MenUCA
Plataforma para centralizar menús, precios y promociones en la Zona Peatonal de la UCA

Diego Javier Alvarenga Hércules #00027222, Reynaldo Alcides Bonilla Ventura #00107117, Daniel Alejandro Meléndez Delgado #00078623, Gabriel Antonio Urquilla Zetino #00056122, Universidad Centroamericana José Simeón Cañas. Catedrático: Rony Remberto Santos Bautista.

***Resumen*—MenUCA es una plataforma web diseñada para centralizar la información de los locales de comida ubicados en la zona peatonal (La PEA) de la UCA. La plataforma permitirá consultar menús, precios, horarios y ubicaciones en tiempo real; además de ofrecer reseñas, calificaciones y notificaciones de promociones. En este primer avance se presenta la justificación del problema, la arquitectura propuesta (Frontend, Backend y Base de Datos), la selección tecnológica y un plan de implementación por módulos. También se describe la metodología de trabajo (Kanban), criterios de calidad y las entregas esperadas. El objetivo es facilitar la toma de decisiones de la comunidad universitaria y aumentar la visibilidad de los comercios.**

I. Introducción

En la zona peatonal de la UCA existe una amplia oferta de locales de comida, pero la información dispersa de cada local dificulta la elección rápida de los estudiantes y personal. Esto provoca pérdida de tiempo y oportunidades de promoción para comercios. El proyecto MenUCA busca digitalizar y centralizar esta información, ofreciendo acceso rápido a menús, precios, horarios, ubicación y reseñas desde una interfaz intuitiva.

Fundamento

Se desarrollará una plataforma web que centralice y digitalice la información de los locales de comida de la Zona Peatonal de la UCA, mostrando menús, precios, horarios y promociones. La herramienta contará con una interfaz intuitiva y una experiencia de usuario optimizada, facilitando el acceso rápido a la oferta gastronómica y aumentando la visibilidad de los negocios locales.

TABLE I
Alcances

Apariencias			
Item	**Estimación de tiempo**	**Recursos requeridos**	**Precio**
Modulo de bienvenida / login	3 días	1 FE, 1 QA	$600
Directorio de locales	5 días	1 FE, 1 BE, 1 QA	$925
Modulo de búsqueda y filtros	1 semana	2 FE, 1 BE, 1 QA	$1680
Edición de restaurante y verificación de datos	1 semana	2 FE, 1 BE, 1 QA	$1945
Sistema de reseñas y calificaciones	6 días	1 FE. 2 BE, 1 QA	$850

Este trabajo será completamente financiado y elaborado por el redactor quién se acredita tanto el diseño e implementación. por el equipo de estudiantes de Ingeniería informática de la Universidad "UCA" Diego Javier Alvarenga Hércules #00027222, Reynaldo Alcides Bonilla Ventura #00107117, Daniel Alejandro Meléndez Delgado #00078623, Gabriel Antonio Urquilla Zetino #0005612.

Fig. 1. Logo de la página web.

II. ARQUITECTURA A UTILIZAR

A. Diseño de solución

Para el diseño de la solución al problema planteado, lo que se pretende lograr es lo siguiente:

a) Haciendo uso de una arquitectura de producto modular, separar una solución en desarrollo frontend (FE) que permita al usuario final tener una herramienta móvil con los módulos propuestos para el desarrollo.
b) Manejando el mismo principio se plantea la creación de un backend o aplicación del lado del servidor basada en Web, que nos permita manejar la lógica, manipulación de datos y conectividad necesaria para la distribución del producto, a través de un REST API.
c) Se hará uso de una base de datos relacional administrada y manipulada por la aplicación de backend.
d) Para este desarrollo se propone el uso de tecnologías orientadas a la innovación y accesibles para todas las partes del desarrollo que esten involucradas.

B. Arquitectura del producto

Para este proyecto se implementará una Arquitectura de producto basada en la combinación de los productos entregados para Frontend (FE), Backend (BE), Base de Datos (BDD).

● Frontend: Framework: React.js con componentes funcionales y hooks

Gestión de Estado: Context API para estado global (autenticación, sesiones de usuario)

● Backend: Node.js con framework Express.js Endpoints

RESTful para operaciones CRUD
Middleware de validación de entrada y manejo de errores

// Autenticación
POST /api/auth/registro
POST /api/auth/login

// Restaurantes
GET /api/restaurantes
GET /api/restaurantes/:id
POST /api/restaurantes (admin)

// Reseñas
GET /api/resenas/restaurante/:id
POST /api/reseñas

● Base de datos: MySQL o SLQ Server (relacional, robusto).

Entidades Implementadas: Usuarios, Restaurantes, ItemsMenu, Reseñas, Promociones.

C. Solución esperada

El entregable final constará de un informe detallando los logros y avances obtenidos cómo resultado de la investigación e implementación del proyecto. Así mismo de un repositorio único donde comparar precios, horarios y promociones. Los locales pierden visibilidad y la comunicación con el cliente es ineficiente. MenUCA pretende resolver estas brechas mediante una plataforma que conecte usuarios y comercios con datos actualizados.

III. TECNOLOGIAS A UTILIZAR

1. *Frontend (Interfaz de usuario – UI)*

- **HTML5, CSS, JavaScript**, base de toda aplicación web.
- *Framework recomendado: React.js, librería moderna, muy usada, con componentes reutilizables.*
- **Estilos:** CSS, aceleran el diseño y garantizan una experiencia responsiva.

```
Cliente-FE > src > utils > api.js > ...
1  import axios from "axios";
2
3  const API = axios.create({
4    baseURL: "http://localhost:3010/api",
5  });
6
7  API.interceptors.request.use((req) => {
8    const token = localStorage.getItem("token");
9    if (token) req.headers.Authorization = `Bearer ${token}`;
10   return req;
11 });
12
13 export const fetchCustomers = async (searchCode = '') => {
14     let url;
15
16     if (searchCode) {
17         url = `/customers/search?code=${searchCode}`;
18     } else {
19         url = "/customers";
20     }
21
22     const response = await API.get(url);
23     return response.data.data;
24 };
25
26 export const registerSale = async (saleData) => {
27     const response = await API.post("/sales", saleData);
28     return response.data;
29 };
30
```

Fig. 2. Fragmento de código de JavaScript

2. *Backend (Servidor y lógica de negocio)*

- Node.js/Express: Ecosistema robusto, ideal para desarrollo ágil de APIs REST

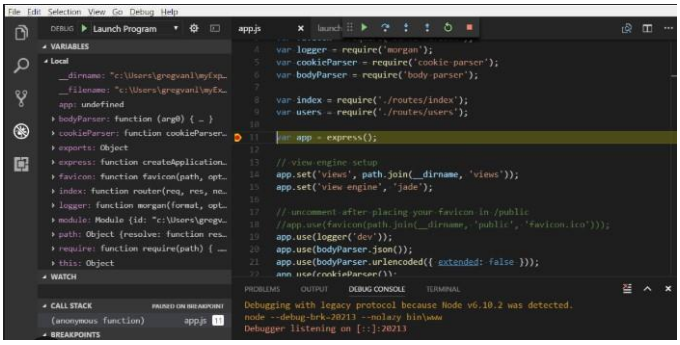


Fig. 3. Fragmento de código de NodeJS

3. *Base de Datos (BDD)*

- **MySQL** o SQLServer, las bases de datos relacionales permiten manejar tablas relacionadas como locales, menús, usuarios y reseñas de forma organizada y eficiente.

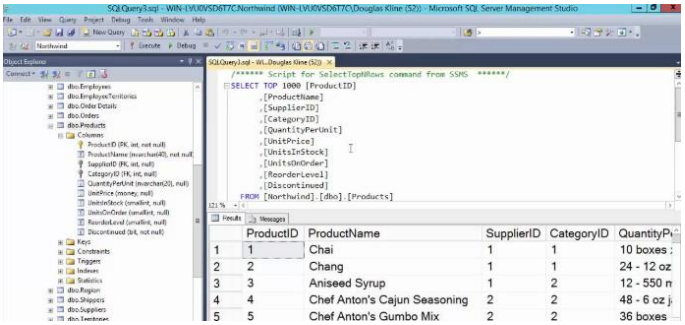


Fig. 4. Fragmento de código de SQLServes

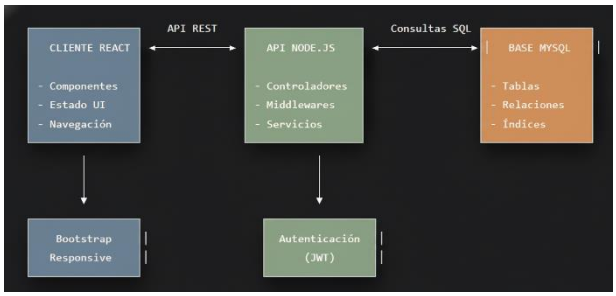
IV. HERRAMIENTAS COMPLEMENTARIAS

- **Git + GitHub**, control de versiones y trabajo en equipo.
- **Kanban** organización ágil de tareas.

V. DISEÑO DE IMPLEMENTACION

Para esta solución se ha definido un diseño de arquitectura modular que conecta el **Frontend** a través de un **Servicio Web** que opera en un entorno *Cloud Provider*. El sistema está conceptualizado bajo el patrón. **"Cliente – Servidor"** , asegurando que la aplicación se ejecute en vivo y esté disponible para el acceso constante de los usuarios. El núcleo de la solución se centra en una. **API REST**, la cual actúa como intermediario para el manejo de los servicios basados en web y los *endpoints* necesarios para la conectividad ***Client App*** (Capa de Presentación) emite una **Request** que es procesada por el Servidor Web (Backend) y luego gestionada por la **Backend App** y los **Cloud Services.**

Fig. 5. Diseño de solución a implementar



VI. SEGUIMIENTO DE LINEAMIENTO

Para la gestión del proyecto se utilizará la metodología ágil Kanban o Github, debido a que facilita la visualización del flujo de trabajo y la priorización de tareas. Se implementará un tablero digital (ej. Git o GitHub) con Commits como Pendiente, En proceso, En revisión y Finalizado.
Cada integrante del equipo asumirá responsabilidades en frontend, backend, base de datos o pruebas, y se realizarán reuniones semanales de seguimiento con el docente para verificar avances.

VII. PATRON DE DESARROLLO DE SOFTWARE

El proyecto seguirá una arquitectura Cliente–Servidor con separación clara de capas:

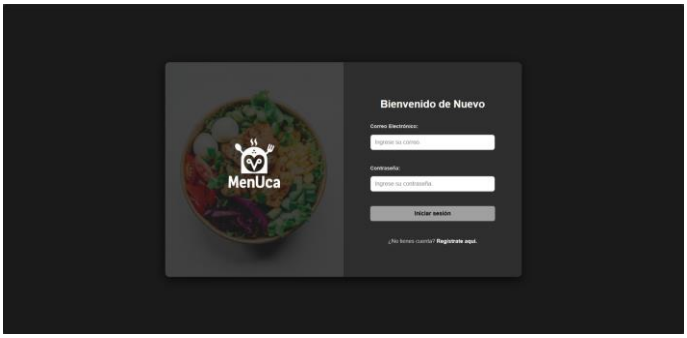
- Capa de presentación (Frontend): interfaz en React.
- Capa de aplicación (Backend): API REST en
- Node.js.
- Capa de datos: base de datos relacional SQLServer.

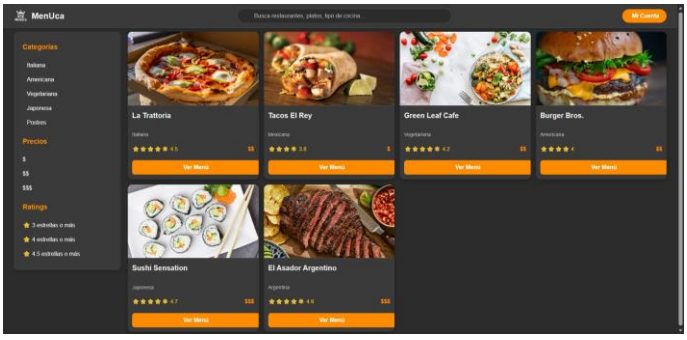
Para el código se aplicarán las siguientes buenas prácticas:

- Convenciones de nombres y consistencia en el estilo.
- Modularidad y reutilización de componentes.
- Manejo de errores y validación de entradas.
- Uso de control de versiones con GitHub.

Este patrón permitirá escalar fácilmente la plataforma y garantizar un mantenimiento ordenado.

VIII. ANEXOS





A. Manual Técnico

- Guía de instalación y configuración
- Documentación de la API
- Esquema de base de datos

B. Manual de Usuario

- Guía de uso de la plataforma
- Funcionalidades disponibles
- Solución de problemas comunes

REFERENCIAS

[1] Lámpsakos “Desarrollo de una plataforma web multimedial para la elaboración de proyectos bajo la metodología de marco lógico” https://www.academia.edu/download/105367725/2178.pdf (accessed Sep 19, 2025)
[2] React Documentation. "Getting Started with React". https://reactjs.org/docs/getting-started.html
[3] Node.js Foundation. "Node.js Guides". https://nodejs.org/en/docs/guides/
[4] MySQL. "SQL SERVER Reference Manual". Oracle Corporation, 2023.
[5] Jack Caulfield.. “IEEE Website Citation” Scribbr. https://www.scribbr.com/ieee/ieee-website-citation (accessed Agu. 25, 2025).