REPOSITORIO DE CÓDIGO FUENTE Y

DOCUMENTACIÓN TÉCNICA FINAL

**Curso:** Administración de Proyectos de Software

**Sesión:** 16

**Docente:** Ing. José Vásquez Pereyra

# Introducción

Este documento describe la estructura, contenido y guía de uso del repositorio final de código fuente del sistema desarrollado, así como la documentación técnica definitiva que acompaña al producto software.

# Estructura del Repositorio

|  |  |
| --- | --- |
| **Carpeta / Archivo** | **Descripción de su Contenido** |
| **.vscode/** | Configuración específica del entorno de desarrollo en Visual Studio Code (extensiones, formato, etc.). |
| **node\_modules/** | Carpeta generada automáticamente que contiene todas las dependencias instaladas del proyecto. |
| **src/** | Carpeta principal del código fuente del sistema (rutas, controladores, vistas, modelos, etc.). |
| **src/chat/** | Contiene la lógica relacionada con el sistema de chat del sistema, si aplica. |
| **src/controllers/** | Archivos que gestionan la lógica del negocio y procesan las solicitudes desde las rutas. |
| **src/database/** | Archivos relacionados con la conexión a la base de datos y ejecución de procedimientos almacenados. |
| **src/locales/** | Archivos de traducción o configuraciones regionales si se usa internacionalización (i18n). |
| **src/public/** | Recursos públicos como imágenes, archivos estáticos, hojas de estilo y scripts de cliente. |
| **src/routes/** | Definición de las rutas de la aplicación. Conecta URLs con controladores correspondientes. |
| **src/views/** | Plantillas .ejs que representan la interfaz gráfica que el usuario final visualiza. |
| **src/index.js** | Punto de entrada principal del servidor. Configura la app, middlewares y servidor Express. |
| **.env** | Archivo de variables de entorno con configuración sensible como credenciales o URLs. |
| **package-lock.json** | Registro exacto de versiones de cada dependencia instalada, garantiza consistencia en instalaciones. |
| **package.json** | Archivo principal de configuración del proyecto Node.js. Contiene scripts, nombre del proyecto y dependencias. |

# Lenguajes y Herramientas Utilizadas

Durante el desarrollo del sistema web para el Hotel Casa Andina se emplearon los siguientes lenguajes, tecnologías y herramientas:

* **Lenguajes de Programación:**
  + JavaScript (lado del servidor con Node.js)
  + HTML5, CSS3 y JavaScript (lado del cliente)
* **Frameworks y Librerías:**
  + **Express.js:** Framework para la creación de servidores web en Node.js
  + **EJS:** Motor de plantillas para generar vistas dinámicas
  + **Bootstrap:** Estilos y componentes responsivos
  + **SweetAlert2:** Librería para alertas personalizadas
* **Base de Datos:**
  + **MySQL:** Sistema de gestión de base de datos relacional
  + Uso de **procedimientos almacenados** para la lógica de datos
* **Herramientas de Desarrollo:**
  + **Visual Studio Code:** Editor principal de código
  + **Git y GitHub:** Control de versiones y repositorio colaborativo
  + **Postman:** Pruebas de endpoints y APIs
  + **npm:** Gestor de dependencias de Node.js

# Instrucciones de Clonación y Ejecución

Para ejecutar el sistema web de Casa Andina de forma local, se deben seguir los siguientes pasos:

1. Clonar el repositorio desde GitHub:  
   git clone https://github.com/ErikUpnGitHub/Casa-Andina.git
2. Acceder a la carpeta del proyecto:  
   cd proyecto-casa-andina
3. Instalar las dependencias del proyecto:  
   npm install
4. Crear un archivo .env en la raíz del proyecto con las siguientes variables:  
   DB\_HOST=localhost  
   DB\_USER=root  
   DB\_PASSWORD=tu\_contraseña  
   DB\_NAME=casaandina\_db  
   PORT=3000
5. Ejecutar la aplicación:  
   npm start
6. Abrir la aplicación en el navegador:  
   <http://localhost:3000>

# Buenas Prácticas de Desarrollo

Durante el desarrollo del sistema web de Casa Andina se aplicaron las siguientes buenas prácticas:

* **Convenciones de Codificación:**  
  Se usó un estilo de codificación coherente en todo el proyecto, con nombres de variables y funciones en camelCase, indentación uniforme y comentarios explicativos en secciones clave del código.
* **Control de Versiones:**  
  Se utilizó Git como sistema de control de versiones. Cada funcionalidad fue desarrollada en ramas independientes y posteriormente integrada a la rama principal mediante *pull requests*.
* **Documentación en el Código:**  
  Se incluyeron comentarios descriptivos en funciones, clases y procedimientos almacenados, explicando su propósito y comportamiento.
* **Pruebas Unitarias Manuales:**  
  Aunque no se implementó un framework automatizado, se realizaron pruebas unitarias manuales sobre funciones clave para validar su correcto funcionamiento.
* **Organización del Código:**  
  El proyecto fue estructurado en carpetas separadas por responsabilidad (controllers, routes, views, etc.), facilitando la mantenibilidad y su escalabilidad.
* **Gestión de Entorno:**   
  Se utilizó un archivo .env para separar la configuración sensible del código fuente, siguiendo buenas prácticas de seguridad.

# Documentación Técnica Final

|  |  |
| --- | --- |
| **Documento** | **Descripción** |
| **Manual Técnico** | Describe la configuración, instalación y despliegue del sistema, así como detalles del entorno de desarrollo. |
| **SRS (Especificación de Requisitos del Software)** | Documento que define los requerimientos funcionales y no funcionales del sistema. |
| **Diseño Arquitectónico** | Explica la estructura del sistema, los componentes principales y su interacción. |
| **Plan de Pruebas** | Define el enfoque y tipos de pruebas realizadas para validar el sistema. |
| **Registro de Pruebas con Evidencias** | Contiene los casos de prueba ejecutados, sus resultados y capturas si aplica. |
| **Diccionario de Datos** | Describe las tablas, campos y relaciones utilizadas en la base de datos. |
| **Manual de Usuario (si aplica)** | Guía práctica para el uso del sistema por parte del usuario final. |

# Licenciamiento y Créditos

Este proyecto es privado y no cuenta con una licencia de uso público.

Queda estrictamente prohibida su copia, distribución, modificación o reutilización total o parcial sin autorización expresa del equipo desarrollador.

**Créditos del equipo de desarrollo:**

* *Chimbor de la Cruz Erik Antonio* – Desarrollo del sistema, backend, frontend, estructuración del repositorio y documentación técnica.
* *Martinez Rodriguez Nilver Jefferson* – Elaboración de la documentación técnica.
* *Menacho Monigote Jover Yerson* – Desarrollo del módulo de chatbot y documentación técnica.

# Aprobaciones y Validaciones del Documento

**Director del Proyecto**

* **Nombre:** Martinez Rodriguez, Nilver Jefferson
* Diagrama

  El contenido generado por IA puede ser incorrecto.**Firma:**
* **Fecha:** 12/05/2025

**Responsable de calidad**

* **Nombre:** Sánchez Saavedra, Brayan José
* Imagen que contiene Carta

  El contenido generado por IA puede ser incorrecto.**Firma:**
* **Fecha:** 12/05/2025

# Conclusiones

El presente documento presenta de forma organizada el repositorio de código fuente y la documentación técnica final del sistema web desarrollado para Casa Andina. La estructura del proyecto refleja una adecuada organización del código, facilitando su mantenimiento y escalabilidad.

Además, se aplicaron buenas prácticas de desarrollo, como el uso de control de versiones, separación de responsabilidades y documentación clara tanto en el código como en archivos complementarios.

Este repositorio representa el resultado de un trabajo colaborativo que integra funcionalidad, orden técnico y documentación completa, permitiendo una futura evolución o despliegue del sistema con base sólida y estructurada.

# Aprendizajes del Equipo

|  |  |
| --- | --- |
| **Aspecto Aprendido** | **Comentario del Equipo** |
| Organización del código fuente | Estructurar correctamente las carpetas facilitó la comprensión y mantenimiento del sistema. |
| Uso de control de versiones con Git | Trabajar con ramas y commits permitió un seguimiento claro y ordenado del desarrollo. |
| Importancia de documentar | La documentación técnica fue clave para garantizar que el sistema pueda ser comprendido por cualquier integrante o futuro desarrollador. |
| Manejo de entornos de desarrollo | Aprendimos a utilizar variables de entorno y separar configuraciones sensibles del código. |
| Colaboración y trabajo en equipo | El trabajo conjunto permitió mejorar la calidad del sistema y reforzar habilidades de comunicación técnica. |