/au...	10	75	60	142	22.36	0.00%	1.1/sec	17.92	0.28	16486.0
TOTAL	10	75	60	142	22.36	0.00%	1.1/sec	17.92	0.28	16486.0

Aggregate Report

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
POST /api/au...	10	75	69	73	73	142	60	142	0.00%	1.1/sec	17.92	0.28
TOTAL	10	75	69	73	73	142	60	142	0.00%	1.1/sec	17.92	0.28

6. Pruebas de integración

Además de las pruebas unitarias, se implementaron pruebas de integración sobre el backend de VisiControl para validar el comportamiento real de la API cuando se conectan varias capas al mismo tiempo (rutas, controladores, middlewares, base de datos y generación de tokens).

Para esto se creó el archivo `src/tests/auth.integration.test.ts`, utilizando Jest junto con Supertest. En lugar de probar funciones aisladas, estas pruebas levantan la aplicación Express (app) en modo de prueba y le envían peticiones HTTP simuladas, como si fuera el frontend.

Los escenarios cubiertos fueron:

1. Login correcto (POST /api/auth/login)
 - o Se envía una solicitud POST al endpoint de login con las credenciales válidas de un usuario registrado (por ejemplo, el administrador de pruebas).
 - o La prueba verifica que la API responda con código 200 OK, que el campo ok sea true y que en la respuesta exista un token JWT y los datos básicos del usuario (id, name, email, role).
 - o Con esto se comprueba que funcionan en conjunto: la validación de credenciales, la consulta a la base de datos y la generación del token de autenticación.
2. Login con credenciales inválidas
 - o Se envía otra petición POST /api/auth/login, pero esta vez con una contraseña incorrecta.
 - o La prueba espera que el servicio responda con código 401 Unauthorized, ok: false y un mensaje de error indicando que las credenciales no son válidas.
 - o Esto confirma que la API maneja correctamente los intentos fallidos de autenticación y no entrega tokens cuando los datos no coinciden.

En conjunto, estas pruebas de integración aseguran que el flujo completo de autenticación funcione como se espera y que, si en algún momento se cambia la lógica interna, exista una red de seguridad automatizada que detecte errores antes de desplegar la aplicación.

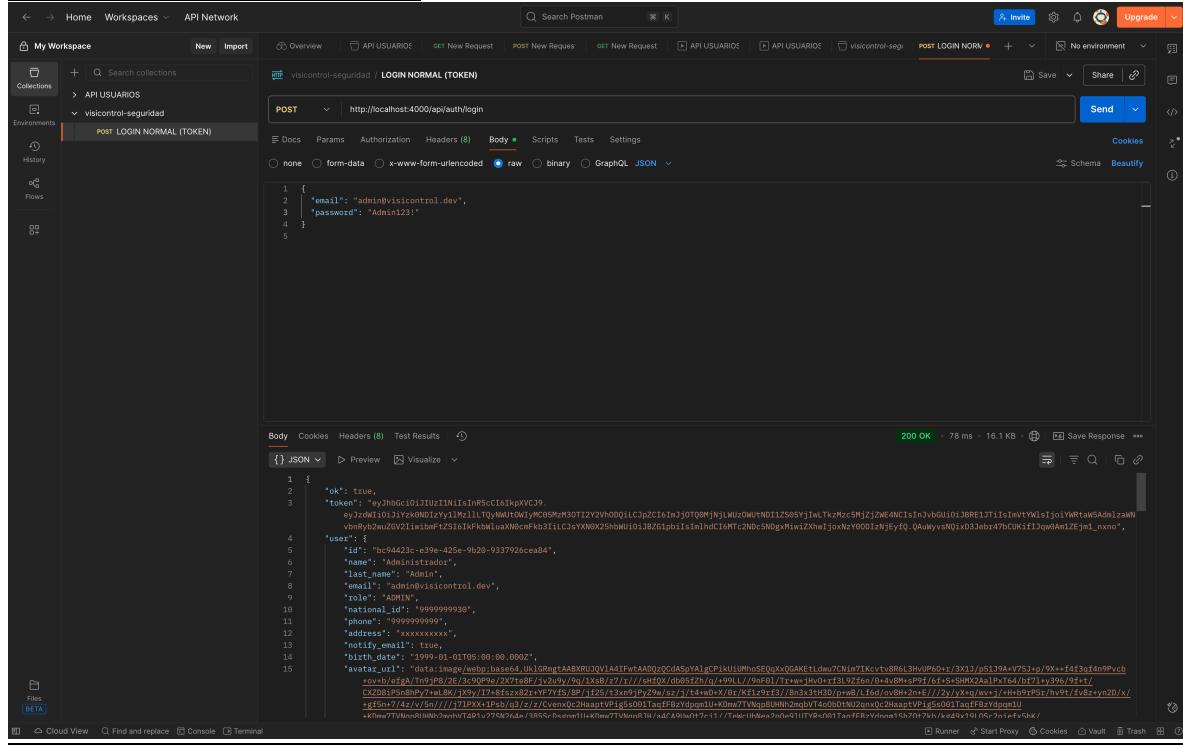
```
[kevings0712@MacBook-Pro-de-Kevin-2 backend % npm test
> backend@1.0.0 test
> jest

PASS  src/tests/visits-helpers.test.ts
PASS  src/tests/auth.integration.test.ts

Test Suites: 2 passed, 2 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        3.804 s
Ran all test suites.
kevings0712@MacBook-Pro-de-Kevin-2 backend %
```

7. Pruebas de seguridad

Caso 1 – Login normal (control):



The screenshot shows the Postman interface with a collection named 'visicontrol-seguridad' and a specific request named 'LOGIN NORMAL (TOKEN)'. The request is a POST to `http://localhost:4000/api/auth/login`. The JSON body contains:

```
1 {
2   "email": "admin@visicontrol.dev",
3   "password": "Admin123!"
4 }
```

The response is a 200 OK status with a response time of 78 ms and a size of 16.1 KB. The response body includes an 'ok' key set to true and a 'token' key containing a long JWT token.

Body:

```
{
  "email": "admin@visicontrol.dev",
  "password": "Admin123!"}
```

Resultado: 200 OK. La API devuelve ok: true, los datos del usuario administrador y un token JWT válido.

Conclusión: el flujo normal de autenticación funciona y sirve como base de comparación para las demás pruebas.

Caso 2 – Intento de inyección SQL en el password:

The screenshot shows the Postman interface with a dark theme. In the left sidebar, there's a collection named 'visicontrol-seguridad' containing two items: 'POST LOGIN NORMAL (TOKEN)' and 'POST LOGIN - SQL INJECTION'. The 'POST LOGIN - SQL INJECTION' item is selected. The main workspace shows a POST request to 'http://localhost:4000/api/auth/login'. The 'Body' tab is selected, showing the following JSON payload:

```
1 {
2   "email": "admin@visicontrol.dev",
3   "password": "' OR '1='1"
4 }
```

Below the body, the response is shown as a 401 Unauthorized status with a response time of 72 ms and a size of 325 B. The response body is:

```
1 {
2   "ok": false,
3   "message": "Credenciales inválidas"
4 }
```

Body malicioso:

```
{
  "email": "admin@visicontrol.dev",
  "password": "' OR '1='1"
}
```

Resultado: 401 Unauthorized con respuesta {"ok": false, "message": "Credenciales inválidas"}.

Conclusión: la API no concatena directamente el password en consultas SQL y trata la entrada como un valor normal.

El intento de inyección no permite autenticarse ni devuelve errores de SQL, por lo que este endpoint no es vulnerable a una inyección simple de este tipo.

Caso 3 - Prueba de inyección en filtros de visitas

The screenshot shows a Postman collection named 'visicontrol-seguridad' with a 'GET INYECCION EN FILTRO VISITAS' request. The URL is `http://localhost:4000/api/visits?date=2025-12-04' OR '1='1`. The 'Authorization' tab shows 'Bearer Token' selected, with a placeholder 'Token'. The response body is JSON:

```
{ "ok": false, "message": "Error interno" }
```

. The status bar indicates a 500 Internal Server Error.

Análisis:

- La aplicación no devuelve resultados masivos ni datos de la base, a pesar del intento de inyección.
- El mensaje de error es genérico y no revela detalles de la consulta SQL ni del stack interno.

Conclusión: el intento de inyección en el filtro no permite acceder a más información, pero genera un error interno.

Se recomienda como mejora futura validar el formato de la fecha antes de ejecutar la consulta y devolver un 400 Bad Request en lugar de 500, para que el error quede mejor controlado.

Caso 4 - Acceso a endpoint protegido SIN token

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'viscontrol-seguridad' which contains several requests: 'POST LOGIN NORMAL (TOKEN)', 'POST LOGIN - SQL INJECTION', 'GET INYECCION EN FILTRO VISITAS', and 'GET Acceso a endpoint protegido SIN token'. The last request is selected. The main panel shows a GET request to 'http://localhost:4000/api/auth/me'. The 'Params' tab is selected, showing a single parameter 'Key' with value 'Value'. Below the request, the 'Test Results' section displays a JSON response:

```
{ "ok": false, "message": "No token" }
```

. The status bar at the bottom indicates a 401 Unauthorized error with a response time of 3 ms.

Resultado: 401 Unauthorized con {"ok": false, "message": "No token"}.

Conclusión: la API obliga a enviar un JWT para acceder a recursos protegidos, cumpliendo con el control de autenticación básica.

Caso 5- Token invalido

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named 'visicontrol-seguridad' containing several requests. One request is selected: 'GET token invalido' with the URL `http://localhost:4000/api/auth/me`. The 'Body' tab is selected, showing the response body as JSON:

```
{ } JSON
```

The response status is 401 Unauthorized, and the response body is:

```
1 | {  
2 |   "ok": false,  
3 |   "message": "Token invalido"  
4 | }
```

Resultado: 401 Unauthorized con {"ok": false, "message": "Token inválido"}.

Conclusión: el backend valida la firma del JWT y no acepta tokens manipulados o inventados, lo que reduce el riesgo de suplantación de identidad.

Caso 6 - Usuario normal accediendo a endpoint de Admin

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections like 'API USUARIOS' and 'visicontrol-seguridad'. The main workspace displays a request for 'visicontrol-seguridad' titled 'Usuario normal accediendo a endpoint de ADMIN'. The request method is 'GET' to 'http://localhost:4000/api/visits/admin'. The 'Authorization' tab is selected, showing 'Bearer Token' as the auth type with a placeholder 'Token'. Below the request, the response section shows a 403 Forbidden status with a JSON body: { "ok": false, "message": "Solo administradores" }. The bottom navigation bar includes 'Cloud View', 'Find and replace', 'Console', 'Terminal', 'Runner', 'Start Proxy', 'Cookies', 'Vault', and 'Trash'.

- Body: credenciales de un usuario sin rol administrador.
- Prueba de acceso:
 - Endpoint: GET /api/visits/admin
 - Auth: Authorization: Bearer <token_usuario_normal>
- Resultado: 403 Forbidden con {"ok": false, "message": "Solo administradores"}.
- Conclusión: además de la autenticación, la API aplica control de autorización por rol.
Un usuario autenticado pero sin rol ADMIN no puede acceder a las rutas de administración de visitas.

8. Tipos de pruebas aplicadas

En el proyecto VisiControl no se hizo solo una prueba aislada, sino una combinación de pruebas **unitarias, de integración y funcionales**, con el objetivo de cubrir desde la lógica interna del backend hasta el comportamiento real que percibe el usuario.

- **Pruebas unitarias**

Las pruebas unitarias se aplicaron sobre funciones específicas del backend, usando **Jest** y **ts-jest**. En particular, se probaron los helpers del módulo visits.service.ts (`formatDateForText`, `formatTimeForText`, `formatStatusLabel` y `buildVisitMeta`). Estas funciones se encargan de formatear fechas y horas, traducir estados internos a textos en español y construir metadatos para el sistema de notificaciones. Al ejecutarse con `npm test` en la carpeta `src/backend`, Jest reportó una suite y 9 tests pasando, lo que confirma que esa lógica crítica funciona correctamente de forma aislada.

- **Pruebas de integración**

Además de las pruebas manuales con Postman, se implementó un **test de integración automatizado** usando **Jest** y **Supertest** en el archivo `src/tests/auth.integration.test.ts`. En lugar de probar funciones aisladas, este test levanta la app Express de VisiControl en modo de prueba y envía peticiones HTTP reales contra el endpoint de autenticación.

Se cubren dos escenarios principales:

1. **Login correcto:** se llama a `POST /api/auth/login` con credenciales válidas y se verifica que la API responda 200 OK, `ok: true`, y devuelva un token JWT junto con los datos del usuario.
2. **Login inválido:** se envía el mismo endpoint con una contraseña incorrecta y se espera 401 Unauthorized, `ok: false` y un mensaje de error.

Con esto se valida que, integrados, los middlewares, controladores y acceso a PostgreSQL funcionan de forma coherente para el flujo de inicio de sesión.

- **Pruebas funcionales**

A nivel funcional se revisó el sistema tal como lo usaría una persona. Primero, mediante herramientas de accesibilidad (Lighthouse y WAVE) se evaluó el comportamiento de varias pantallas clave: login, dashboard, agenda de visitas e historial. Se validó que los formularios tuvieran labels, que el contraste de colores fuera adecuado y que la estructura de encabezados fuese legible para lectores de pantalla.

Además, se planificó la grabación de un **video con usuarios reales utilizando la aplicación**, para observar si logran completar tareas como iniciar sesión, agendar una visita o revisar el historial sin necesidad de ayuda. Esta parte complementa las pruebas técnicas con evidencia de uso real.

En conjunto, estas tres capas de pruebas (unitarias, integración y funcionales) dan una visión más completa del estado de calidad de VisiControl: se valida tanto la lógica interna del código como el comportamiento del sistema cuando se conecta con la base de datos y cuando es usado por personas reales.