Universidad Rafael Landívar
Campus Central
Dirección de Tecnologías de la Información

Prueba Técnica para Analista Desarrollador II

Realizada por: Carlos Daniel Barrientos Castillo

Nueva Guatemala de la Asunción, agosto 24 del 2025

Tabla de contenido

Enfoque del Documento:

Dentro del siguiente documento se defiende la arquitectura utilizada para la prueba técnica para el puesto de Analista Desarrollador II en el Departamento de Tecnologías de la Información, el nombre del proyecto fue nombrado como “Foodie Test”, por lo cual dentro del proyecto y siguiente documento se encontrarán multiplex referencias a este nombre para recalcarlo como el nombre del proyecto.

En el siguiente documento se podrá encontrar todo lo necesario para la evaluación del proyecto desde una perspectiva más técnica y con un enfoque mas practico, por lo que se recomienda que se tengan los siguientes conceptos claros.

- API
- Json Web Token
- React
- Axios
- Servicios Web
- Bases de datos relacionales
- Node.js
- Express

Requerimientos:

Dentro del documento origen de la prueba técnica se especifican los siguientes requerimientos técnicos funcionales y no explictos. Por lo que el proyecto se limita a estos requerimientos

Requerimientos Funcionales:

- Registro de clientes
- Registro de compras
- Visibilidad Restringida para clientes
- Gestiones por administradores
- Envió de Ofertas o Anuncios
- Eliminación de Clientes
- Edición de datos básicos del cliente
- Interfaz grafica conectada a una API o servicio Web
- Autenticación de usuarios
- Autorización por perfiles
- CRUD de clientes parcial (Crear, leer, editar y eliminar, pero solo por administradores)
- CRUD de compras parcial (Crear, leer por cliente y lectura por administradores)

Requerimientos no explícitos:

- Almacenamiento seguro de credenciales (hash de contraseñas)
- Controles de autorización en API
- Privacidad por parte de los datos entre clientes
- Rendimiento de la API
- Disponibilidad y confiabilidad
- Manejo de errores con la interfaz grafica
- Usabilidad
- Pruebas básicas
- Portabilidad

Criterios de Aceptación:

- Un cliente puede registrarse, iniciar sesión, cerrar sesión y ver sus propias compras
- Un administrador puede iniciar sesión, cerrar sesión, ver el listado de clientes y sus detalles, editar clientes, eliminar usuarios y enviar ofertas/anuncios
- La UI consume la API, ambos entregables vienen con instrucciones
- El repositorio debe ser público, con una rama master y el commit final debe ser previo al 24/08/2025 a las 20:00

Enfoque del Producto:

El enfoque del producto es crear una aplicación web que pueda ayudar a una empresa ficticia a conectar a los usuarios con el negocio, esto por medio de envió de ofertas y anuncios a los usuarios por medio de la misma aplicación web. Los usuarios de la aplicación pueden comprar productos desde la misma aplicación desde un dashboard creado solo para ellos que lleva el registro de sus compras y anuncios.

Los administradores pueden realizar diferentes actividades por medio de un dashboard especial con los usuarios como lo seria editar la información de cada uno, eliminar a los usuarios del sistema y enviar las ofertas y anuncios

Arquitectura Utilizada:

Para la arquitectura se eligió la manera más rápida de completar el producto en el tiempo estimado, para lo cual fue necesario utilizar React para el frontend, utilizando Bootstrap y Node.js con Express para el backend, para la base de datos relacional se utilizó sqlite. A continuación, se presentan los puntos de la elección de las tecnologías.

React + Bootstrap:

Agilidad, se requiere de un proyecto que pueda estar listo en menos de 12 horas, por lo cual utilizar React que ya tiene sus propias maneras de distribuir components y separar las responsabilidades hizo más rápido el trabajo de desarrollo, en cuanto a Bootstrap, requeríamos una librería confiable que pudiera adaptarse a cualquier proyecto sin necesidad de definir muchos estilos y tener combinaciones por defecto para centrarse en la funcionalidad

Node.js + Express:

Confiabilidad, se necesitó una plataforma segura con lo cual el tener JWT para separar responsabilidades y sesiones de usuario que era algo indispensable para el proyecto, por lo cual el tener una plataforma con buena compatibilidad y fácil adoptabilidad era muy necesario, node.js por su parte nos ofrece una buena adaptabilidad entre con los JWT aparte de que crear una API con ayuda de Express nos daba mas facilidad y agilidad para no perder tiempo en configuraciones como lo hubiéramos tenido que hacer en .NET.

Diagramas de Conceptos:

Diagrama Entidad Relacion de la base de datos

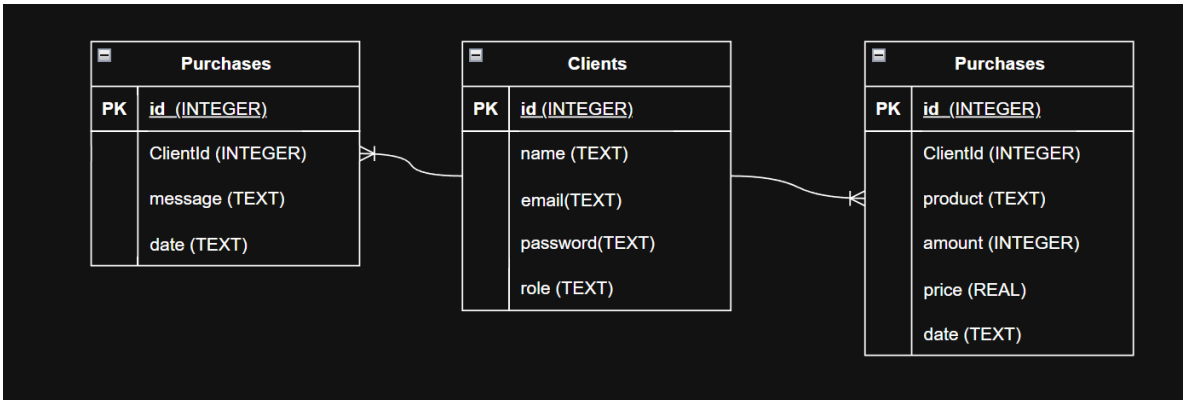


Diagrama de Componentes de API:

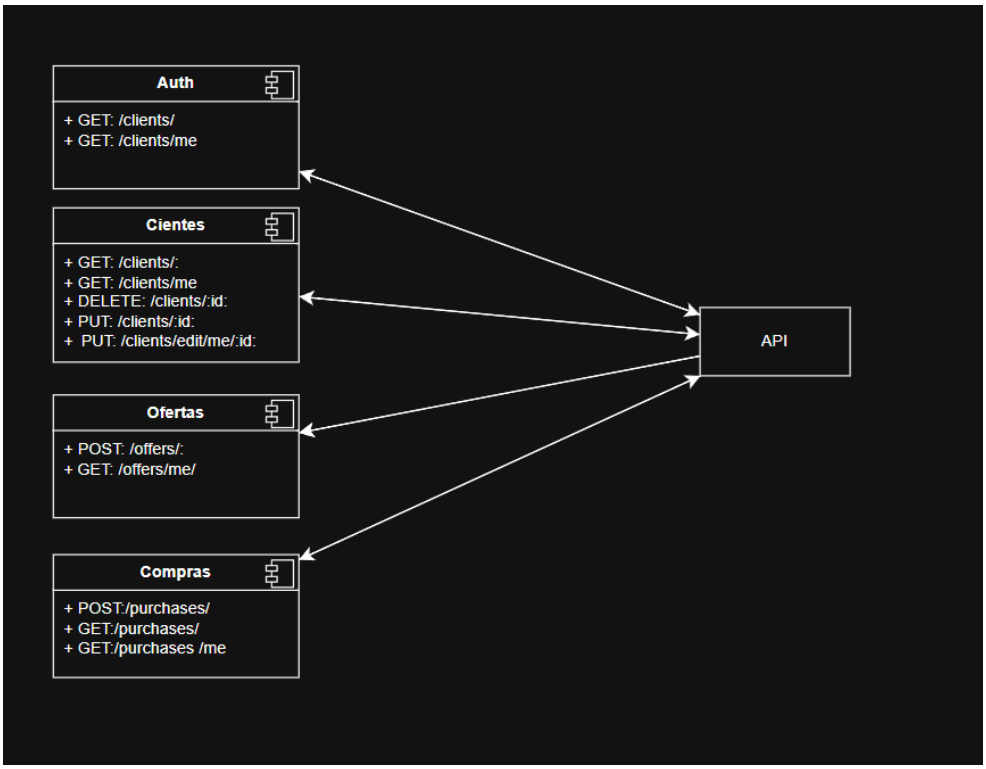
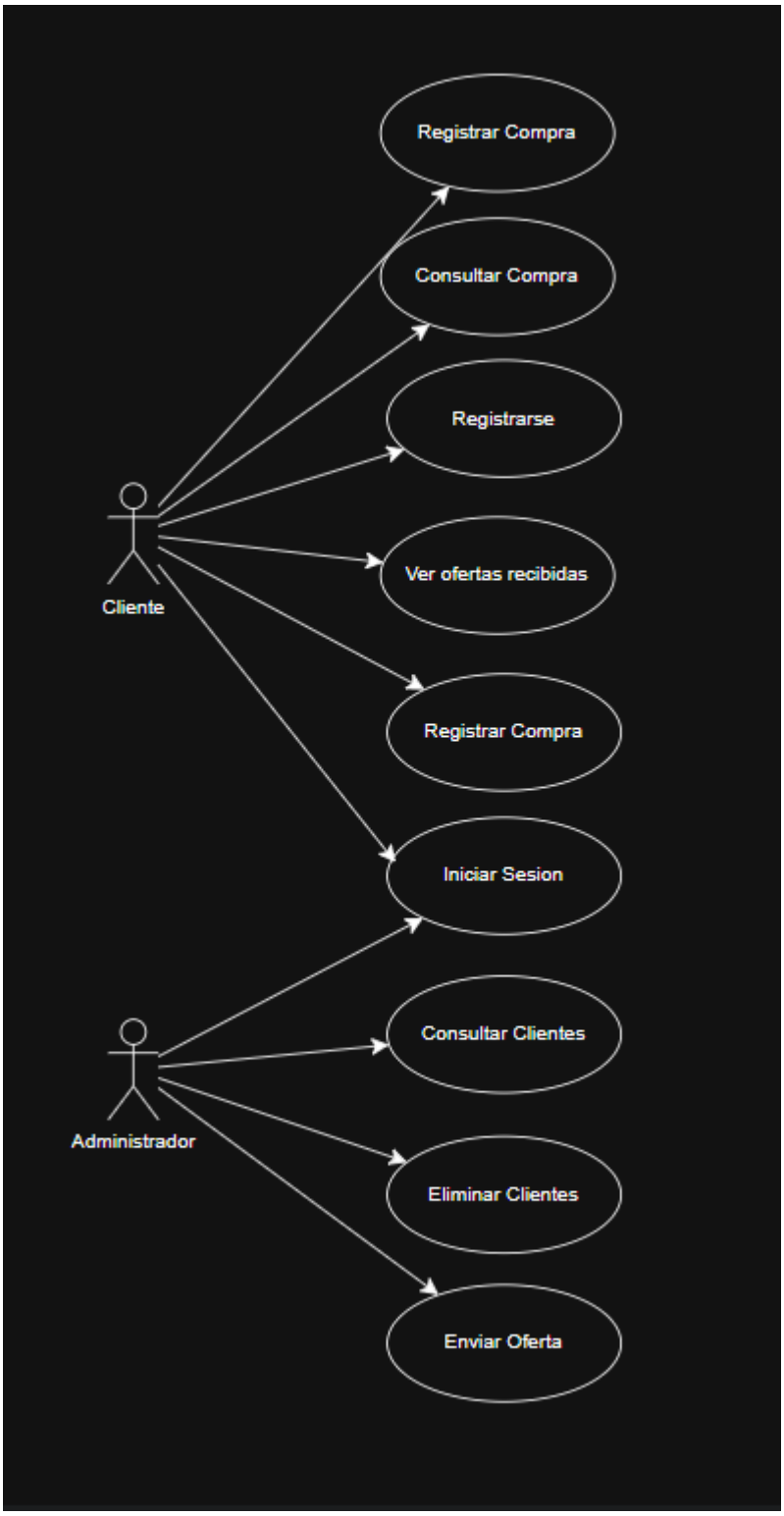


Diagrama de Casos de Uso


Registrar Compra
Consultar Compra
Registrarse
Cliente
Ver ofertas recibidas
Registrar Compra
Iniciar Sesion
Consultar Clientes
Administrador
Eliminar Clientes
Enviar Oferta

Interconexión de Componentes:

La API del backend cuenta con los siguientes endpoints:

Autenticación:

- POST:/auth/register:
  - Requiere 3 parámetros: nombre, correo y clave del usuario
  - Inserta un registro a la tabla Clientes
- POST: /auth/login
  - Requiere 2 parámetros: correo y clave del usuario
  - Realiza una selección del usuario que tenga el mismo correo que se mando en el parámetro, se le aplica un hash a la contraseña del parámetro y si coinciden las contraseñas, regresa el token, el rol del usuario y establece una sesión de 2 horas

Clientes:

- GET: /clients/:
  - Tiene restricción de Middleware para administradores solamente
  - No requiere parámetros
  - Realiza una selección del usuario y devuelve el id, nombre, email y rol del cliente
- GET: /clients/me
  - El único parámetro es el id de la persona
  - Realiza una selección del usuario que tenga el mismo id que se mando en el parámetro y devuelve todo el registro
- DELETE: /clients/:id:
  - Tiene restricción de Middleware para administradores solamente
  - Requiere 1 parámetros: id del usuario
  - Elimina el registro del cliente que tiene el id del parámetro, devuelve una respuesta json indicando que todo salió bien.
- PUT: /clients/:id:
  - Tiene restricción de Middleware para administradores solamente
  - Requiere 3 parámetros: nombre, correo e Id de usuario
  - Hace una actualización a la tabla de usuarios editando el nombre y correo del usuario que tenga el id que se manda por los parametros, estos nuevos datos que se insertan son los de los parametros. Devuelve una respuesta en json indicando que todo salió bien.
- PUT: /clients/edit/me/:id:
  - Este difiere del de arriba debido a que este no tiene restricción de middleware y sirve para que el propio usuario pueda editar sus datos.
  - Requiere 3 parámetros: nombre, correo y Id de usuario
  - Hace una actualización a la tabla de usuarios editando el nombre y correo del usuario que tenga el id que se manda por los parametros, estos nuevos datos que se insertan son los de los parametros. Devuelve una respuesta en json indicando que todo salió bien.

Ofertas:

- POST: /offers/:
  - Requiere 2 parámetros: id del cliente y mensaje a mandar
  - Inserta un registro a la tabla Ofertas el cual llega el destinatario, la fecha y el mensaje, devuelve los datos de la orden.
- GET: /offers/me/
  - Requiere 1 parámetro: el id del usuario
  - Realiza una selección de las ofertas que correspondan al mismo ID del usuario y las devuelve

Compras:

- POST:/purchases/:
  - Requiere 3 parámetros: producto, cantidad y precio del producto
  - Inserta un registro a la tabla Compras donde relaciona en el registro, el cliente, la cantidad, el precio y la fecha, devuelve la confirmación de que todo salió bien
- GET:/purchases/:
  - Tiene restricción de Middleware para administradores solamente
  - No requiere parámetros
  - Selecciona todos los registros de la tabla compras
  -
- GET:/purchases /me
  - Requiere 1 parámetro: id del usuario
  - Realiza una selección las compras del usuario que tengan el mismo id del parámetro que fue mandado.

Instalación para Desarrolladores:

Para empezar la instalación se requiere

- Una versión de Node superior a la 20.17
- NPM 10.8 o superior
- La versión mas reciente de sqlite para agosto del 2025

Para la instalación se tienen que realizar los siguientes pasos:

- Clonar el repositorio
- Abrir el repositorio local con su editor de texto favorito
- En la terminal integrada (o puede ser aparte) se corren los siguientes comandos
    - cd .\Backend\
    - npm i
    - cd ..
    - cd .\frontend\
    - npm i

Para correr el proyecto se necesitan dos terminales, una que tendrá el backend y la otra que tendrá el frontend.

Para levantar la del backend se corren los siguientes comandos en una terminal que ya este enrutada en la dirección del repositorio

- cd .\Backend\
- node server.js

Para levantar la del frontend se corren los siguientes comandos en una terminal que ya este enrutada en la dirección del repositorio

- cd .\frontend\
- npm run dev

Recomendaciones:

En base a lo que se observo mientras se desarrollaba el sistema y se realizaba el respectivo control de calidad se pueden dejar las siguientes recomendaciones para el sistema.

- A pesar de tener una base de datos SQLite, estaría mucho mejor tener un servidor de base de datos aislado del sistema propio.
- El sistema de productos debe elaborarse en base a una tabla extra debido a que no se realizo de manera coherente, con esto me refiero a que el cliente no debería de asignar precios y debería de tener sus productos ya establecidos.
- El sistema debe manejar mejor las ofertas en el sentido de poder vincularlas a una tabla de productos extra.