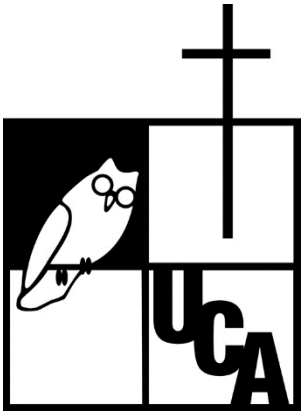
Universidad Centroamericana
José Simeón Cañas



Ingeniería de software
Primera entrega

"Historia, épicas y criterios de aceptación"

Catedrática
Nelson Giovanni Castro Rodas

Estudiantes

Claudia María Chávez Grande	00037221
Rodolfo Rafael Garcia Castillo	00082421
Luis David Guevara Rodríguez	00093217
Carlos Alberto Hernández Guerra	00002721

Fecha de entrega
Lunes 11 de noviembre del 2024

Iniciativa: E-commerce Car Connection

Épica: Módulo de usuario

Historias de usuario:

Número de historia	**1C-Frontend**
Historia: Llenar formulario de cotización	**Como** front-end developer, **Quiero** un diseño en FIGMA que sea responsive de un formulario para la cotización de vehículos que contenga los siguientes campos: Primer nombre del cliente, Primer apellido del cliente, correo electrónico del cliente, número de teléfono del cliente, marca de vehículo y modelo del vehículo **Para** hacer una cotización.
Criterios de aceptación	• El diseño del formulario debe estar en Figma y adaptarse adecuadamente a diferentes tamaños de pantalla.

Número de historia	**2C-Frontend**
Historia: Llenar formulario de cotización	**Como** front-end developer, **Quiero** quiero que el formulario de cotización sea fácil de usar y visualmente atractivo **Para** que el usuario pueda completarlo sin problemas.
Criterios de aceptación	• El formulario tiene campos de texto claros y botones visibles para enviar. • El formulario se adapta a diferentes dispositivos (responsive design).

Número de historia	**3C-Frontend**
Historia: Llenar formulario de cotización	**Como** front-end developer, **Quiero** mostrar mensaje de error específicos en caso de que el usuario olvide llenar algún campo obligatorio o introduzca datos inválidos, **Para** mejorar la experiencia del usuario.
Criterios de aceptación	● Si el usuario deja un campo obligatorio vacío, se muestra un mensaje indicando que debe completarse. ● Los mensajes de error desaparecen cuando el usuario corrige el campo.

Número de historia	**4C-Backend**
Historia: Llenar formulario de cotización	**Como** back-end developer, **Quiero** un endpoint que reciba los campos de la cotización y retorne los códigos de estado correctos. **Para** hacer una cotización.
Criterios de aceptación	● El endpoint debe recibir los siguientes campos obligatorios en el cuerpo de la solicitud. ● El endpoint debe validar que todos los campos obligatorios están presentes y el formato correcto. ● Si todos los campos son válidos, el endpoint debe almacenar los datos en la base de datos. ● Si todos los datos son válidos y el almacenamiento es exitoso, enviar un mensaje de confirmación.

Número de historia	**5C-Backend**
Historia: Llenar formulario de cotización	**Como** back-end developer, **Quiero un modelo** de datos no relacional que contenga los campos que representan la entidad “cotización”. **Para** hacer almacenar los datos del formulario.
Criterios de aceptación	● Los datos se guardan en una base de datos segura tras el envío. ● La información del formulario es accesible para el equipo de ventas a través de la API.

Número de historia	**6C-Cliente**
Historia: Llenar formulario de cotización	**Como** cliente, **Quiero** poder completar y enviar un formulario de cotización **Para** recibir un precio estimado del vehículo de mi interés.
Criterios de aceptación	● Puedo completar todos los campos del formulario necesarios para la cotización. ● Recibo un mensaje de confirmación después de enviar el formulario.

Número de historia	**7C-Cliente**
Historia: Llenar formulario de cotización	**Como** cliente, **Quiero** recibir sugerencias al llenar el formulario **Para** evitar errores.
Criterios de aceptación	● El formulario ofrece mensajes

	que me guían para completar correctamente cada campo.

Número de historia	**8C-Cliente**
Historia: Ver opciones de modelos	**Como** front-end developer, **Quiero** un diseño en FIGMA que sea responsive de una lista que muestre todos los modelos de vehículos disponibles. **Para** poder ver los modelos de vehículos.
Criterios de aceptación	● El diseño del formulario debe estar en Figma y adaptarse adecuadamente a diferentes tamaños de pantalla.

Número de historia	**9C-Cliente**
Historia: Ver opciones de modelos	**Como** backend developer **Quiero** un endpoint que devuelva las opciones de modelos disponibles para un vehículo seleccionado, **Para** que los datos estén listos para ser mostrados al usuario.
Criterios de aceptación	● El endpoint recibe como parámetro el id o el nombre de vehículo y devuelve la lista de modelos disponibles. ● Si el vehículo seleccionado no tiene modelos disponibles, devuelve un mensaje de “Modelos no encontrados” y un código de estado 404 Not Found.

Número de historia	**10C-Cliente**
Historia: Ver opciones de modelos	**Como** frontend developer **Quiero** mostrar las opciones de modelos de forma clara y organizada, **Para** que el usuario pueda verlas y seleccionar el que le interese.
Criterios de aceptación	● Las opciones de modelos se presentan en una lista con el nombre, año y características relevantes de cada modelo. ● Cada modelo tiene un botón o enlace para obtener más detalle o para iniciar una solicitud de cotización. ● Si no se encuentran modelos para el vehículo seleccionado, se muestra un mensaje claro de que no hay modelos disponibles.

Número de historia	**11C-Cliente**
Historia: Ver opciones de modelos	**Como** cliente, **Quiero** poder ver las diferentes opciones de modelos para el vehículo que seleccioné, **Para** elegir el que mejor se adapte a mis necesidades.
Criterios de aceptación	● Puedo ver la lista de modelos disponibles para el vehículo seleccionado con información detallada de cada uno. ● Si no hay modelos disponibles, recibo un mensaje informando que no hay modelos disponibles y sugerirme otros modelos relacionados.

Épica: Módulo de administrador

Historias de usuario:

Número de historia	**1A-Frontend**
Historia: Lista de empleados de una sucursal	**Como** front-end developer, **Quiero** una lista con todos los empleados asignados a una sucursal, donde se detalle el nombre, email, y codigo de empleado, además, que se pueda agregar empleados para cada sucursal. **Para** gestionarlos, es decir, actualizar su información o desvincularlo de la sucursal respectiva.
Criterios de aceptación	● Al acceder a la sucursal, el administrador puede ver una lista de todos los empleados asignados, mostrando el nombre, email y código de empleado, además puede agregar nuevos empleados. ● En la lista de empleado, el administrador tiene un botones para actualizar información del empleado o desvincularlo de la sucursal.

Número de historia	**2A-Frontend**
Historia: Dashboard de funcionalidades principales	**Como** front-end developer, **Quiero** un dashboard donde pueda acceder rápidamente a las funcionalidades principales del sistema, como Sucursales, Empleados, Vehículos y Cotizaciones

	Para gestionar de manera eficiente las operaciones diarias.
Criterios de aceptación	● El dashboard debe tener un diseño limpio y ordenado, con íconos o botones claramente identificados para cada funcionalidad (Sucursales, Empleados, Vehículos, Cotizaciones). ● El diseño debe ser completamente funcional tanto en dispositivos de escritorio como móviles, adaptándose a diferentes tamaños de pantalla.

Número de historia	**3A-Frontend**
Historia: Formulario para agregar un vehiculo al inventario	**Como** front-end developer, **Quiero** un formulario para agregar vehículos al inventario y vincularlos a una sucursal específica, el formulario debe llevar los siguientes campos: Empresa fabricante del vehículo, modelo del vehículo, año del vehículo, descripción, imágenes, tipo de combustible, motor, tipo de vehículo, unidades en stock, transmisión, kilometraje, estado, código VIN, Tren de tracción, número de puertas, color, capacidad de asientos y sucursal **Para** gestionar eficientemente los vehículos disponibles por cada ubicación.
Criterios de aceptación	● Mostrar un formulario con los campos de Empresa fabricante del vehículo, modelo del vehículo, año del vehículo,

	descripción, imágenes, tipo de combustible, motor, tipo de vehículo, unidades en stock, transmisión, kilometraje, estado, código VIN, Tren de tracción, número de puertas, color, capacidad de asientos y sucursal. ● Los campos obligatorios deben ser validados antes de enviar el formulario.

Número de historia	**4A-Frontend**
Historia: Lista de los vehículos en existencia	**Como** front-end developer, **Quiero** una lista que me permita visualizar todos los vehículos en existencia y tener la opción de modificar o eliminar su información, **Para** gestionar el inventario de vehículos de manera eficiente.
Criterios de aceptación	● La lista debe mostrar la información básica de cada vehículo. ● Cada vehículo en la lista debe tener un botón o enlace para editar o eliminar su información. Al hacer clic en "Modifica o eliminar", el sistema debe llevar al administrador a un formulario prellenado con la información del vehículo, donde podrá realizar cambios y guardarlos o eliminar el vehículo. ● Mostrar un mensaje de confirmación si ha sido modificado o eliminado con éxito.

Número de historia	**5A-Backend**
Historia: Registro de vehículos en el sistema	**Como** back-end developer, **Quiero** implementar un endpoint para el registro de vehículos en el sistema, **Para** que los administradores puedan agregar nuevos vehículos al inventario de manera eficiente.
Criterios de aceptación	• El endpoint debe utilizar el método `POST`, ya que se está creando un nuevo recurso (vehículo) en el sistema.

Número de historia	**6A-Backend**
Historia: Registro de empleados en el sistema	**Como** back-end developer, **Quiero** implementar un endpoint para el registro de empleados en el sistema, **Para** que los administradores puedan agregar nuevos empleados de manera eficiente y con todos los datos necesarios.
Criterios de aceptación	• El endpoint debe utilizar el método `POST`, ya que se está creando un nuevo recurso (empleado) en el sistema.

Número de historia	**7A-Backend**
Historia: Obtención de vehículos del inventario	**Como** back-end developer, **Quiero** implementar un endpoint para la obtención de todos los vehículos del inventario,

	Para que los administradores puedan consultar toda la información de los vehículos registrados en el sistema.
Criterios de aceptación	• El endpoint debe utilizar el método `GET`, ya que se está realizando una operación de lectura.

Número de historia	**8A-Backend**
Historia: Obtener vehículos por sucursal	**Como** back-end developer, **Quiero** implementar un endpoint para obtener todos los vehículos vinculados a una sucursal específica, **Para** para que los administradores puedan consultar rápidamente los vehículos de una sucursal en particular.
Criterios de aceptación	• El endpoint debe utilizar el método `GET`, ya que es una operación de lectura.

Número de historia	**9A-Backend**
Historia: Lista de empleados por sucursal	**Como** back-end developer, **Quiero** implementar un endpoint para obtener todos los empleados vinculados a una sucursal específica **Para** que los administradores puedan consultar rápidamente los empleados que trabajan en una sucursal determinada.
Criterios de aceptación	• El endpoint debe utilizar el método `GET`, ya que es una operación de lectura.

Número de historia	**10A-Backend**
Historia: Obtener empleados	**Como** back-end developer, **Quiero** implementar un endpoint para obtener todos los empleados **Para** que los administradores puedan consultar la lista completa de empleados de la empresa.
Criterios de aceptación	• El endpoint debe utilizar el método `GET`, ya que es una operación de lectura.

Número de historia	**11A-Backend**
Historia: Vincular a los empleados a una sucursal	**Como** back-end developer, **Quiero** implementar un endpoint para vincular un empleado a una sucursal específica, **Para** que los administradores puedan gestionar de manera eficiente a qué sucursal pertenece cada empleado.
Criterios de aceptación	• El endpoint debe utilizar el método `POST`, ya que es una operación de vinculación.

Número de historia	**12A-Backend**
Historia: Lista de los vehículos en existencia por cada sucursal	**Como** back-end developer, **Quiero** implementar un endpoint para vincular vehículos en existencia a una sucursal específica, **Para** que los administradores puedan gestionar y asignar vehículos de manera eficiente dentro de las distintas sucursales.

Criterios de aceptación	• El endpoint debe utilizar el método `POST`, ya que se trata de una operación de vinculación.

Número de historia	**14A-Backend**
Historia: CRUD para las sucursales	**Como** back-end developer, **Quiero** implementar el conjunto completo de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) **Para** la gestión de sucursales, de manera que los administradores puedan gestionar las sucursales de forma eficiente.
Criterios de aceptación	• Cuando un administrador desee gestionar las sucursales, deberá enviar una solicitud `POST, GET y PUT` (Para leer, actualizar, crear)

Número de historia	**15A-Backend**
Historia: CRUD para las cotizaciones	**Como** back-end developer, **Quiero** quiero implementar el conjunto completo de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) **Para** la gestión de cotizaciones, de manera que los administradores puedan gestionar las cotizaciones de los clientes de forma eficiente.
Criterios de aceptación	• Cuando un administrador desee crear una nueva cotización, deberá enviar una solicitud `POST, GET y PUT` (Para leer, actualizar, crear)

Épica: Módulo de empleado

Historias de usuario:

Número de historia	**1E-Backend**
Historia: Revisar cotizaciones.	**Como** backend developer, **Quiero** almacenar y gestionar las cotizaciones enviadas por los usuarios, **Para** que los empleados puedan acceder y revisarlas cuando sea necesario.
Criterios de aceptación	• Las cotizaciones deben almacenarse de manera segura y accesible. • Se debe crear una API que permita la recuperación de cotizaciones según criterios específicos (fecha, nombre del cliente, etc).

Número de historia	**2E-Frontend**
Historia: Revisar cotizaciones.	**Como** frontend developer, **Quiero** mostrar las cotizaciones enviadas por los clientes, **Para** que el empleado pueda revisarlas y verificar los detalles.
Criterios de aceptación	• El empleado puede ver una lista de cotizaciones y hacer clic en cada una para ver los detalles.

Número de historia	**3E-Empleado**
Historia: Revisar cotizaciones.	**Como** empleado, **Quiero** acceder a las cotizaciones enviadas por los usuarios,

	Para revisarlas y verificar la información proporcionada.
Criterios de aceptación	• El empleado debe poder ver la lista de cotizaciones y acceder a los detalles completos de cada una. • El sistema debe proporcionar acceso seguro y restringido.

Número de historia	**4E-Backend**
Historia: Ver tabla de cuotas	**Como** backend developer, **Quiero** gestionar y proveer los datos de la tabla de cuotas para cada vehículo, **Para** que cada empleado pueda consultarla.
Criterios de aceptación	• La tabla de cuotas debe ser calculada según los datos del vehículo (año, modelo, precio). • La información debe estar actualizada y accesible mediante una API.

Número de historia	**5E-Frontend**
Historia: Ver tabla de cuotas	**Como** frontend developer, **Quiero** mostrar una tabla de cuotas que tenga los siguientes campos: porcentaje de interés, cantidad de años para el pago de la cuota, id del modelo que está cotizando, prima y método de pago, de manera detallada para un vehículo específico, **Para** que el empleado pueda consultar la información de pagos.
Criterios de aceptación	• La interfaz debe mostrar la tabla

Número de historia	**5E-Frontend**
Historia: Ver tabla de cuotas	**Como** frontend developer, **Quiero** mostrar una tabla de cuotas que tenga los siguientes campos: porcentaje de interés, cantidad de años para el pago de la cuota, id del modelo que está cotizando, prima y método de pago, de manera detallada para un vehículo específico, **Para** que el empleado pueda consultar la información de pagos.
	de cuotas de manera clara y accesible. ● Debe permitir al empleado seleccionar el vehículo y ver la información de cuotas correspondientes.

Número de historia	**6E-Empleado**
Historia: Ver tabla de cuotas	**Como** empleado, **Quiero** consultar la tabla de cuotas de un vehículo específico, **Para** informar al cliente sobre los pagos.
Criterios de aceptación	● El empleado debe poder acceder fácilmente a la tabla de cuotas. ● La tabla debe presentar el desglose de cuotas y términos de pago.

Número de historia	**6E-Backend**
Historia: Ver información del cliente	**Como** backend developer,

	Quiero almacenar y proveer de forma segura la información personal del cliente, **Para** que el empleado pueda consultarla cuando sea necesario.
Criterios de aceptación	● La información debe ser accesible mediante una API. ● Los datos deben estar vinculados a sus respectivas cotizaciones.

Número de historia	**7E-Frontend**
Historia: Ver información del cliente	**Como** frontend developer, **Quiero** mostrar la información personal del cliente relacionada con una cotización, **Para** que el empleado pueda revisar y verificar los detalles.
Criterios de aceptación	● La interfaz debe mostrar la información de manera clara y ordenada. ● Debe permitir una fácil navegación entre los datos de los clientes y los detalles de la cotización.

Número de historia	**8E-Empleado**
Historia: Ver información del cliente	**Como** empleado, **Quiero** visualizar la información personal del cliente asociado con una cotización, **Para** verificar que los datos son correctos y están actualizados.
Criterios de aceptación	● El empleado puede acceder a la

	información del cliente vinculada a una cotización.

Número de historia	**9E-Backend**
Historia: Ver sucursales disponibles	**Como** backend developer, **Quiero** gestionar y proveer información actualizada sobre la disponibilidad de vehículos en cada sucursal, **Para** que el administrador pueda consultar estos datos.
Criterios de aceptación	● La disponibilidad debe actualizarse en tiempo real y estar accesible mediante una API. ● Los datos deben incluir detalles sobre el tipo de vehículos y su cantidad en cada sucursal.

Número de historia	**10E-Frontend**
Historia: Ver sucursales disponibles	**Como** frontend developer, **Quiero** mostrar la disponibilidad de vehículos en cada sucursal, **Para** que el administrador pueda visualizar y consultar las opciones disponibles.
Criterios de aceptación	● La interfaz debe permitir al administrador seleccionar una sucursal y ver los vehículos disponibles en ella. ● Debe incluir un desglose claro de los vehículos y sus cantidades por sucursal.

Número de historia	**11E-Administrador**
Historia: Ver sucursales disponibles	**Como** administrador, **Quiero** consultar la disponibilidad de vehículos en cada sucursal, **Para** dar opciones precisas de ubicación y disponibilidad a los clientes.
Criterios de aceptación	● El administrador puede seleccionar y consultar cada sucursal. ● La interfaz debe ser fácil de usar y brindar datos actualizados.

Número de historia	**12E-Backend**
Historia: Ver contrato	**Como** backend developer, **Quiero** almacenar y permitir el acceso a los contratos generados para los clientes, **Para** que el empleado pueda consultarlos y editarlos si es necesario.
Criterios de aceptación	● Los contratos deben almacenarse de manera segura y ser accesibles mediante la API. ● Debe haber un registro de cambios o actualizaciones en cada contrato.

Número de historia	**13E-Empleado**
Historia: Ver contrato	**Como** empleado**,** **Quiero** revisar y generar el contrato para un cliente y que tenga los siguientes campos: Nombre completo del cliente, DUI e información predeterminada.

	Para asegurarme de que toda la información esté correcta antes de formalizar el contrato.
Criterios de aceptación	• El empleado puede acceder al contrato, revisarlo y realizar cambios si es necesario. • El contrato debe poder generarse en un formato formal.

Épica: Pase a producción

Número de historia	**1PP-Backend**
Historia: Prepara el script de despliegue	**Como** back-end developer **Quiero** crear un script que configure automáticamente el entorno de producción (instalación de dependencias, configuración de la base de datos y el servidor), **Para** que el proceso de despliegue se a automatizado y rápido.
Criterios de aceptación	• El script debe instalar Node.js y las dependencias necesarias para el backend. • El script debe configurar las variables de entorno necesarias (puerto, base de datos, claves de API). • El script debe realizar migraciones y configuraciones de base de datos si es necesario.

Número de historia	**2PP-Backend**
Historia: Prepara el script de despliegue	**Como** back-end developer **Quiero** crear un script que construya y prepare automáticamente la versión

	optimizada de la aplicación React para producción **Para** asegurar que el código esté listo para el despliegue en el servidor.
Criterios de aceptación	• El script debe ejecutar `npm run build` o `yarn build` para optimizar los archivos para producción. • El script debe copiar los archivos generados en el directorio adecuado para que sean servidos por el servidor web (Nginx). • El script debe generar un archivo `.env` para que la configuración de la API del frontend sea correcta en producción.

Número de historia	**3PP-Backend**
Historia: Prepara el script de despliegue	**Como** administrador **Quiero** tener un proceso de despliegue automatizado que se encargue de instalar, configurar y poner en marcha tanto el backend como el frontend de manera sencilla y rápida, **Para** minimizar los errores y garantizar un despliegue eficiente.
Criterios de aceptación	• El administrador puede ejecutar un solo script para completar todo el proceso de despliegue. • El script debe configurar el entorno y la base de datos automáticamente. • El administrador debe recibir un informe claro con el estado del despliegue y posibles errores.

Número de historia	**4PP-Backend**
Historia: Despliegue de la base de datos en MongoDB Atlas	**Como** backend developer, **Quiero** desplegar la base de datos de mi aplicación en MongoDB Atlas, **Para** asegurarme de que la base de datos esté disponible en la nube, sea escalable y esté protegida.
Criterios de aceptación	• El desarrollador debe crear un clúster en MongoDB Atlas. • Debe configurar las credenciales de acceso (usuario y contraseña) para conectar la aplicación a MongoDB Atlas. • La base de datos debe ser configurada correctamente en MongoDB Atlas (incluyendo las variables de entorno necesarias, como `MONGODB_URI`). • Debe realizarse la migración de datos desde la base de datos local (si es necesario) a MongoDB Atlas. • El backend debe estar configurado para conectarse a MongoDB Atlas sin problemas de rendimiento o seguridad.

Número de historia	**5PP-Administrador**
Historia: Despliegue de la base de datos en MongoDB Atlas	**Como** administrador, **Quiero** verificar que la base de datos en MongoDB Atlas esté correctamente desplegada y funcionando, **Para** asegurarme de que no haya errores de conexión y que el sistema esté listo para producción.

Criterios de aceptación	• El administrador debe validar que los datos estén sincronizados y accesibles desde la aplicación.

Número de historia	**6PP-Frontend**
Historia: Obtener datos desde el servidor	**Como** frontend developer, **Quiero** conectar la vista de la aplicación con el servidor usando Axios y Redux Hooks, **Para** obtener los datos de cotizaciones y mostrarlos de manera eficiente en la interfaz de usuario.
Criterios de aceptación	• El desarrollador debe configurar el dispatch de la acción asincrónica `fetchCotizaciones` para obtener las cotizaciones desde la API usando Axios. • La vista debe mostrar un estado de carga mientras se esperan los datos, y debe mostrar las cotizaciones una vez que la solicitud se complete correctamente. • Se debe manejar un estado de error en caso de que la solicitud falle. • Los datos obtenidos deben almacenarse en el estado global de Redux, para que otros componentes puedan acceder a ellos. • El componente debe actualizarse automáticamente cuando los datos cambien.

Número de historia	**7PP-Frontend**
Historia: Actualizar los datos del servidor	**Como** frontend developer, **Quiero** quiero enviar datos al servidor para actualizar las cotizaciones (o cualquier otro recurso), usando Axios y Redux Hooks, **Para** que la interfaz de usuario refleje los cambios en tiempo real.
Criterios de aceptación	• El desarrollador debe configurar una acción asincrónica `updateCotizacion` para enviar datos modificados al servidor utilizando Axios. • El componente frontend debe tener un formulario o interfaz que permita al usuario modificar las cotizaciones. • El componente debe despachar la acción `updateCotizacion` al hacer el envío de los datos. • El estado global de Redux debe actualizarse para reflejar los cambios, asegurando que la vista se sincronice con el servidor. • Si la solicitud es exitosa, la cotización modificada debe reflejarse en la interfaz; si ocurre un error, debe mostrarse un mensaje de error adecuado.

Número de historia	**8PP-Frontend**
Historia: Testear la conexión de las rutas al servidor	**Como** frontend developer, **Quiero** testear las rutas de conexión al servidor

	Para asegurarme de que las solicitudes y respuestas funcionan correctamente.
Criterios de aceptación	● Las solicitudes HTTP deben llegar al servidor y devolver los datos esperados. ● Los errores deben ser manejados adecuadamente.

Número de historia	**9PP-Frontend**
Historia: Testear la conexión de las rutas al servidor	**Como** frontend developer, **Quiero** testear las rutas de navegación **Para** asegurarme de que las vistas se carguen correctamente al hacer clic en los enlaces.
Criterios de aceptación	● Las rutas deben redirigir a las vistas correctas al navegar. ● Las rutas protegidas deben funcionar según el acceso del usuario.

Número de historia	**10PP-Developer**
Historia: Actualizar el conocimiento sobre las últimas modificaciones para el despliegue.	**Como** developer, **Quiero** aprender sobre las últimas modificaciones necesarias para el despliegue en DigitalOcean, **Para** asegurarme de que el entorno de producción esté correctamente configurado.
Criterios de aceptación	● El desarrollador debe obtener información sobre las actualizaciones recientes en las configuraciones de DigitalOcean. ● Debe revisar la documentación oficial y cualquier cambio en la

	infraestructura.

Número de historia	**11PP-Developer**
Historia: Actualizar el conocimiento sobre las últimas modificaciones para el despliegue.	**Como** developer, **Quiero** aplicar las últimas modificaciones al entorno de DigitalOcean **Para** garantizar que el sistema esté desplegado correctamente con las configuraciones más recientes.
Criterios de aceptación	• Las modificaciones deben implementarse siguiendo las mejores prácticas de despliegue. • El entorno de DigitalOcean debe estar completamente actualizado con las nuevas configuraciones.

Número de historia	**12PP-Developer**
Historia: Preparar el backend para el despliegue.	**Como** developer, **Quiero** preparar el código del backend para el despliegue, asegurándome de que todas las dependencias y configuraciones estén listas **Para** producción.
Criterios de aceptación	• El código debe estar completamente probado y libre de errores. • Todas las dependencias deben estar actualizadas y listadas en el archivo de configuración (por ejemplo, `package.json`). • La configuración del entorno de

	producción debe ser correcta (variables de entorno, claves API, bases de datos, etc.). ● El proyecto debe ser compilado o empaquetado correctamente para el despliegue.

Número de historia	**13PP-Developer**
Historia: Preparar el backend para el despliegue.	**Como** developer, **Quiero** desplegar el backend en el servidor de producción, **Para** que la aplicación esté accesible y funcione correctamente para los usuarios.
Criterios de aceptación	● El backend debe ser desplegado sin problemas en el servidor (DigitalOcean) ● El servidor debe estar configurado correctamente (puertos, balanceo de carga, bases de datos). ● Las API y servicios backend deben estar funcionando correctamente en producción.

Número de historia	**14PP-Developer**
Historia: Preparar el frontend para el despliegue.	**Como** developer, **Quiero** preparar el código del frontend para el despliegue, asegurándome de que todos los archivos y configuraciones estén listos, **Para** ser desplegados en el entorno de producción.
Criterios de aceptación	● El código debe ser optimizado para producción (minificación,

	optimización de imágenes, etc.). ● Las dependencias deben estar actualizadas y listadas en el archivo de configuración (por ejemplo, `package.json`). ● Las variables de entorno deben estar configuradas adecuadamente para el entorno de producción.

Número de historia	**15PP-Developer**
Historia: Preparar el frontend para el despliegue.	**Como** developer, **Quiero** desplegar el frontend en el servidor de producción, **Para** que la aplicación esté accesible y funcione correctamente para los usuarios.
Criterios de aceptación	● El frontend debe estar accesible a través de la URL de producción. ● La aplicación debe cargar correctamente, sin errores en la consola del navegador. ● El sistema debe ser probado en el entorno de producción para asegurar que no haya fallos en la interfaz de usuario.