

UNIVERSIDAD INTERNACIONAL SAN ISIDRO  
LABRADOR

NOMBRE DEL PROYECTO FINAL  
PROGRAMADO:  
PLATAFORMA WEB PARA CONSUMIDORES  
Y PRODUCTORES DE CAFÉ

NOMBRE DEL  
ESTUDIANTE/DESARROLLADOR:  
VICTOR ALONSO FALLAS ROJAS

CEDULA:  
604550381

PROFESOR/CURSO:  
ESTEFANÍA BOZA VILLALOBOS

PROGRAMACION AVANZADA  
ISB-32

FECHA:  
22 DE NOVIEMBRE 2025

## Tabla de Contenido

Parte I: Información General del Proyecto.....	4
Objetivo General.....	5
Objetivos Específicos .....	5
Justificación del Proyecto .....	6
Alcances Esperados.....	7
Requerimientos del Proyecto.....	8
Parte II: Diseño del Sistema (Pre-diseño visual).....	12
Elementos de Diseño.....	12
Explicación .....	12
Parte III: Base de Datos .....	17
Definición.....	17
1. Tipos de Datos.....	17
3. Normalización .....	19
Aunque MongoDB no utiliza normalización clásica (1FN, 2FN, 3FN), se aplican principios equivalentes: Sin datos repetidos (evitar redundancia de productor o producto).....	19
4. Triggers o funciones.....	19
• Middleware (Hooks) de Mongoose.....	19
Funciones de agregación .....	20
5. Conexión.....	20
Parte IV: Desarrollo del Backend.....	21
Arquitectura del Backend.....	21
1. Arquitectura Monolítica.....	21
2. Arquitectura por Capas (Recomendada para este proyecto).....	21
3. Arquitectura de Microservicios .....	22
4. Arquitectura Serverless / Event-driven .....	22
Elementos Clave del Backend .....	23

1. Seguridad.....	23
2. Gestión de Errores .....	23
3. Servicios Externos.....	23
4. Pruebas.....	23
5. Escalabilidad .....	24
Conclusión .....	25
Evidencia en Video .....	25
Referencias.....	26

## **Parte I: Información General del Proyecto**

El proyecto tiene como propósito desarrollar una plataforma web que conecte a consumidores y productores de café, con el objetivo de facilitar el acceso a información verificada sobre origen, métodos de procesamiento y sostenibilidad. Esta plataforma debe ofrecer una experiencia de compra segura y transparente, impulsar la trazabilidad del producto y apoyar a los pequeños productores mediante herramientas de venta, gestión de inventario y análisis simples. En conjunto, se busca promover prácticas justas y mejorar la percepción y satisfacción del usuario final.

La visión del proyecto es crear un ecosistema digital donde consumidores y productores coexistan en un flujo de información fiable, reduciendo la brecha entre origen y experiencia de compra. Para los consumidores, la plataforma ofrecerá acceso a información verificada sobre origen, variedades, región y procesos, junto con una experiencia de compra segura y transparente que incluya claridad en precios, plazos y políticas. Además, se propondrán recomendaciones basadas en perfiles de sabor, certificaciones y prácticas sostenibles. En beneficio de los productores, la plataforma proporcionará venta directa con mayor control sobre precios y promociones, herramientas de gestión de inventario, ventas y desempeño, y una mayor visibilidad que puede respaldar certificaciones y marketing. A nivel de la cadena de valor, se favorecerá la trazabilidad integral, la transparencia y el cumplimiento de prácticas de comercio justo, así como la capacidad de medir impactos sociales y ambientales a través de indicadores claros.

El alcance del proyecto se orienta a dos actores clave: consumidores y productores. Para los consumidores, se contemplan funciones como la exploración de cafés por origen, variedad y perfil de sabor, perfiles de productores con información de prácticas agrícolas y métodos de procesamiento, un carrito de compras con pagos seguros y seguimiento de pedidos, además de reseñas, calificaciones y recomendaciones personalizadas.

## **Objetivo General**

Desarrollar una plataforma web que conecte a consumidores y productores de café, facilitando el acceso a información verificada sobre origen, métodos de procesamiento y sostenibilidad. La plataforma debe permitir una experiencia de compra segura y transparente, impulsar la trazabilidad del producto y apoyar a los pequeños productores mediante herramientas de venta, gestión de inventario y análisis sencillos, con la finalidad de promover prácticas justas y mejorar la percepción y satisfacción del usuario final.

## **Objetivos Específicos**

### **1. Establecer y verificar la trazabilidad del café**

Implementar un registro completo del origen, variedad, región, método de procesamiento y certificaciones para cada lote, de modo que consumidores y productores puedan acceder a información verificada en cualquier punto de la cadena de valor.

### **2. Facilitar una experiencia de compra segura y transparente**

Diseñar y desplegar una experiencia de compra intuitiva, segura y transparente, con información clara sobre precios, plazos de entrega, políticas y protección de datos, para aumentar la confianza del usuario y la tasa de conversión.

### **3. Empoderar a los productores con herramientas de gestión.**

Proveer módulos de gestión de inventario, precios, promociones, análisis de ventas y dashboards específicos, para que las fincas, cooperativas y microempresas puedan tomar decisiones informadas.

### **4. Fomentar la sostenibilidad y el comercio justo**

Incorporar indicadores de sostenibilidad y prácticas de comercio justo, facilitar certificaciones y generar reportes periódicos de impacto para productores y consumidores, contribuyendo a prácticas responsables en toda la plataforma.

### **Justificación del Proyecto**

Este proyecto surge como respuesta a la necesidad de cerrar la brecha entre la calidad y trazabilidad del café de origen y la experiencia de compra del consumidor. En un mercado cada vez más competitivo y consciente de prácticas sostenibles, existen importantes desafíos: la falta de información verificada sobre origen y procesos de producción, la opacidad de precios y costos a lo largo de la cadena de valor, y el limitado acceso de pequeños productores a herramientas de gestión y comercialización. Al abordar estas problemáticas, la plataforma propuesta permite mejorar la transparencia y trazabilidad; facilitar compras seguras y responsables; empoderar a los productores; y promover prácticas sostenibles y justas.

La plataforma facilitará disponer de registros verificados de origen, región, variedad, métodos de procesamiento y certificaciones para cada lote, favoreciendo la confianza del consumidor y el cumplimiento de estándares de sostenibilidad y comercio justo. Al ofrecer un entorno de compra con pagos seguros, información clara de precios y políticas, y seguimiento de pedidos, se fortalece la confianza del usuario y la lealtad a la plataforma. Además, al proporcionar herramientas de gestión de inventario, precios, promociones y análisis de ventas, se mejora la eficiencia operativa, la visibilidad de mercado y la capacidad de tomar decisiones informadas por parte de los productores. Mediante indicadores de sostenibilidad, certificaciones facilitadas y reportes de impacto, se incentivan mejoras continuas en las prácticas agrícolas y comerciales a lo largo de la cadena de valor.

La adopción de una solución digital integral permite aprovechar los datos para la toma de decisiones, reducir costos operativos asociados a la intermediación y mejorar la experiencia del usuario. Dado que se trata de un producto nuevo en desarrollo ágil, se facilita la validación temprana del valor propuesto, la iteración rápida y la obtención de retroalimentación real de consumidores y productores. E

### **Alcances Esperados**

1. Registro, consulta y gestión de información clave del café
2. Registrar, verificar y consultar datos de origen (finca/cooperativa), región, variedad, métodos de procesamiento y certificaciones por lote.
3. Acceso rápido y verificado para consumidores y productores, con actualizaciones en tiempo real ante cambios o certificaciones nuevas.
4. Verificación de la información y control de acceso basado en roles (usuarios, productores, administradores.).
5. Generación de reportes y estadísticas para la toma de decisiones
6. Generación de informes y dashboards sobre ventas, inventario, demanda por origen, rendimiento de promociones y trazabilidad.
7. Acceso a la información en tiempo real
8. Disponibilidad de datos en tiempo real o cercano a él para eventos críticos (estado de pedidos, verificación de lotes y disponibilidad de inventario).
9. Integración de actualizaciones de estado en el flujo de compra y gestión de inventario.
10. Soporte a la experiencia de compra segura y transparente
11. Mantenimiento y mejora del carrito de compras, pagos seguros, seguimiento de pedidos y políticas claras.
12. Protección de datos personales y cumplimiento normativo aplicable a pagos y privacidad.
13. Transparencia de costos y condiciones de venta, incluyendo políticas de devolución cuando aplique.
14. Herramientas de gestión para productores
15. Gestión de inventario, definición de precios y promociones, y herramientas de análisis de ventas.
16. Dashboards de desempeño adaptados a fincas, cooperativas y microempresas.
17. Interoperabilidad y cumplimiento técnico



**18.** Integración con pasarelas de pago, servicios de envío y herramientas de marketing.

**19.** Normativas de seguridad, privacidad y cumplimiento vigentes consideradas desde el diseño.

## **Requerimientos del Proyecto**

### **1) Requerimientos funcionales**

#### **1.1 Registro y gestión de usuarios**

- Registro de usuarios: consumidores y productores pueden crear cuentas con verificación de correo.
- Autenticación y roles: login seguro; roles definidos (Consumidor, Productor, Administrador).
- Gestión de perfiles:
- Consumidores: preferencias de sabor, direcciones, métodos de pago, historial de compras, reseñas.
- Productores: datos de finca, certificaciones, métodos de procesamiento, inventario, precios y promociones, historial de ventas.

#### **1.2 Catálogo y trazabilidad del café**

- Registro de lotes: origen, variedad, región, método de procesamiento, certificaciones, lote, fecha de cosecha, productor(es).
- Trazabilidad: trazado completo desde lote hasta unidad de venta, con historial de movimientos y cambios.
- Perfiles de productores: información de prácticas agrícolas, métodos de procesamiento, certificados y prácticas de sostenibilidad.
- Exploración y filtrado: por origen, variedad, región, perfil de sabor, certificaciones, procesos.

#### **1.3 Compras y pagos**



- Carrito de compras con resumen de precios, costos de envío y impuestos.
- Procesos de pago seguros: integraciones con pasarelas de pago (SINPE), cumplimiento
- Transparencia de precios y plazos: desglose de costos, estimaciones de entrega, políticas de devolución.
- Seguimiento de pedidos: estado en tiempo real, notificaciones de cambios.
- Políticas y protección de datos: políticas de privacidad, términos de uso accesibles.

#### 1.4 Gestión de inventario y ventas para productores

- Inventario: stock por lote, lotes pendientes, alertas de agotamiento.
- Precios y promociones: gestión de precios por lote, descuentos, ofertas y condiciones.
- Análisis de ventas y desempeño: dashboards con indicadores clave (volumen, ingresos, margen, ventas por región).
- Gestión de pedidos: procesamiento, synchronized with inventario, estados de envío.
- 1.5 Evaluaciones y contenido
- Reseñas y calificaciones de productos y productores.
- Contenido verificado: información de origen, prácticas agrícolas, métodos de procesamiento verificada.

## 2) Requerimientos no funcionales

### 2.1 Seguridad

- Cifrado en tránsito y en reposo (TLS/HTTPS, cifrado de bases de datos).
- Gestión de secretos: almacenamiento seguro de claves y tokens.
- Cumplimiento de normativas de privacidad y protección de datos.

### 2.2 Rendimiento

- Tiempos de respuesta: página inicial < 2–3 s en condiciones normales; operaciones críticas < 1–2 s.
- Optimización de consultas: índices adecuados en búsquedas por origen, variedad, región.

## 2.3 Escalabilidad

- Arquitectura modular: microservicios o módulos bien acoplados.
- Escalabilidad horizontal: capacidad de escalar componentes del backend y de la base de datos.
- Despliegue continuo y CI/CD: pipelines para pruebas, build y deploy.

## 2.4 Usabilidad y accesibilidad

- Diseño responsivo para dispositivos móviles y escritorio.
- UI intuitiva con navegación clara de información de origen y políticas.
- Accesibilidad: cumplimiento básico de WCAG (color, contraste, lectura de pantallas).

## 2.5 Disponibilidad y resiliencia

- Alta disponibilidad con clústeres y replicación de base de datos.
- Backups regulares y pruebas de recuperación.
- Monitoreo y alertas de rendimiento y seguridad.

## 2.6 Mantenibilidad

- Código limpio y documentado.
- Pruebas automatizadas (unitarias, de integración, end-to-end).
- Gestión de configuración y secretos en entornos distintos (dev, prod, staging).

### **3) Infraestructura mínima**

#### **3.1 Servidor y hosting**

- Procesador Intel i5 o equivalente, 8 GB RAM, 250 GB disco, conexión a internet estable.

#### **3.2 Base de datos**

- MongoDB Atlas, configurado como servicio gestionado en la nube, con capacidad para manejar múltiples usuarios concurrentes, escalado automático y alta disponibilidad.
- CI/CD: GitHub Actions, GitLab CI o equivalent. SQL Server o MySQL, configurada para soportar múltiples usuarios concurrentes.

- 3.4 Seguridad y cumplimiento de infraestructura
- Servidor de correo para notificaciones automáticas, navegador moderno para acceso de usuarios, y sistema operativo compatible con el framework elegido.

## Parte II: Diseño del Sistema (Pre-diseño visual)

El diseño presentado corresponde a un pre-diseño, es decir, una propuesta inicial de cómo se espera que luzca el sistema visual de la marca Trading Coffee Costa Rica.

En este corto proceso, se buscó mantener un balance y equidad visual, con el fin de ofrecer una experiencia coherente y atractiva tanto para el usuario consumidor como para el usuario vendedor dentro de la plataforma.

El objetivo general fue definir una línea visual representativa y funcional, capaz de reflejar la esencia del café de especialidad costarricense y fortalecer la conexión entre productores, tostadores y consumidores.

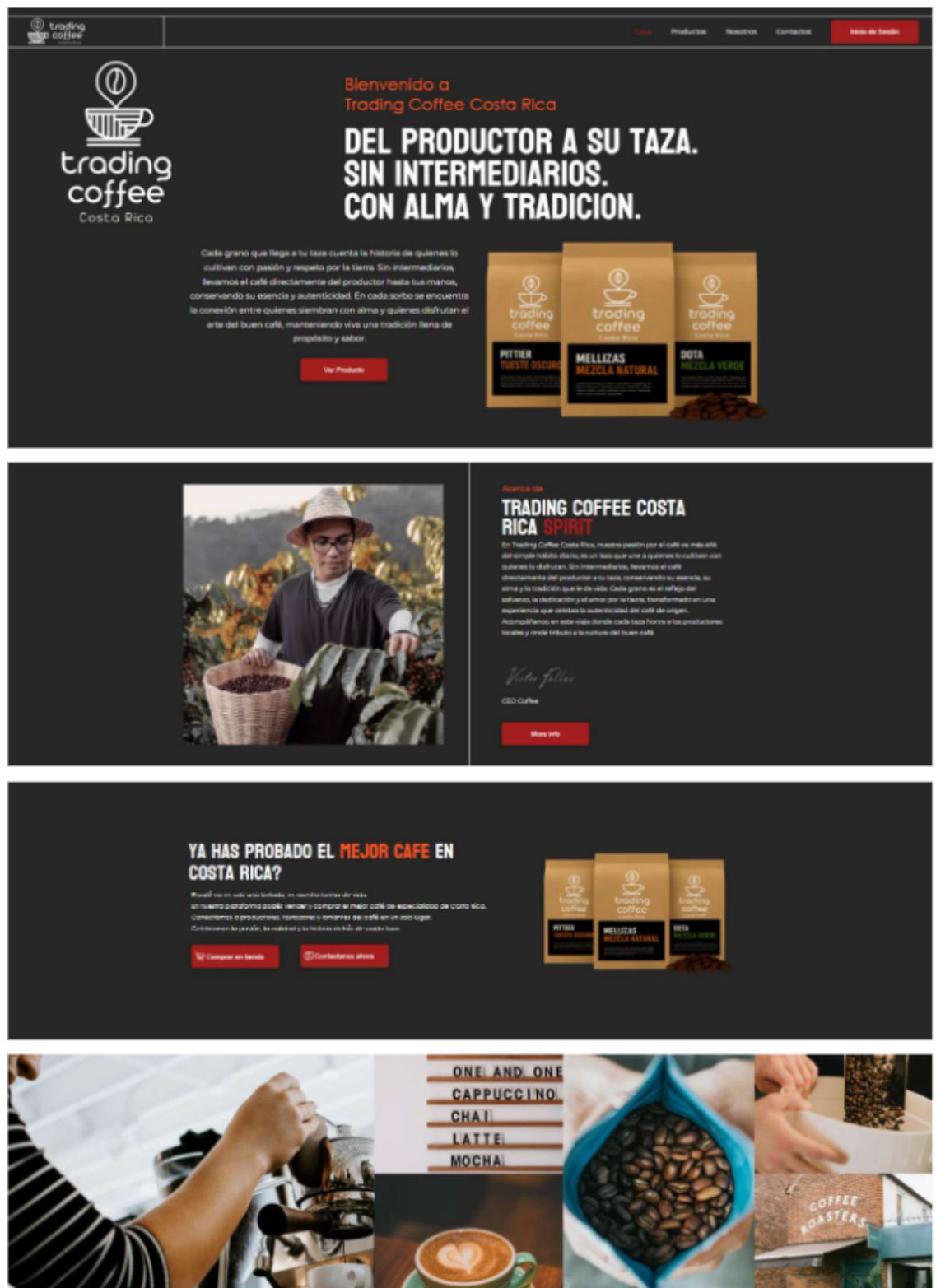
### Elementos de Diseño

- Incluye las principales secciones del sistema: **login, panel principal, menús de navegación, formularios de registro y reportes**. Esta estructura busca ofrecer una navegación intuitiva y organizada para ambos tipos de usuarios (vendedor y consumidor).
- Colores: Paleta basada en turquesa y gris claro para transmitir modernidad y simplicidad.
- Tipografía: Se utilizaron las fuentes **Bayon, Montserrat y Century Gothic**, seleccionadas por su combinación entre modernidad, legibilidad y estilo contemporáneo.
- Imágenes: Uso de fotografías realistas y empaques de café, destacando la autenticidad del producto.
- Íconos: Uso de librerías estándar (ej. FontAwesome, Lucide-React).
- Slogan: Origen & Esencia'.

### Explicación

El prediseño sirve como una guía visual que permitirá a los usuarios y al equipo de desarrollo comprender cómo se verá el sistema antes de su implementación, facilitando la toma de decisiones y asegurando una base sólida para el desarrollo final.

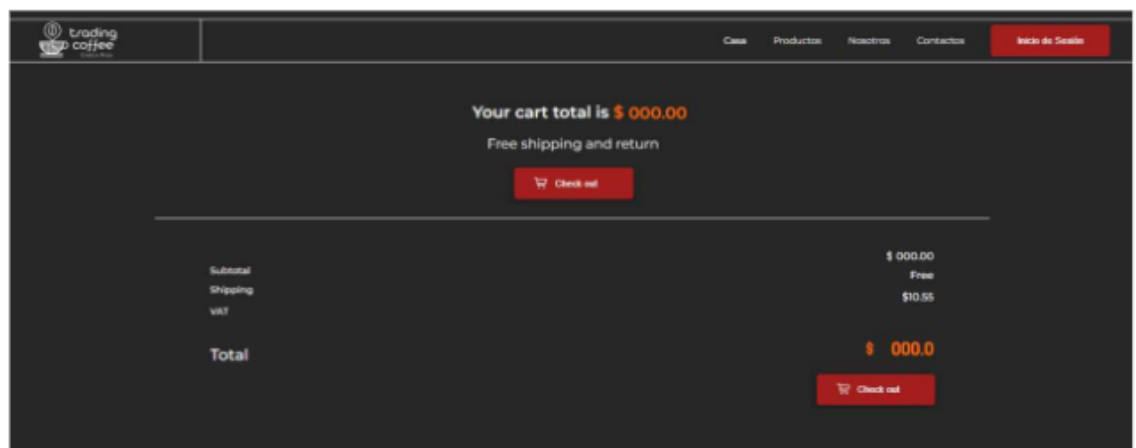
## 1. Pagina Principal (Main)



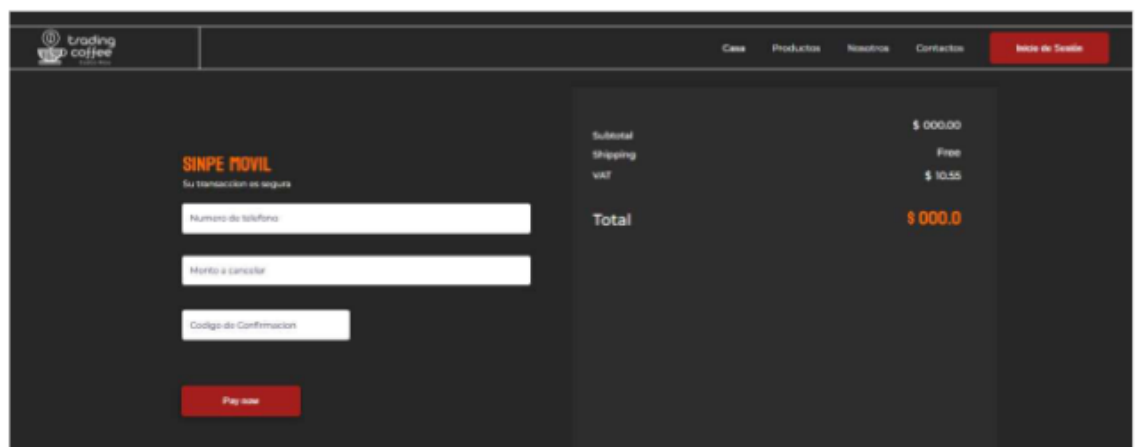
## 2. Vista de selección del producto o Market



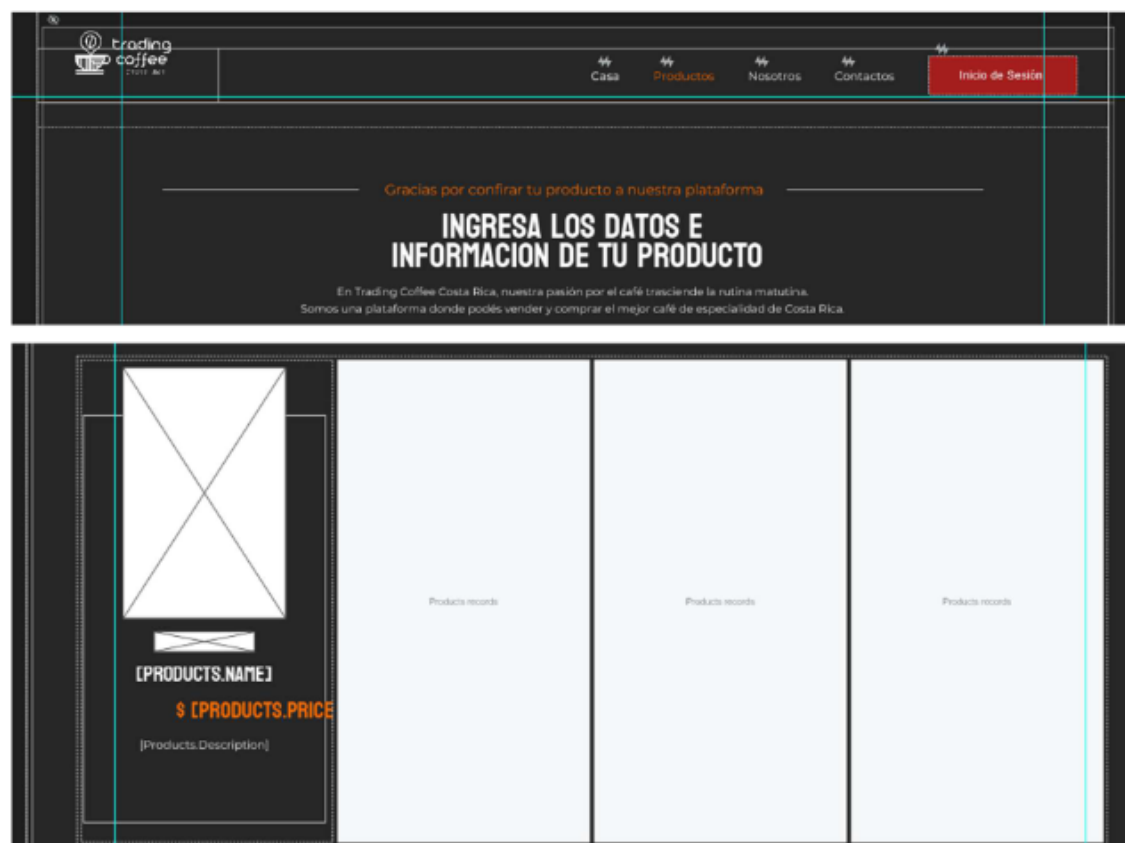
## 3. Generación de Factura.




## 4. Seccion de Pago.

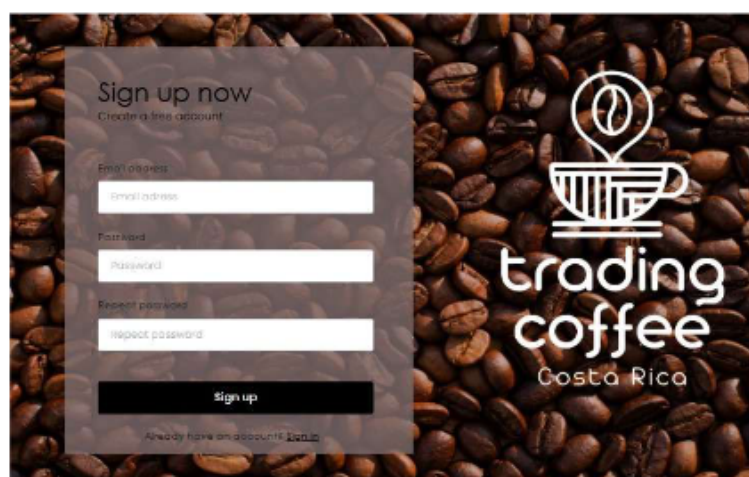


## 5. Experiencia del Productor, para ingresar el producto a la plataforma



 [PRODUCTS.NAME] \$ [PRODUCTS.PRICE] [Products.Description]	Producto records	Producto records	Producto records

## 6. Creación de cuenta; Esto lo utilizara el productor a la hora de ingresar para poder postular su producto





7. Inicio de Sesión; Esto lo utilizara el productor a la hora de ingresar para poder postular su producto



## Parte III: Base de Datos

### Definición

La base de datos del sistema estará basada en un modelo **NoSQL orientado a documentos**, estructurada mediante **colecciones** que representan las principales entidades del negocio (usuarios, productores, consumidores, productos, lotes, pedidos, pagos y reseñas). Este diseño permite flexibilidad, escalabilidad y eficiencia en las operaciones de lectura y escritura, especialmente en aplicaciones desarrolladas con Node.js.

El modelo se implementará en **MongoDB Atlas**, utilizando documentos con identificadores **ObjectId** y referencias entre colecciones para mantener consistencia en la información. Además, se aplicarán validaciones a través de **Mongoose**, lo que asegura integridad y control sobre la estructura de los datos.

### 1. Tipos de Datos

Se emplearán tipos nativos de MongoDB como **String, Number, Boolean, Date, ObjectId y Array**, asegurando precisión y compatibilidad con las operaciones del sistema.

- **Usuario** (nombre, correo, contraseña\_hash, rol, activo)
- **Productor** (usuarioid, nombre\_finca, ubicación, certificaciones)
- **Consumidor** (usuarioid, preferencias, dirección)
- **Producto** (productoid, nombre, variedad, región, método\_proceso, precio, stock, activo)
- **Lote** (productoid, fecha\_cosecha, certificación, cantidad)
- **Pedido** (consumidorId, detalles[], total, estado, fecha)
- **DetallePedido** (productoid, cantidad, subtotal)
- **Reseña** (productoid, consumidorId, calificación, comentario, fecha)
- **Pago** (pedidoid, método, monto, estado, fecha)

### Relaciones:

- Un **Productor** tiene muchos **Productos**.
- Un **Producto** pertenece a un **Lote** o varios.
- Un **Pedido** tiene muchos **Detalles** (como subdocumentos).
- Un **DetallePedido** referencia un **Producto**.
- Un **Consumidor** puede hacer muchos **Pedidos y Reseñas**.

En MongoDB estas relaciones se gestionan mediante **referencias (ObjectId)** entre colecciones y mediante operaciones de agregación cuando corresponde.

## 2. Llaves primarias y foráneas:

En MongoDB, cada documento posee una clave primaria única denominada **\_id**, generada automáticamente como **ObjectId**.

Las relaciones entre colecciones se implementan mediante campos que almacenan **ObjectId**, equivalentes funcionalmente a las llaves foráneas (FK) en bases de datos SQL.

Ejemplo base usando Mongoose:

```
const Usuario = new mongoose.Schema({
  nombre: String,
  correo: { type: String, unique: true },
  contraseña_hash: String,
  rol: { type: String, enum: ["Consumidor", "Productor", "Administrador"] },
  activo: { type: Boolean, default: true }
});

const Producto = new mongoose.Schema({
  productorId: { type: mongoose.Schema.Types.ObjectId, ref: "Productor"
},
  nombre: String,
  variedad: String,
  region: String,
  metodo_proceso: String,
  precio: Number,
  stock: Number
});
```

### 3. Normalización

Aunque MongoDB no utiliza normalización clásica (1FN, 2FN, 3FN), se aplican principios equivalentes: Sin datos repetidos (evitar redundancia de productor o producto).

- Evitar datos repetidos innecesarios (sin redundancia excesiva).
- Cada valor pertenece a un solo documento de forma coherente.
- Las dependencias lógicas se separan en colecciones independientes (por ejemplo, **Usuario** y **Productor**).
- Subdocumentos se usan cuando los datos pertenecen a una unidad lógica inseparable (por ejemplo, **DetallePedido** dentro de **Pedido**).

Este enfoque garantiza consistencia y buen rendimiento.

### 4. Triggers o funciones

- MongoDB no usa triggers tradicionales, pero se emplean:
- **Middleware (Hooks) de Mongoose**
- Permiten ejecutar lógica automáticamente al crear, actualizar o eliminar documentos.
- Ejemplo: actualización de inventario al generar un pedido:

```
PedidoSchema.post("save", async function (pedido) {  
  for (const item of pedido.detalles) {  
    await Producto.updateOne(  
      { _id: item.productoId },  
      { $inc: { stock: -item.cantidad } }  
    );  
  }  
});
```

## Funciones de agregación

Se utilizan para generar reportes, calcular totales y procesar información:

- Cálculo del total de ventas por productor.
- Listado de pedidos por consumidor.
- Agregaciones de stock por lote.

## 5. Conexión

La conexión con la base de datos se realizará mediante **Mongoose**, empleando una cadena de conexión segura provista por **MongoDB Atlas**, almacenada en un archivo de configuración o variables de entorno.

Ejemplo:

```
mongoose.connect(process.env.MONGO_URI)

  .then(() => console.log("Conectado a MongoDB Atlas"))

  .catch(err => console.error("Error en la conexión", err));
```

## Parte IV: Desarrollo del Backend

El backend constituye la lógica central del sistema. Se encarga de procesar la información, garantizar los mecanismos de seguridad y coordinar la comunicación entre la base de datos, el frontend y los servicios externos. En este proyecto se utiliza **Node.js con Express** como framework principal y **MongoDB Atlas** como base de datos en la nube, lo que permite flexibilidad, escalabilidad y alta disponibilidad.

### Arquitectura del Backend

El diseño backend puede variar según la complejidad del sistema y los requisitos del proyecto. Las arquitecturas utilizadas con mayor frecuencia son las siguientes:

#### 1. Arquitectura Monolítica

En este modelo todo el sistema se ejecuta dentro de una única aplicación de Node.js que contiene rutas, controladores, modelos y lógica de negocio.

##### Características:

- Un solo punto de despliegue.
- Toda la lógica se agrupa en un único paquete o servidor.

##### Ventaja:

- Facilidad de implementación en proyectos pequeños.

##### Desventaja:

- Dificultad para escalar cuando la aplicación crece.

#### 2. Arquitectura por Capas (Recomendada para este proyecto)

La aplicación se divide en módulos separados según su función. La estructura típica en Node.js con Express incluye:

- **Capa de Presentación:** rutas y controladores que reciben solicitudes HTTP.
- **Capa de Negocio:** servicios donde se implementan las reglas y flujos del sistema.
- **Capa de Datos:** modelos Mongoose y operaciones sobre MongoDB Atlas.

**Ventajas:**

- Mayor orden y mantenibilidad.
- Estructura clara para extender funcionalidad.

**3. Arquitectura de Microservicios**

El sistema se divide en múltiples servicios independientes que se comunican mediante APIs o colas de mensajes.

**Características:**

- Cada servicio administra su propia lógica y base de datos o colecciones.
- Se pueden escalar módulos de forma independiente.

**Ventaja:**

- Alto nivel de escalabilidad y modularidad.

**Desventaja:**

- Mayor complejidad en la gestión, monitoreo y comunicación entre servicios.

**4. Arquitectura Serverless / Event-driven**

El procesamiento se ejecuta mediante funciones en la nube (como AWS Lambda, Firebase Functions o Vercel Functions), las cuales se activan bajo demanda.

**Ventajas:**

- No requiere administración directa de servidores.
- Optimiza costos cuando existen cargas variables.

**Desventajas:**

- Complejidad para flujos largos o transacciones.
- Dependencia de proveedores externos.



## Elementos Clave del Backend

Independientemente de la arquitectura empleada, todo backend debe considerar varios componentes fundamentales:

### 1. Seguridad

Incluye mecanismos que garantizan la integridad y protección de los datos:

- Autenticación y autorización mediante **JWT**.
- Hash seguro de contraseñas utilizando **bcrypt**.
- Control de acceso por roles: Consumidor, Productor y Administrador.
- Validación y sanitización de datos para evitar ataques como inyección o XSS.

### 2. Gestión de Errores

Es necesario contar con una estrategia que permita identificar, registrar y manejar fallos de forma controlada:

- Middleware global para capturar excepciones.
- Registro de errores mediante herramientas como Winston o Logtail.
- Respuestas consistentes en formato JSON para facilitar la interpretación desde el frontend.

### 3. Servicios Externos

El backend puede integrar servicios complementarios como:

- Pasarelas de pago (SINPE Móvil).
- Sistemas de envío y logística.
- Plataformas de correo o notificaciones.
- Webhooks para recibir confirmaciones externas (por ejemplo, pagos aprobados).

Estas integraciones se implementan mediante módulos o controladores especializados.

### 4. Pruebas

Para asegurar el funcionamiento adecuado del sistema, se deben realizar pruebas en distintos niveles:

- **Unitarias:** validan funciones individuales y servicios.

- **Integración:** prueban la interacción entre rutas, controladores y base de datos (MongoDB).
- **Funcionales / E2E:** verifican el comportamiento completo del sistema desde el punto de vista del usuario.

Herramientas típicas: Jest, Supertest y otros entornos de prueba.

## 5. Escalabilidad

El backend debe estar preparado para soportar una mayor demanda con el tiempo. Algunas estrategias incluyen:

- Uso de **MongoDB Atlas** con escalado automático.
- Balanceo de carga mediante Nginx u otro proxy.
- Implementación de caching con Redis para consultas intensivas.
- Separación eventual de módulos en microservicios si la carga lo requiere.
- Contenedorización con Docker para lograr despliegues consistentes.

## Conclusión

El desarrollo del proyecto **Tradding Coffee CR** constituye un avance sustancial en la modernización y automatización de los procesos asociados a la gestión del café y la interacción entre productores, consumidores y administradores. La integración de un backend construido con **Node.js y Express**, junto con una base de datos alojada en **MongoDB Atlas**, proporciona una estructura tecnológica robusta, escalable y orientada al crecimiento continuo de la plataforma.

La definición de objetivos claros, el diseño visual preliminar y la arquitectura backend seleccionada permiten garantizar un flujo de información seguro, eficiente y adaptable a las necesidades del negocio. Asimismo, el uso de tecnologías modernas y herramientas basadas en la nube posiciona a **Tradding Coffee CR** como una solución preparada para operar en entornos de alta demanda y expansión futura.

Se concluye que la estrategia utilizada no solo cumple con los requerimientos actuales del sistema, sino que establece una base sólida para incorporar nuevas funcionalidades, extender servicios, integrar herramientas externas y fortalecer el ecosistema digital de la marca. Con esta estructura tecnológica, **Tradding Coffee CR** se consolida como un proyecto capaz de evolucionar de manera sostenida y responder a los retos del mercado digital contemporáneo.

## Referencias

Express.js. (s.f.). *Express: Fast, unopinionated, minimalist web framework for Node.js*. Recuperado de <https://expressjs.com/>

Node.js Foundation. (s.f.). *Node.js documentation*. Recuperado de <https://nodejs.org/>

MongoDB Inc. (s.f.). *MongoDB Atlas: Fully managed cloud database*. Recuperado de <https://www.mongodb.com/atlas>

MongoDB Inc. (s.f.). *MongoDB manual: Data models and document structure*. Recuperado de <https://www.mongodb.com/docs/manual/>

Mongoose Team. (s.f.). *Mongoose ODM documentation*. Recuperado de <https://mongoosejs.com/>

OpenJS Foundation. (s.f.). *JavaScript language resources*. Recuperado de <https://developer.mozilla.org/es/docs/Web/JavaScript>

Fowler, M. (2002). *Patterns of enterprise application architecture*. Addison-Wesley.

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3.<sup>a</sup> ed.). Addison-Wesley.

ISO/IEC 25010. (2011). *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)*. International Organization for Standardization.

Sistema de Certificación del Café de Costa Rica. (s.f.). *Estándares y certificaciones del café en Costa Rica*. Recuperado de <https://sca.cr/>

OpenAI. (2025). *ChatGPT (modelo GPT-5.1)* [Herramienta de IA generativa]. Recuperado de <https://chat.openai.com/>