Desarrollo de sistema de trazabilidad alimentaria confiable basado en Blockchain

Anexo 3 – Especificación de diseño

Trabajo de Fin de Máster

MÁSTER EN INGENIERÍA INFORMÁTICA (Semi-presencial)

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autor

Konstantin Danielov Kostandev

Tutor

Vidal Moreno Rodilla

Tabla de contenido

Tabla de contenido I
Índice de Ilustraciones II
1 Arquitectura global del sistema 1
1.1 Incremento 1: Modelado de contratos inteligentes 2
1.2 Incremento 2: Servidor intermediario para comunicación con blockchain 5
1.2.1 Modelo-Vista-Controlador 5
1.2.2 Modelo de Datos 6
1.2.3 API HTTP 7
1.3 Incremento 3: Interfaz gráfica de usuario 9
1.3.1 Pantalla principal 10
1.3.2 Pantalla de solicitud de alta de usuario 11
1.3.3 Pantalla de inicio de sesión 12
1.3.4 Pantalla principal tras iniciar sesión 13
1.3.5 Pantalla de detalle de operaciones 14
1.3.6 Pantallas de alta de operaciones 15
1.3.7 Pantallas de modificación de datos del usuario 16
1.3.8 Pantalla de lista de usuarios 17
1.3.9 Pantalla de detalle de usuario 18
1.3.10 Securización del enrutamiento 19
1.3.11 Validación de la información 20
1.3.12 Notificaciones para el usuario 20

Índice de Ilustraciones

Ilustración 1. Arquitectura global. 1
Ilustración 2. Patrón de diseño de software MVC. 5
Ilustración 3. Diagrama Entidad-Relación. 6
Ilustración 4. Diseño de la pantalla principal. 10
Ilustración 5. Captura de la pantalla principal. 10
Ilustración 6. Diseño de las páginas de solicitud de alta. 11
Ilustración 7. Captura de la pantalla de alta de industria. 11
Ilustración 8. Diseño de pantalla de inicio de sesión. 12
Ilustración 9. Captura de pantalla de inicio de sesión. 12
Ilustración 10. Diseño pantalla principal una vez autenticado. 13
Ilustración 11. Captura de la pantalla principal de un usuario tipo ganadero. 13
Ilustración 12. Diseño de pantalla de detalle de operaciones. 14
Ilustración 13. Captura de pantalla de detalle de operaciones. 14
Ilustración 14. Diseño de pantalla de alta de operación. 15
Ilustración 15. Captura de pantalla de alta de operación. 15
Ilustración 16. Diseño pantalla de modificación de datos. 16
Ilustración 17. Captura de la pantalla de modificación de datos del usuario. 16
Ilustración 18. Diseño de pantalla de lista de usuarios. 17
Ilustración 19. Captura de pantalla de lista de usuarios. 17
Ilustración 20. Diseño de pantalla de detalle de usuario. 18
Ilustración 21. Captura de pantalla de detalle de usuario. 18
Ilustración 22. Captura de pantalla a una ruta sin autorización. 19
Ilustración 23. Captura de pantalla para ruta inexistente. 19
Ilustración 24. Validaciones de los campos de los formularios. 20
Ilustración 25. Captura de mensaje NOK. 20
Ilustración 26. Captura de mensaje OK. 21

1 Arquitectura global del sistema

El sistema planteado se dividirá en tres bloques principales, correspondientes a cada uno de los incrementos definidos, que interactuarán entre sí para formar el sistema completo. El flujo de la información comenzará en el servidor web (interfaz gráfica de usuario) donde los actores podrán realizar solicitudes a la API del servidor intermediario y este será el encargado de gestionar las transacciones en la blockchain.

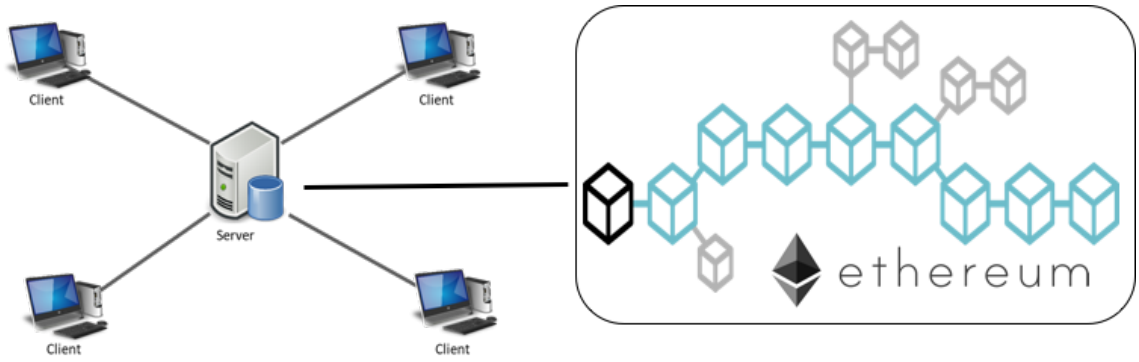

Ilustración 1. Arquitectura global.

En este proyecto se ha tomado un planteamiento en el que únicamente existe una cartera (*wallet*) manejada por el servidor para conectarse a la blockchain por sencillez, para el caso de estudio, perdiendo de esta manera una de las grandes ventajas de la blockchain: que es un sistema distribuido y descentralizado. En un entorno real o de producción, cada cliente debería tener su propia cartera y conectarse directamente a la blockchain desde el navegador para firmar sus propias transacciones.

Para el desarrollo existen herramientas que permiten crear blockchain locales, ofreciendo una enorme comodidad a la hora de desarrollar ya que la velocidad de las transacciones es considerablemente mayor a la que se obtiene de las redes oficiales, además de no necesitar utilizar criptodivisas reales, que tienen un alto valor económico. Además de las redes oficiales existen redes para pruebas, en las que probar los contratos inteligentes en un entorno real, pero también se obtienen tiempos de respuesta por transacción muy altos. En este caso se utilizará una red blockchain local para poner a prueba el sistema, pero para un entorno real o de producción, tras el testeo correspondiente en las redes habilitadas para tal uso se debería utilizar la red principal.

1.1 Incremento 1: Modelado de contratos inteligentes

Los contratos inteligentes serán los encargados de generar las transacciones para almacenar la información de los procesos de los usuarios en la blockchain.

Se debe tener en cuenta que una blockchain no es una base de datos, ya que cuanta más información se almacena en el contrato inteligente más gas (tarifa de transacción) costarán las transacciones. Los métodos de los contratos inteligentes que recuperan información de la blockchain no tienen coste, pero los métodos que almacenan información sí.

Todas las transacciones pueden generan un log o registro de eventos, que es una estructura de datos indexada en la que se puede escribir la información de manera más "barata" mediante los eventos de Solidity. Los contratos inteligentes no pueden acceder a los datos del log después de crearse, pero estos datos pueden ser accedidos desde fuera de la blockchain de forma eficiente, pudiendo utilizar hasta tres parámetros para filtrar la información del log.

Con esta premisa se crea un único contrato inteligente (llamado `Trazabilidad`) en el que se almacenará la información mínima necesaria para el funcionamiento del sistema, enviando el resto de información al log mediante eventos.

El contrato inteligente contará con cuatro atributos:

Tipo	Visibilidad	Atributo	Descripción
`address`	`public`	`owner`	Almacena la dirección que despliega el contrato.
`uint256`	`private`	`s_numLote`	Almacena el ultimo identificador de lote.
`uint256`	`private`	`s_numPieza`	Almacena el último identificador de pieza.
`mapping(uint256 =>uint256)`	`private`	`s_piezaALote`	Relación que determina a qué lote pertenece cada pieza.

El tipo de visibilidad `private` indica que estos atributos no pueden escribirse ni leerse desde fuera del contrato inteligente, mientras que el tipo de visibilidad `public` sí lo permite.

Para poder recuperar el valor de los atributos privados se hace uso de los siguientes métodos:

Método	Tipo	Descripción
`getNumLotes()`	`public view`	Devuelve el valor del atributo `s_numLote`.
`getNumPiezas()`	`public view`	Devuelve el valor del atributo `s_numPieza`.
`getLoteOfPieza(uint256 _idPieza)`	`public view`	Devuelve el lote al que pertenece la pieza.

Los métodos con la palabra clave `view` indican que el método únicamente devuelve información de la blockchain y no escribirá ningún valor.

Para garantizar la seguridad de la información almacenada se ha creado el modificador `onlyOwner`, que restringe los métodos a los que se aplica para que solo puedan ser ejecutados por la dirección almacenada en `owner`, haciendo que nadie más pueda incluir operaciones en la blockchain a través de este contrato. Este modificador se ha aplicado a todos los métodos del contrato.

```
    modifier onlyOwner() {
        require(owner == msg.sender, "Must be owner");
        _;
    }
```

Si una dirección que no sea `owner` ejecuta un método que tenga este modificador recibirá el mensaje de error `"Must be owner"`.

Para el almacenamiento de la información en el log se utilizan los siguientes eventos:

Evento	Descripción

`NuevaOperacionGanadero(`
`uint256 indexed numLote,`
`string indexed idGanadero,`
`uint8 indexed tipoOperacion,`
`string info,`
`uint256 timestamp`
`)`
	Evento emitido siempre que se introduzca una operación de un operario tipo ganadero en la blockchain.

`NuevaOperacionIndustriaLote(`
`uint256 indexed numLote,`
`string indexed idIndustria,`
`uint8 indexed tipoOperacion,`
`string info,`
`uint256 timestamp`
`)`
	Evento emitido siempre que se introduzca una operación de un operario tipo industria relacionado con un lote en la blockchain.

`NuevaOperacionIndustriaPieza(`
`uint256 indexed numPieza,`
`string indexed idIndustria,`
`uint8 indexed tipoOperacion,`
`string info,`
`uint256 timestamp`
`)`
	Evento emitido siempre que se introduzca una operación de un operario tipo industria relacionado con una pieza en la blockchain.

Los atributos con el literal `indexed` indican que son parámetros por los que se puede buscar para recuperar el log, devolviendo los registros que coincidan con los parámetros introducidos o todos los registros si el parámetro es `null`. Esta búsqueda es muy potente, ya que no se limita a un único valor, es capaz de recibir, por ejemplo, un array de valores y devolver los registros que tengan alguno de los valores del array en el parámetro especificado.

Dado que se pueden almacenar diferentes tipos de operaciones, el campo `info` se rellenará con la información específica de cada operación en formato JSON.

Por último, se añade un campo `timestamp` para poder ordenar los registros cronológicamente cuando se recuperen.

Para la emisión de estos eventos el contrato cuenta con tres métodos:

Método	Tipo	Descripción

`registrarNuevaOperacionGanadero(`
`uint256 _idLote,`
`string memory _idGanadero,`
`uint8 _tipoOperacion,`
`string memory _info`
`)`
	`external`	Emite el evento `NuevaOperacionGanadero` para registrar la operación.

`registrarNuevaOperacionIndustriaLote(`
`uint256 _numLote,`
`string memory _idIndustria,`
`uint8 _tipoOperacion,`
`uint256 _numNuevasPiezas,`
`string memory _info`
`)`
	`external`	Emite el evento `NuevaOperacionIndustriaLote` para registrar la operación. Cuando es una operación de tipo sacrificio actualiza los atributos del contrato.

`registrarNuevaOperacionIndustriaPieza(`
`uint256 _idPieza,`
`string memory _idIndustria,`
`uint8 _tipoOperacion,`
`string memory _info`
`)`
	`external`	Emite el evento `NuevaOperacionIndustriaPieza` para registrar la operación.

Los métodos de tipo `external` son aquellos que únicamente se pueden ejecutar desde fuera del contrato inteligente.

Con este planteamiento donde únicamente el `owner` puede ejecutar los métodos de inserción de datos en la blockchain se está centralizando la aplicación. Se ha planteado de esta manera por sencillez, pero en un entorno real o de producción habría que proporcionar un método para dar permiso a ciertas direcciones aparte del `owner` y descentralizar el sistema.

1.2 Incremento 2: Servidor intermediario para comunicación con blockchain

El servidor intermediario es el elemento central de la arquitectura ya que será el encargado de conectar la interfaz de usuario con las transacciones de la blockchain, por lo que contendrá toda la lógica de negocio. Las funciones del servidor serán las siguientes:

- Gestionar la interacción con el nodo de Ethereum, tanto para generar transacciones que carguen información como para recuperar la información de los eventos y poder presentársela al usuario a través de la interfaz gráfica.
- Gestionar la base de datos MongoDB, donde se almacenarán los datos de los usuarios.
- Securizar el acceso los recursos que ofrece, para que únicamente puedan acceder usuarios autenticados a la información.
- Atender las peticiones HTTP que reciba a través de la API o de la interfaz gráfica.

El servidor se ha creado utilizando nodeJS para permitir la utilización de Javascript en backend, con el framework ExpressJS. La conexión con la blockchain se realiza a través de la biblioteca de funciones Ethers.

El patrón de diseño de software utilizado para el servidor intermediario es el Modelo-Vista-Controlador.

1.2.1 Modelo-Vista-Controlador

Modelo-Vista-Controlador (MVC) es un patrón de diseño de software que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos [27].

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, sobre multitud de lenguajes y plataformas de desarrollo [27].

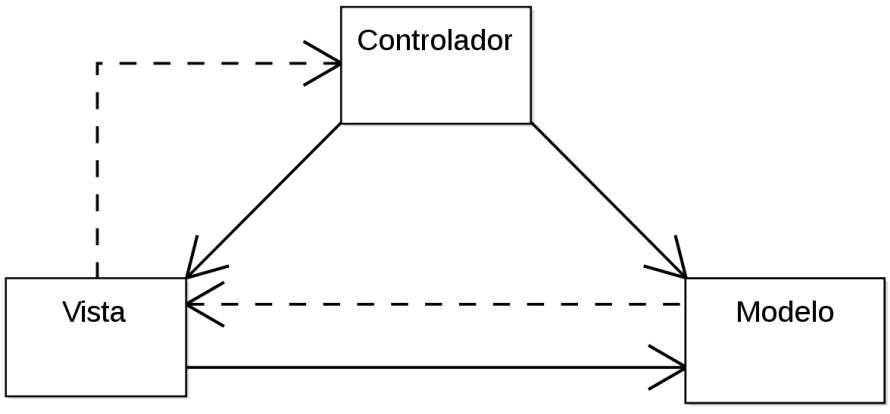

Ilustración 2. Patrón de diseño de software MVC.

El Modelo contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia [27].

La Vista (o interfaz de usuario) que compone la información que se envía al cliente y los mecanismos de interacción con este [27].

El controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno [27].

El flujo que se sigue con este tipo de patrones es generalmente el siguiente [27]:

1. El usuario interactúa con la interfaz gráfica (vista) de alguna forma.
2. El controlador recibe la notificación de acción solicitada por el usuario y la gestiona.
3. El controlador accede al modelo para llevar a cabo la acción solicitada por el usuario.
4. El controlador devuelve la información a la vista, que será la encargada de mostrarla de manera correcta.
5. La interfaz de usuario espera nuevas interacciones del usuario.

1.2.2 Modelo de Datos

Esta sección corresponde al Modelo del patrón MVC.

- **[RI-001] Usuario**: Se debe almacenar la información de los usuarios para su autenticación en el sistema y permitir la interacción con la API.
- **[RI-002] Lote**: Se deben almacenar las operaciones que generan los usuarios referentes a lotes ganaderos.
- **[RI-003] Pieza:** Se deben almacenar las operaciones que generan los usuarios referentes a las piezas.

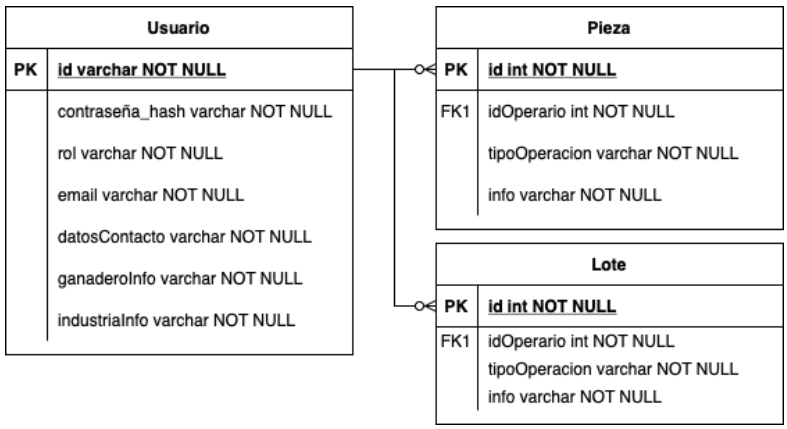

Ilustración 3. Diagrama Entidad-Relación.

La información de la entidad Usuario se almacenará en la base de datos MongoDB, basada en documentos, con formato JSON. La información de las entidades Lote y Pieza se almacenará directamente en la blockchain, ya que es información sensible que debe ser extremadamente difícil de manipular de manera maliciosa.

1.2.3 API HTTP

Esta sección corresponde al Controlador del patrón MVC, donde se definirán las funcionalidades del servidor intermediario para poder comunicarse y brindar la información necesaria a la interfaz gráfica (Vista) a través de una API HTTP.

La estructura de la API se ha dividido por funcionalidades para una mejor organización, existiendo tres grupos diferenciados:

API de autenticación: Funciones para la autenticación de los usuarios. Si no están autenticados únicamente podrán acceder a las rutas públicas.

Método	Ámbito	Ruta	Descripción
POST	Publico	`/auth`	Inicia la sesión del usuario a través de nombre de usuario y contraseña. Si la información es correcta le devuelve un token de acceso y un token de refresco a través de una cookie.
GET	Publico	`/auth/refresh`	Se solicita cuando el token de acceso ha expirado. Devuelve un nuevo token de acceso si el token de refresco no ha expirado.
POST	Público	`/auth/logout`	Cierra la sesión del usuario. Elimina la cookie.

Para la implementación de la API de seguridad se han utilizado JSON Web Token (JWT): un estándar abierto basado en JSON para la creación de tokens de acceso que permiten la propagación de la identidad y privilegios. El token está firmado por la clave del servidor, por lo que tanto el cliente como el servidor son capaces de verificar que el token es legítimo [28].

Para la securización de las rutas privadas se ha implementado un middleware que comprueba si el solicitante manda la cookie con el token de acceso, en caso de no tener cookie se deniega el recurso. Adicionalmente se ha implementado otro middleware que limita el número de intentos de inicio de sesión de los usuarios y deberán esperar cierto tiempo antes de poder volver a intentarlo.

Todas las rutas son públicas para que todos los participantes puedan acceder a la aplicación.

API de usuarios: Funciones para la gestión de los usuarios del sistema.

Método	Ámbito	Ruta	Descripción
GET	Privado	`/users`	Devuelve una lista con todos los usuarios del sistema, excluyendo al administrador y excluyendo las contraseñas de los usuarios.
POST	Público	`/users`	Crea los usuarios que solicitan acceder a la aplicación, de tipo consumidor, ganadero o industria.
PATCH	Privado	`/users`	Modifica los datos de contacto de un usuario.
DELETE	Privado	`/users`	Elimina un usuario.
POST	Privado	`/users/auditor`	Crea un usuario de tipo auditor.
POST	Privado	`/users/getUser`	Devuelve un usuario concreto.

POST	Privado	`/users/changePassword`	Modifica la contraseña de un usuario.

API de trazabilidad: Funciones para interactuar con la blockchain.

Método	Ámbito	Ruta	Descripción
POST	PRIVADO	/trazabilidad/operation	Genera una operación en blockchain en función de los parámetros recibidos.
POST	PRIVADO	/trazabilidad/filter	Devuelve los eventos que satisfacen la búsqueda con los parámetros recibidos.

1.3 Incremento 3: Interfaz gráfica de usuario

Esta sección corresponde a la Vista del patrón MVC, donde se captarán las solicitudes de los usuarios para hacérselas llegar al Controlador y se mostrará la información que este devuelva.

La Vista se ha creado utilizando el framework ReactJS, cuyos puntos fuertes son las páginas de tipo "single page" y la reutilización de los componentes. Con esta biblioteca no será necesario recargar la página cada vez que se quieran actualizar los datos a mostrar gracias a que controla el estado y muestra los componentes actualizados cuando cambia su valor, reacciona a los cambios. Además, se utiliza la biblioteca react-bootstrap que contiene componentes genéricos para React con diseño responsive, facilitando enormemente la creación de la Vista.

Debido a que esta interfaz de usuario debe ser utilizada por participantes que pueden no conocer el funcionamiento de esta, se deben tener en cuenta algunas pautas:

- Crear un diseño consistente, donde los componentes de la interfaz sean fáciles de localizar para el usuario y sigan el mismo patrón.
- Evitar los errores humanos: se deberán colocar comprobaciones en los campos donde el usuario introducirá la información para que no llegue información errónea a la API.
- Mostrar mensajes que informen al usuario de la ruta en la que se encuentra y del resultado de las acciones que realiza.
- La vista debe adaptarse a diferentes tamaños de pantalla para poder utilizarse desde cualquier dispositivo (diseño responsive).

Todas las pantallas siguen el mismo patrón: una barra de navegación en la parte superior que indica la ruta en la que se encuentra el usuario, una barra de información en la parte inferior que mostrará los datos del usuario cuando haya iniciado sesión y el contenido principal de la página que variará en función del usuario y la ruta.

A continuación, se mostrará el diseño de cada una de las pantallas, junto con la información que proporciona y el resultado final de la misma.

1.3.1 Pantalla principal

Esta pantalla es la toma de contacto con los usuarios: debe contener ciertas instrucciones sobre la aplicación y cómo utilizarla.

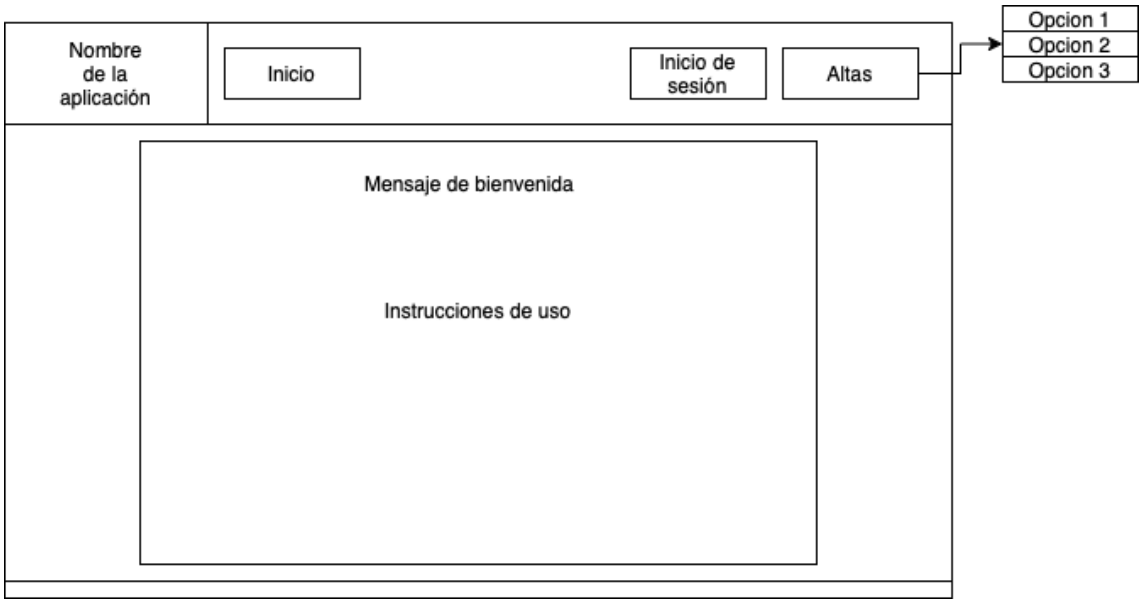

Ilustración 4. Diseño de la pantalla principal.

Se compone de un menú en la barra superior en el que se indican las opciones que tiene un usuario: ir a la página de inicio, iniciar sesión o darse de alta por alguna de las opciones.

Una zona principal que contendrá un mensaje de bienvenida y unas instrucciones para la correcta utilización de la aplicación.

Una barra inferior que estará vacía mientras el usuario no haya iniciado sesión.

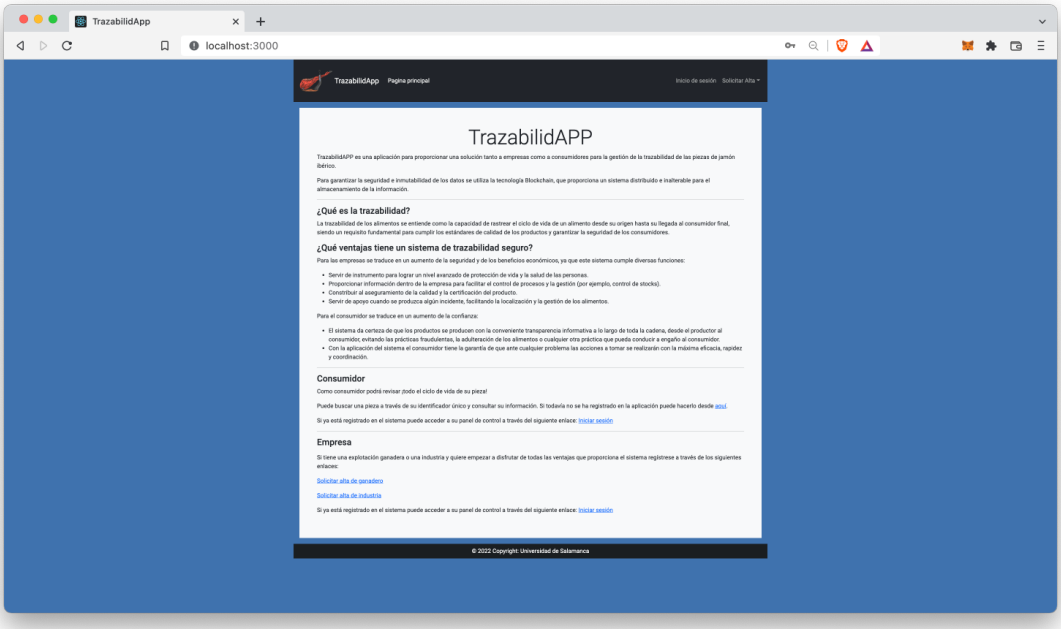

Ilustración 5. Captura de la pantalla principal.

1.3.2 Pantalla de solicitud de alta de usuario

A través de estas vistas se permitirá el alta de los usuarios en el sistema. Seguirá el mismo patrón que la pantalla principal, pero en la zona principal tendrá el formulario para la introducción de los datos del usuario.

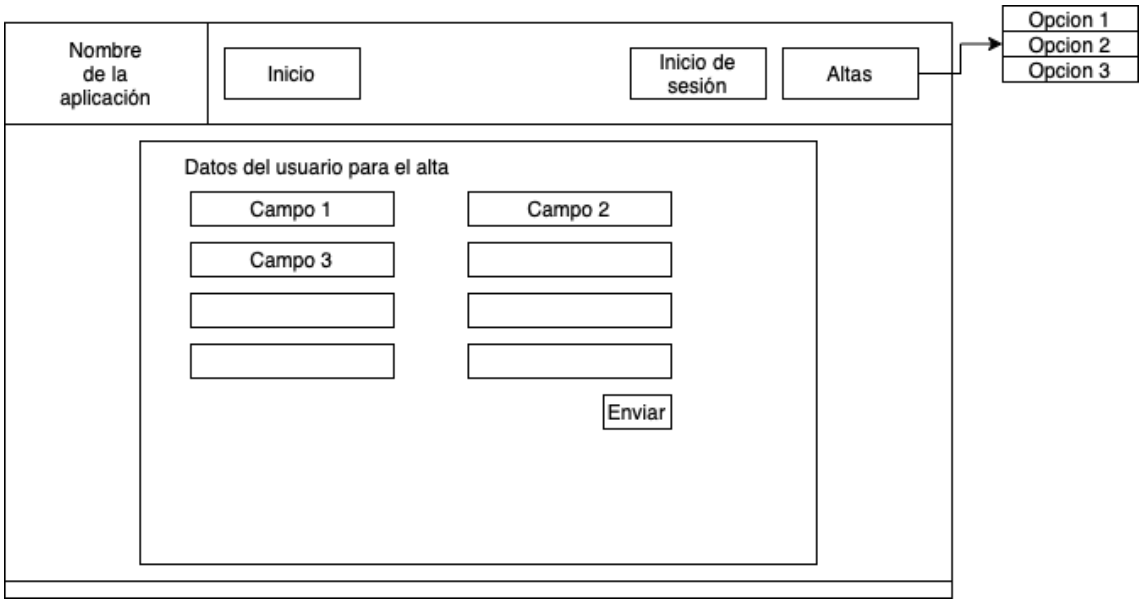

Ilustración 6. Diseño de las páginas de solicitud de alta.

El diseño de las tres pantallas de alta es el mismo, lo único que difieren es en los campos del formulario ya que se necesita información específica para cada tipo de usuario.

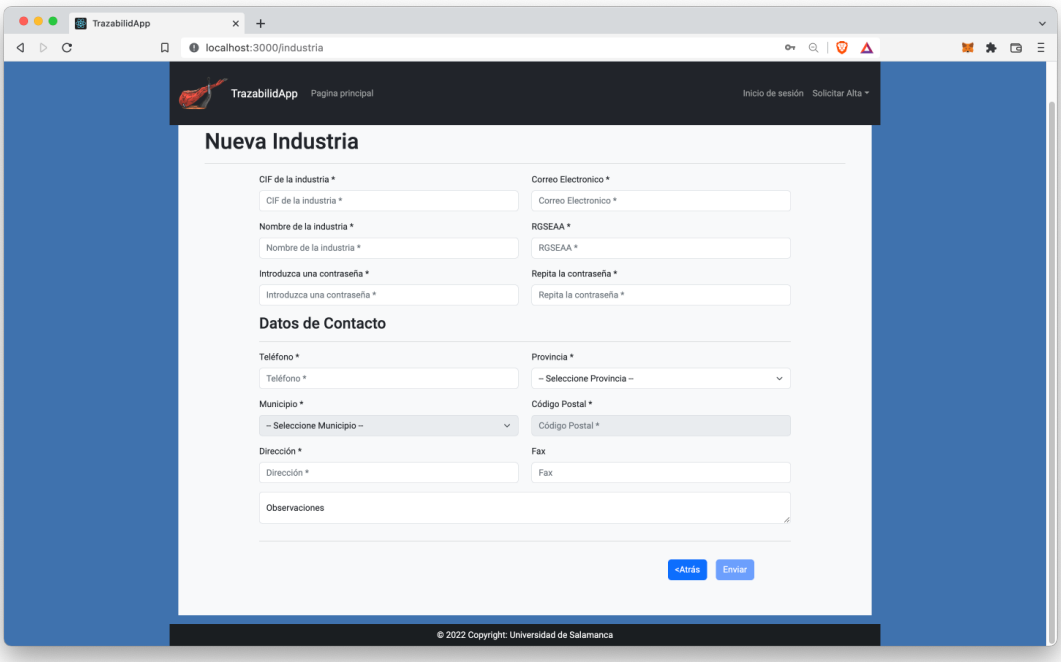

Ilustración 7. Captura de la pantalla de alta de industria.

1.3.3 Pantalla de inicio de sesión

Esta pantalla tendrá la misma estructura global que las anteriores, pero en la parte principal tendrá un formulario para introducir el nombre de usuario y la contraseña.

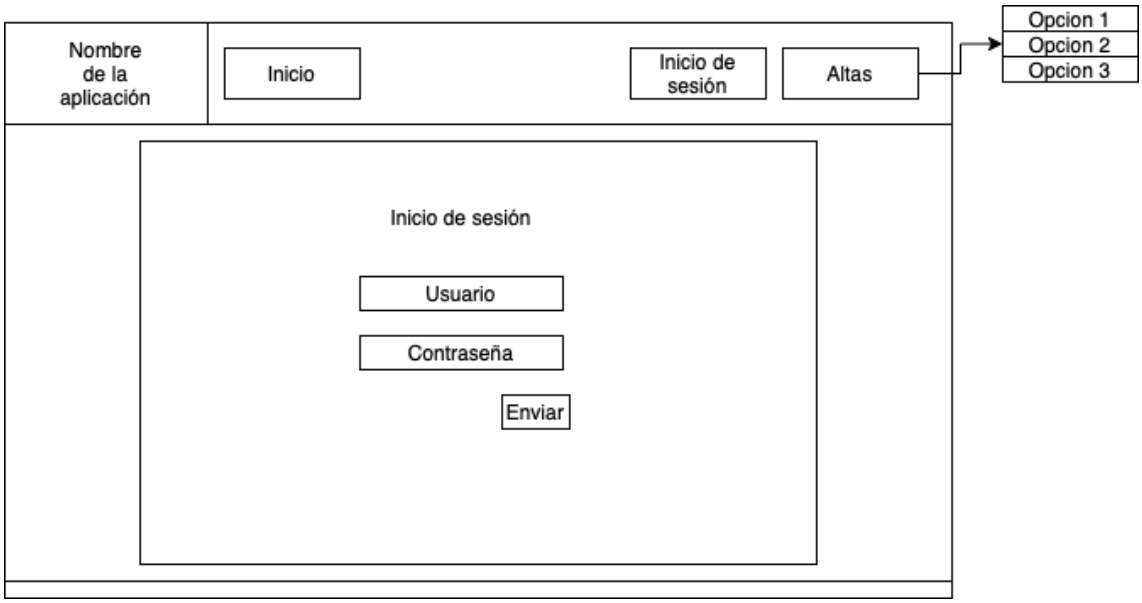

Ilustración 8. Diseño de pantalla de inicio de sesión.

Al haber discrepancias entre la información a introducir entre unos tipos de usuario y otros se ha decidido finalmente añadir unas instrucciones para el inicio de sesión.

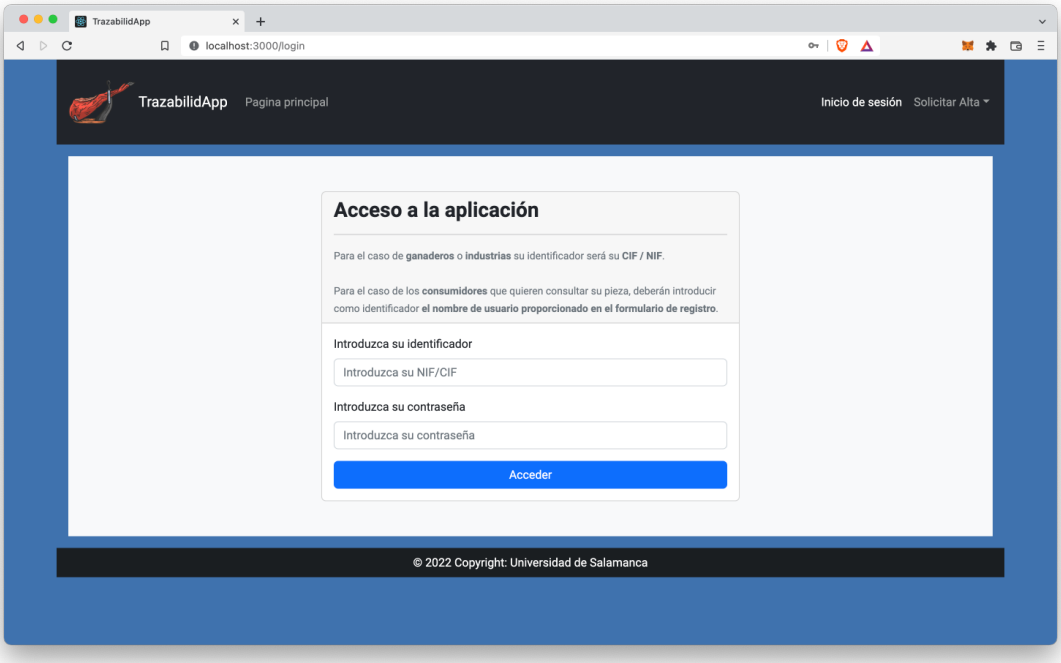

Ilustración 9. Captura de pantalla de inicio de sesión.

1.3.4 Pantalla principal tras iniciar sesión

Esta pantalla seguirá el mismo patrón, pero en la barra inferior aparecerá el nombre del usuario y su rol (consumidor, ganadero, industria, auditor o administrador).

En la barra superior encontraremos como opciones las operaciones que puede realizar el usuario y además opciones de gestión de su perfil como cambiar contraseña, editar información de contacto o cerrar sesión.

En la pantalla principal se encontrará un cuadro de diálogo que permitirá introducir los parámetros para buscar los eventos de las operaciones que ha creado y mostrarlas en la misma pantalla en una tabla.

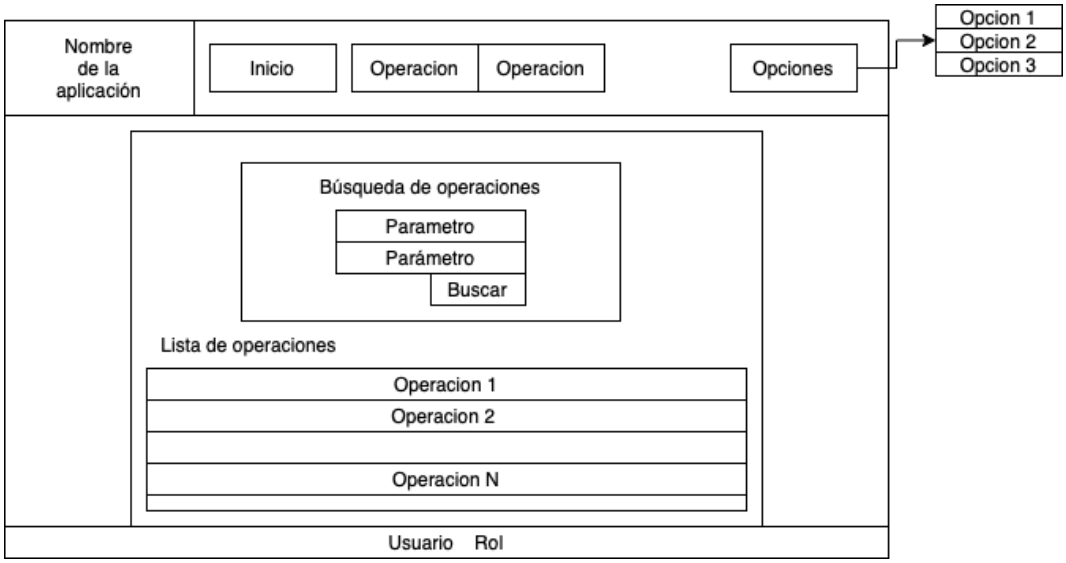

Ilustración 10. Diseño pantalla principal una vez autenticado.

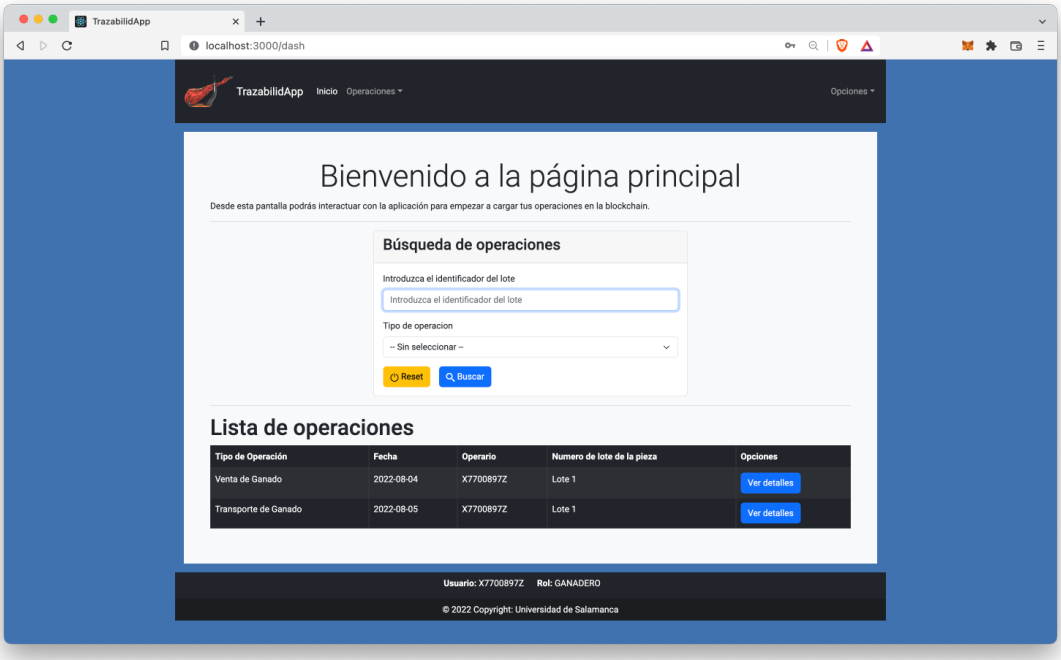

Ilustración 11. Captura de la pantalla principal de un usuario tipo ganadero.

1.3.5 Pantalla de detalle de operaciones

Todos los usuarios excepto los consumidores tendrán la posibilidad de consultar los detalles de las operaciones mediante un botón situado en cada una de las filas que forman la tabla de operaciones.

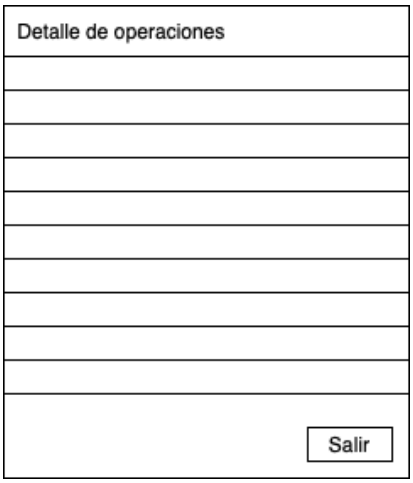

Ilustración 12. Diseño de pantalla de detalle de operaciones.

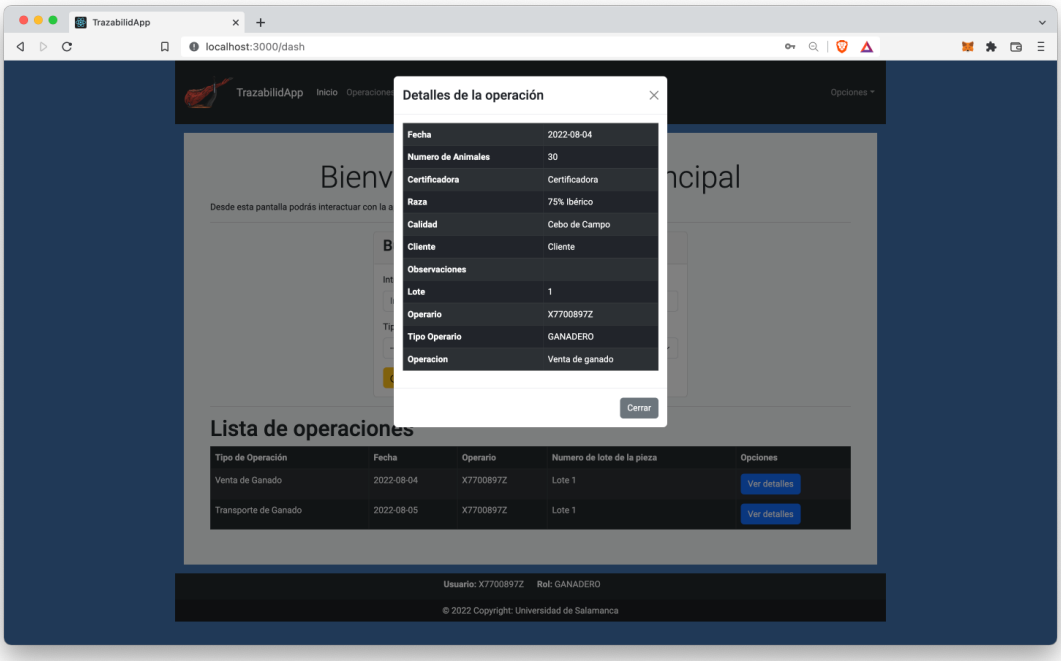

Ilustración 13. Captura de pantalla de detalle de operaciones.

1.3.6 Pantallas de alta de operaciones

Se muestra como ejemplo una pantalla para la creación de operaciones, ya que todas siguen el mismo diseño y lo único en lo que difieren es en la información solicitada en el formulario.

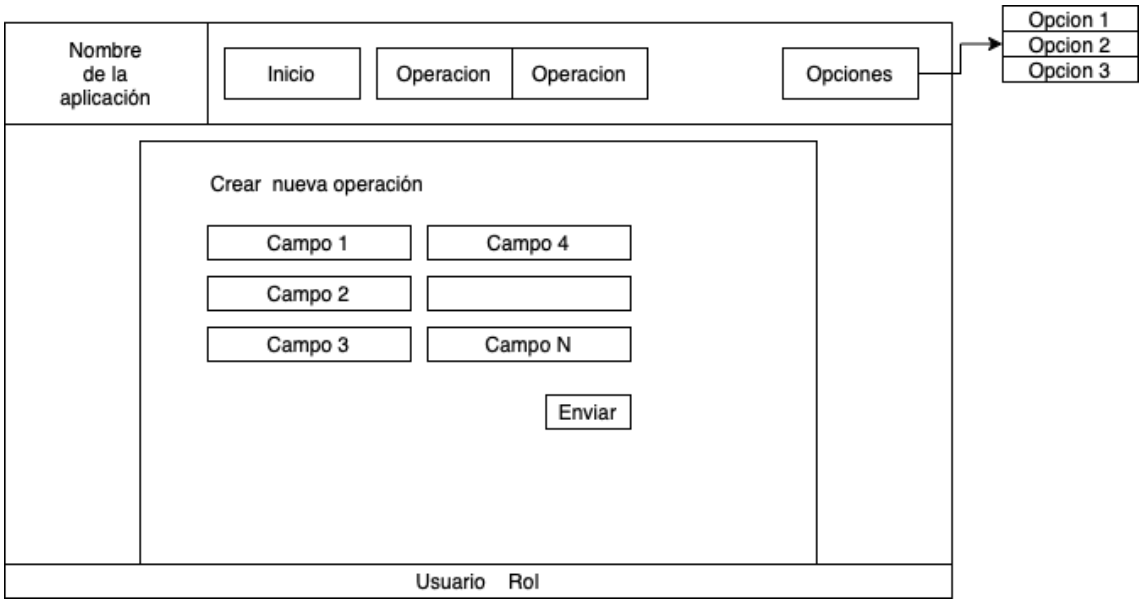

Ilustración 14. Diseño de pantalla de alta de operación.

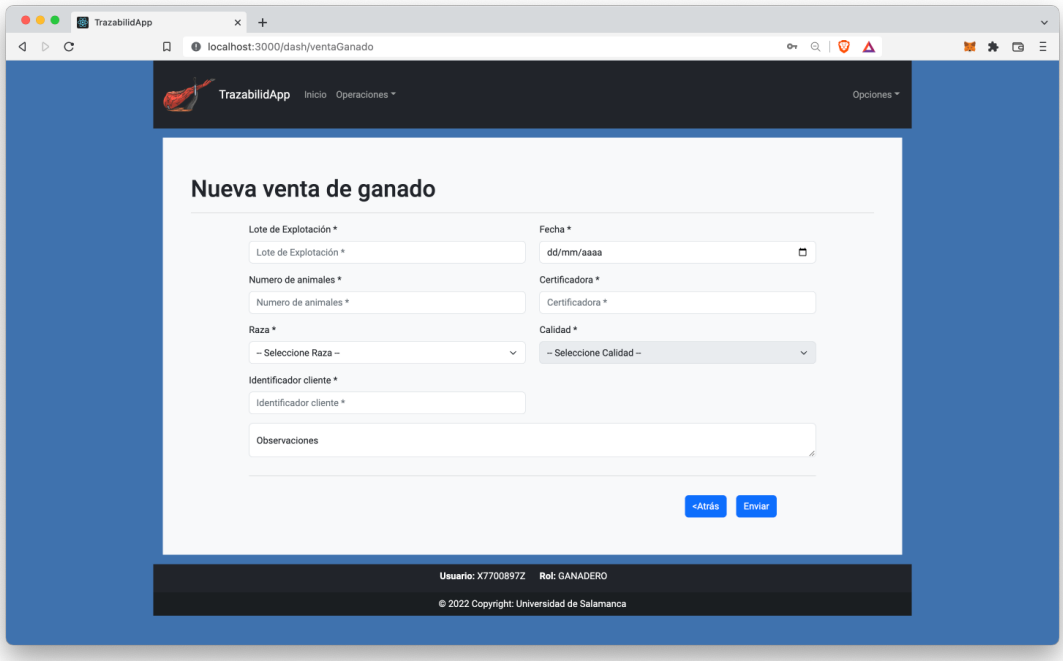

Ilustración 15. Captura de pantalla de alta de operación.

1.3.7 Pantallas de modificación de datos del usuario

Tanto para la modificación de los datos de contacto como para el cambio de contraseña del usuario el diseño será el mismo. Se mostrará únicamente una de las pantallas como ejemplo.

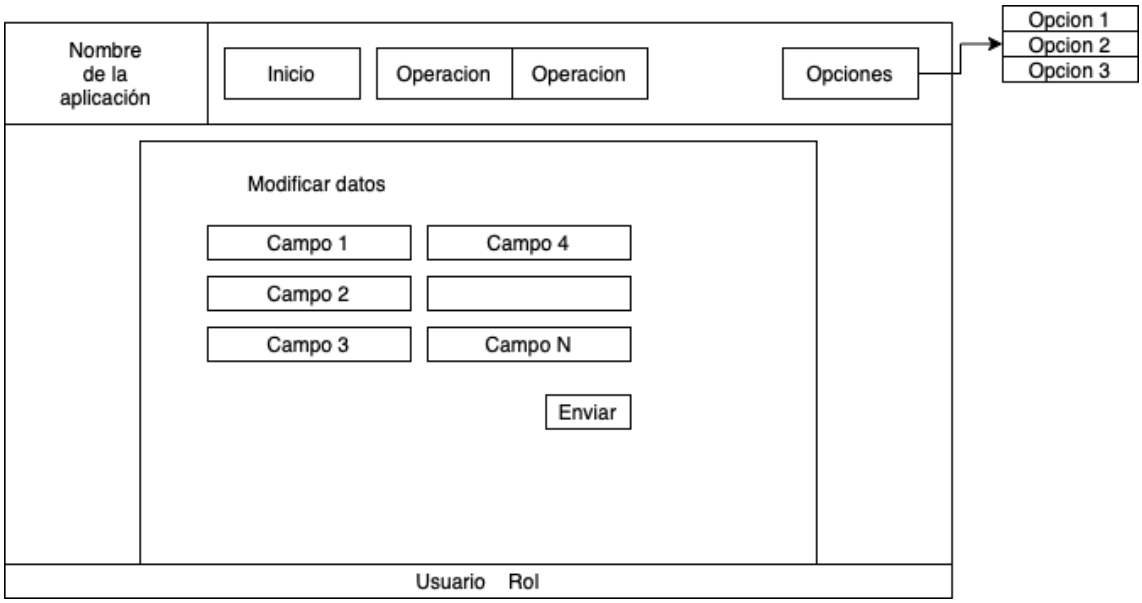

Ilustración 16. Diseño pantalla de modificación de datos.

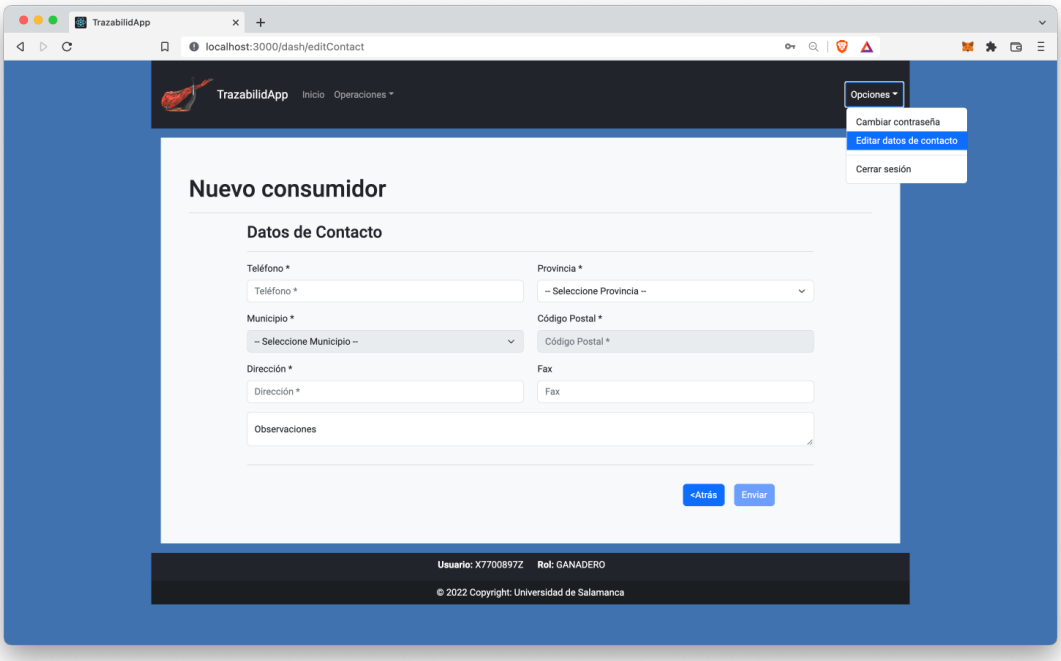

Ilustración 17. Captura de la pantalla de modificación de datos del usuario.

1.3.8 Pantalla de lista de usuarios

Esta pantalla será únicamente accesible para el administrador y le permitirá obtener un listado de todos los usuarios del sistema, junto con un botón para ver la información del usuario y otro botón para eliminarlo del sistema.

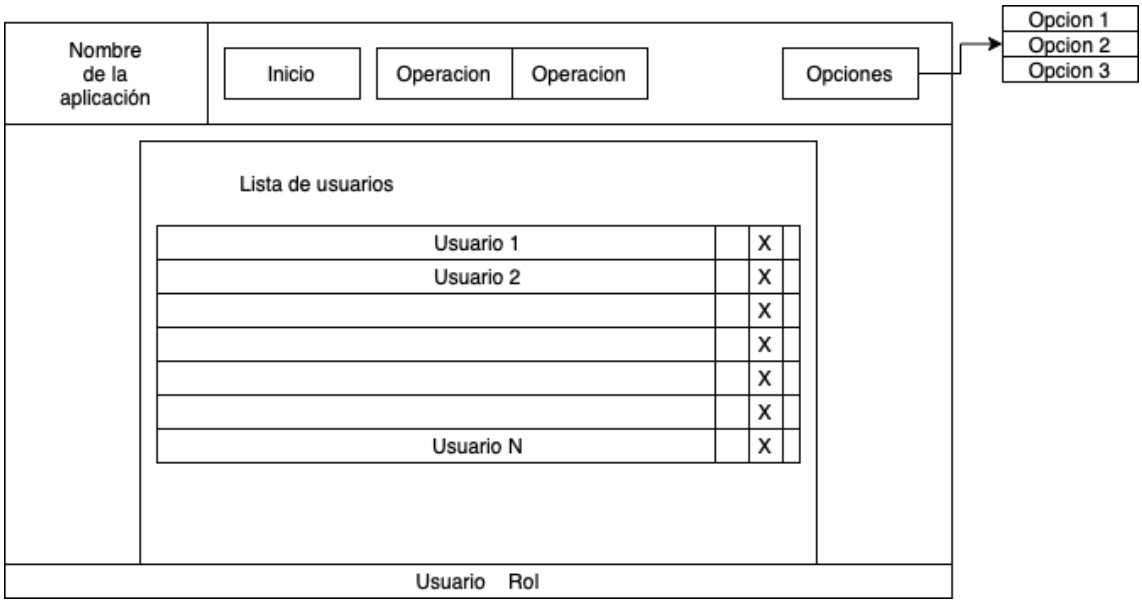

Ilustración 18. Diseño de pantalla de lista de usuarios.

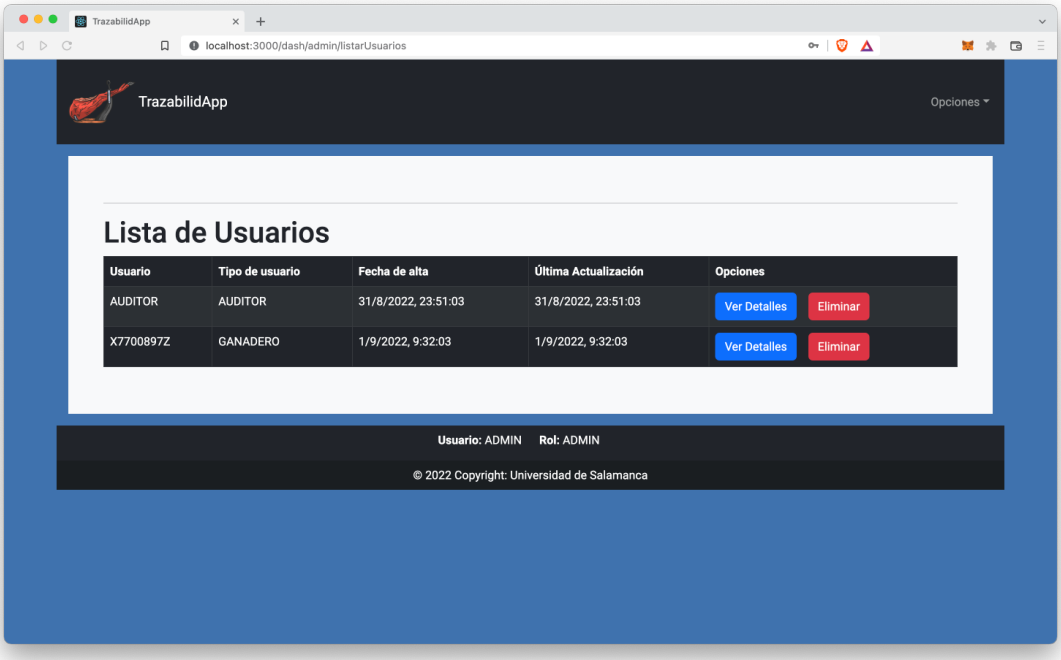

Ilustración 19. Captura de pantalla de lista de usuarios.

1.3.9 Pantalla de detalle de usuario

Esta pantalla es muy similar a la pantalla de detalle de operación.

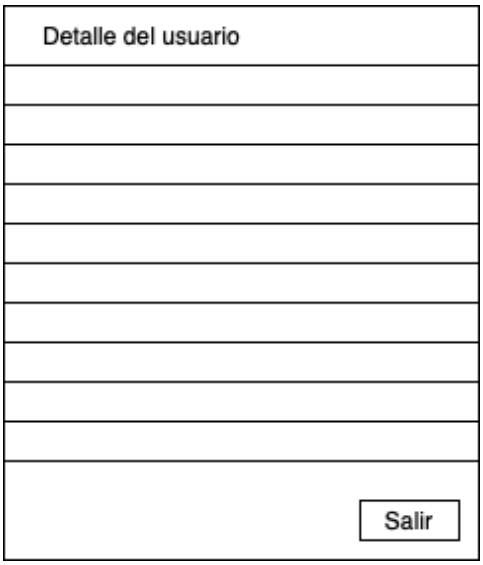

Ilustración 20. Diseño de pantalla de detalle de usuario.

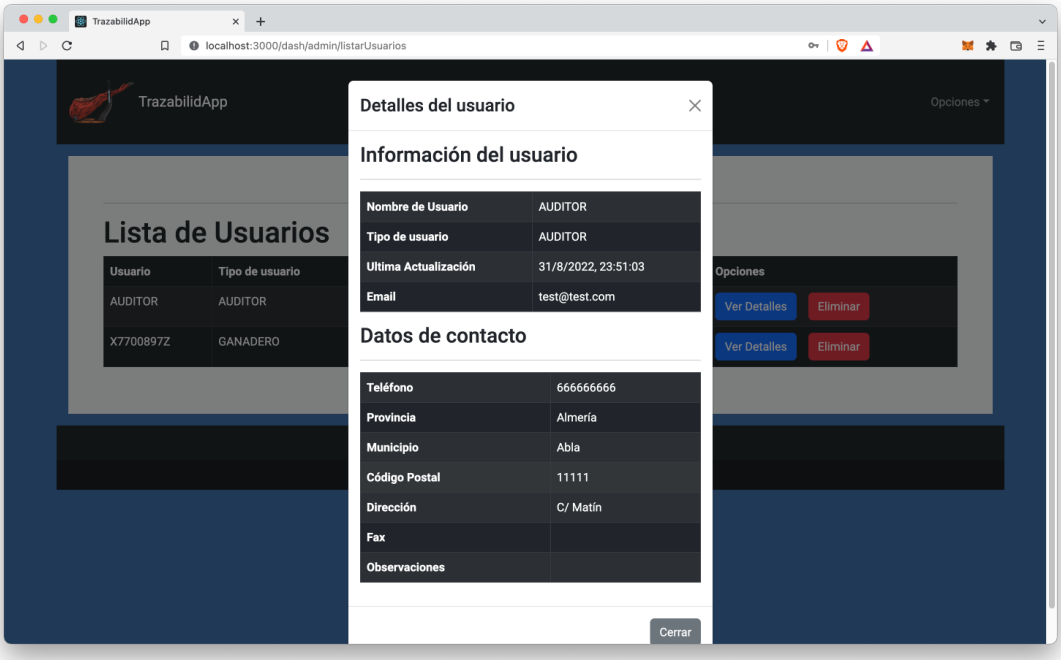

Ilustración 21. Captura de pantalla de detalle de usuario.

1.3.10 Securización del enrutamiento

De la misma manera que se ha aplicado una capa de seguridad en el controlador a través del token de acceso y refresco para que únicamente los usuarios con permiso puedan realizar peticiones a la API desde la interfaz gráfica, es necesario adaptarla para evitar que una persona maliciosa pueda acceder a rutas que no le corresponden. Esta seguridad se aplica teniendo en cuenta las rutas que tienen los usuarios en función de si han iniciado sesión o si su rol les permite acceder al recurso.

En caso de no haber iniciado sesión e intentar acceder a una ruta protegida, el sistema enviará directamente al usuario a la pantalla de inicio de sesión.

En caso de haber iniciado sesión e intentar acceder a un recurso que no existe o para el que no tiene permiso será redirigido a una pantalla de error.

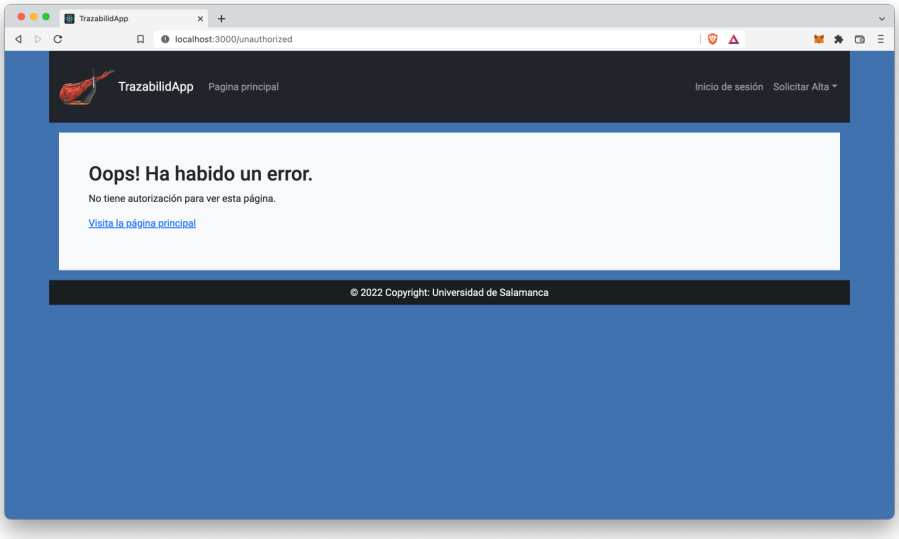

Ilustración 22. Captura de pantalla a una ruta sin autorización.

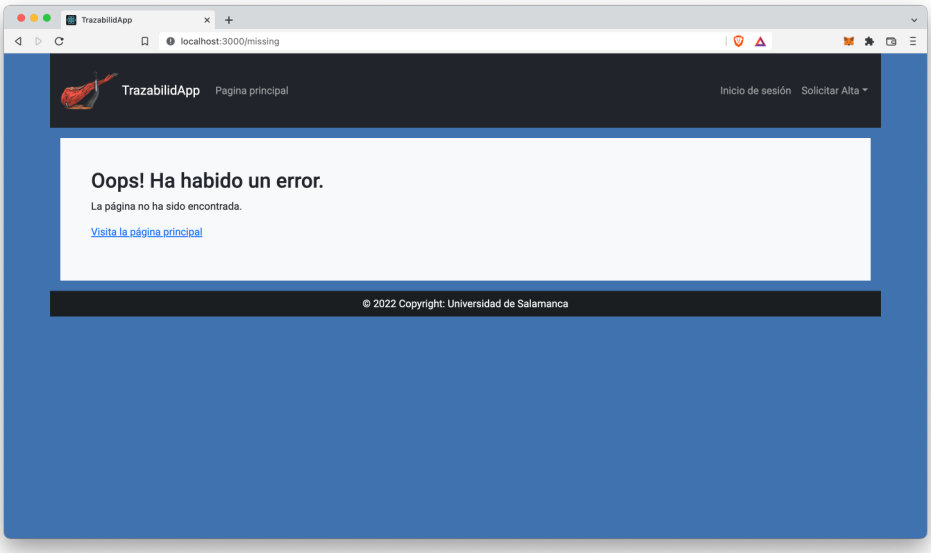

Ilustración 23. Captura de pantalla para ruta inexistente.

1.3.11 Validación de la información

Se han introducido comprobaciones en los formularios para asegurar que el usuario no introduce datos erróneos intencionalmente o por error.

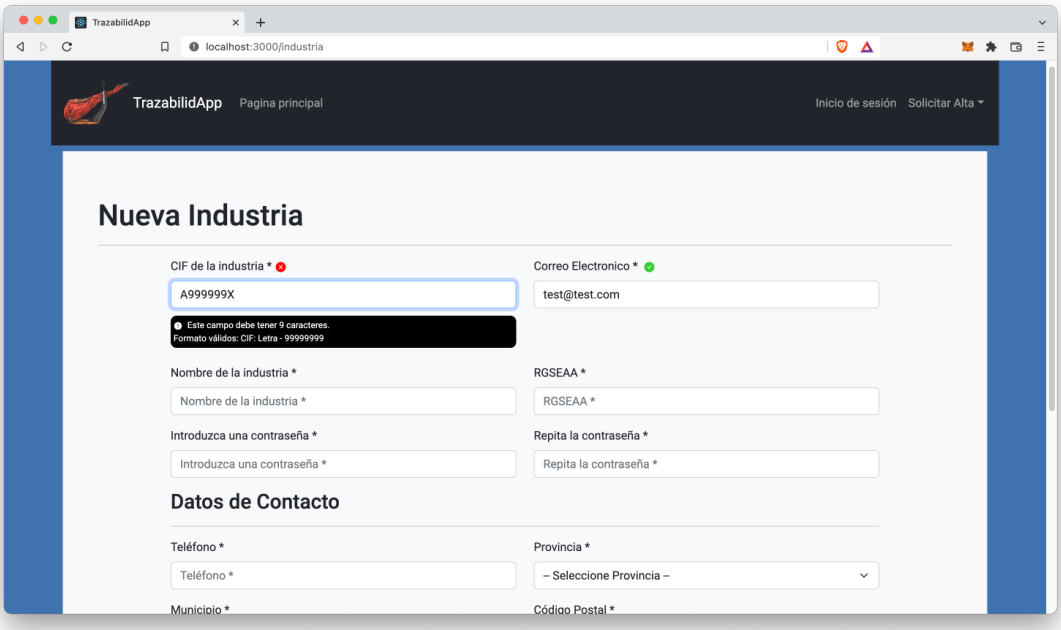

Ilustración 24. Validaciones de los campos de los formularios.

1.3.12 Notificaciones para el usuario

Siempre que el usuario realiza alguna acción recibirá un mensaje flotante indicando si el procesamiento ha sido correcto o erróneo.

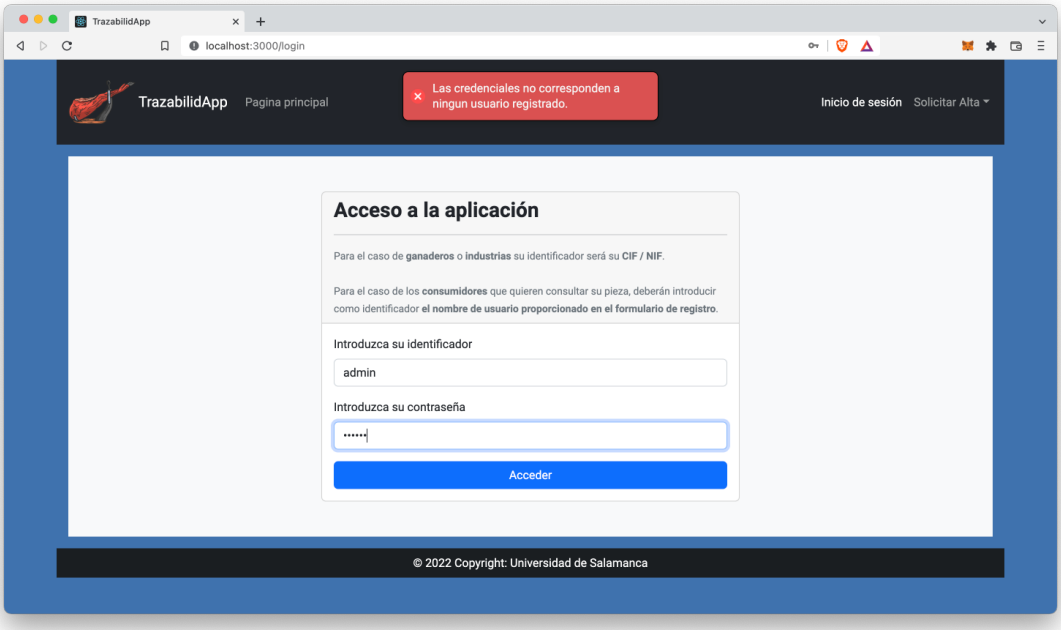

Ilustración 25. Captura de mensaje NOK.

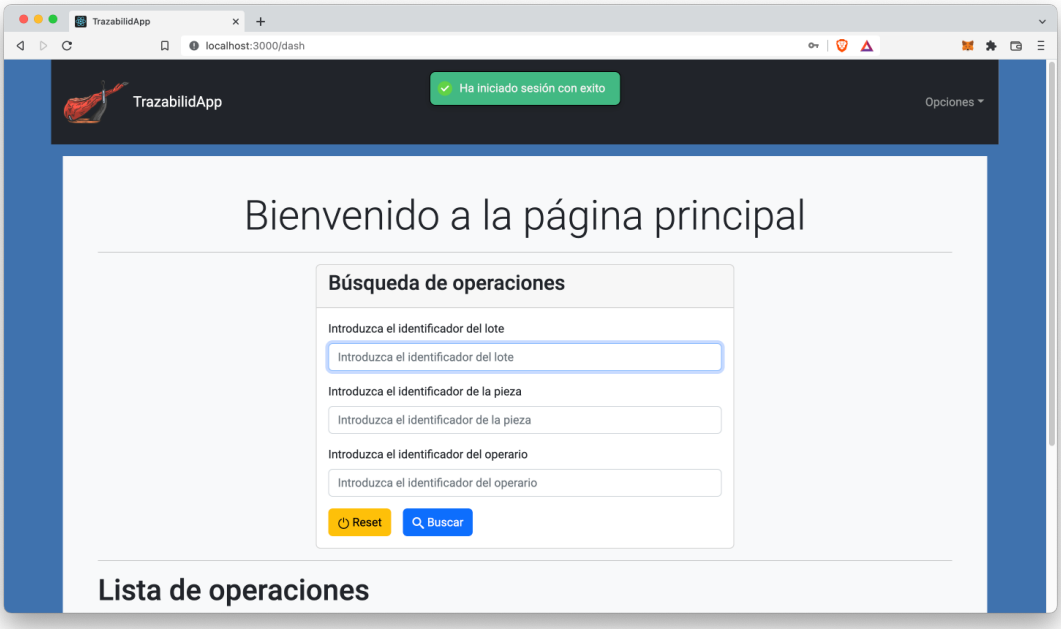

Ilustración 26. Captura de mensaje OK.