Campus Qro.

Materia:

Construcción de software y toma de decisiones.

Actividad:

Evidencia Entrevista STC0204
Evidencia Entrevista STC0202

Alumno:

Juan Jose Goyeneche Sánchez
A01712547

Fecha:

2025-05-27

Esta evidencia a demostrar es el **Definición de requerimientos de software,** mi trabajo a lo largo del curso, enfocandome en esta competencia en nuestro proyecto dentro del equipo BigCaesar, se ve demostrado en las siguientes actividad:

- Ayuda en requisitos Funcionales
- Requisitos No Funcionales
- Diagrama de Flujo
- Diagrama de Contexto

Esta evidencia a demostrar es el **Desarrollo de componentes de software,** mi trabajo a lo largo del curso, enfocandome en esta competencia en nuestro proyecto dentro del equipo BigCaesar, se ve demostrado en las siguientes actividad:

- BackEnd.
- Conexión con la base de datos.
- Interacción con la base de datos.
- Repositorio de GitHub.

STC0202

Ejemplo de ficha funcional:
Requisitos Funcionales.

RF-001. Registro de usuario

RF-001
Funcional - Inicial
Alta
Por aprobar

Narrativa:

una vez aceptado por el administrador.

La plataforma debe permitir a los usuarios registrarse dentro de la misma proporcionando los datos siguientes:

Nombre completo: Este será un campo por rellenar de manera obligatoria, y además solo aceptara texto alfabético, y además tendrá validación para evitar que el usuario ingrese caracteres que no son válidos.

Correo electrónico: Este será un campo por rellenar de manera obligatoria que podrá verificar en tiempo real si el correo ingresado cumple con el formato correcto (usuario@dominio.com) para de esta manera poder garantizar que se trate de un correo válido y funcional.

Contraseña: Este será un campo por rellenar de manera obligatoria. Esta contraseña deberá cumplir con los siguientes requisitos mínimos.

- Tener una longitud de al menos 8 caracteres.
- La contraseña debe tener al menos una letra mayúscula
- La contraseña debe tener al menos un número
- La contraseña debe tener al menos un caracter especial, por ejemplo:
    - @
    - #
    - $
    - %

En este campo de la contraseña también vendrá incluido un indicador visual que le muestre al usuario el nivel de seguridad con el que cuenta la contraseña que está creando el usuario.

Selección de rol: El usuario podrá hacer una petición para que se le asigne uno de los roles disponibles dentro de la plataforma

- Empresa
- Vendedor

Para completar la solicitud de rol, los usuarios deberán cargar algunos de sus documentos de identificación oficial, para de esta manera tener un mejor control y una mayor seguridad. Por ejemplo, deberá cargar:

- INE (o identificación nacional de su país)
- Documentación especifica según el rol

En caso de no tener alguno de estos dos roles, se asignará el rol de visitante.

El proceso de registro dentro de la plataforma incluirá validaciones que se harán de manera automática y en tiempo real, para así de esta manera evitar que se den errores que pueden llegar a ser comunes antes de enviar lo datos; por ejemplo, el usuario será notificado si el correo que desea registrar ya fue previamente registrado, o si la contraseña no cumple con el formato adecuado. Al terminar de llenar todo el registro de usuario, la plataforma enviará un correo electrónico al usuario con un enlace para verificar y activar la autenticidad de la dirección proporcionada. Hasta no completar este paso, el usuario no podrá acceder al sistema.

La interfaz de llenado de registro será intuitiva y accesible para todos, adaptándose a diferentes dispositivos y tamaños de pantalla. Tendrá instrucciones claras y ayudas para orientar a los usuarios en cada uno de los pasos a seguir; En caso de que el usuario cometa algún error al momento de llenar el formulario, la plataforma mostrará mensajes como los siguientes:

- “El correo no tiene un formato válido”
- “La contraseña no cumple con el formato requerido”
- “Este correo ya fue registrado previamente”

Origen:

Este requerimiento tiene su origen en la necesidad de gestionar el acceso y las funcionalidades específicas de cada rol dentro de la plataforma, dado que esta involucra diferentes funcionalidades dependiendo el rol del usuario

Dependencias

Este requerimiento depende de

- RF - 002
- RF - 003

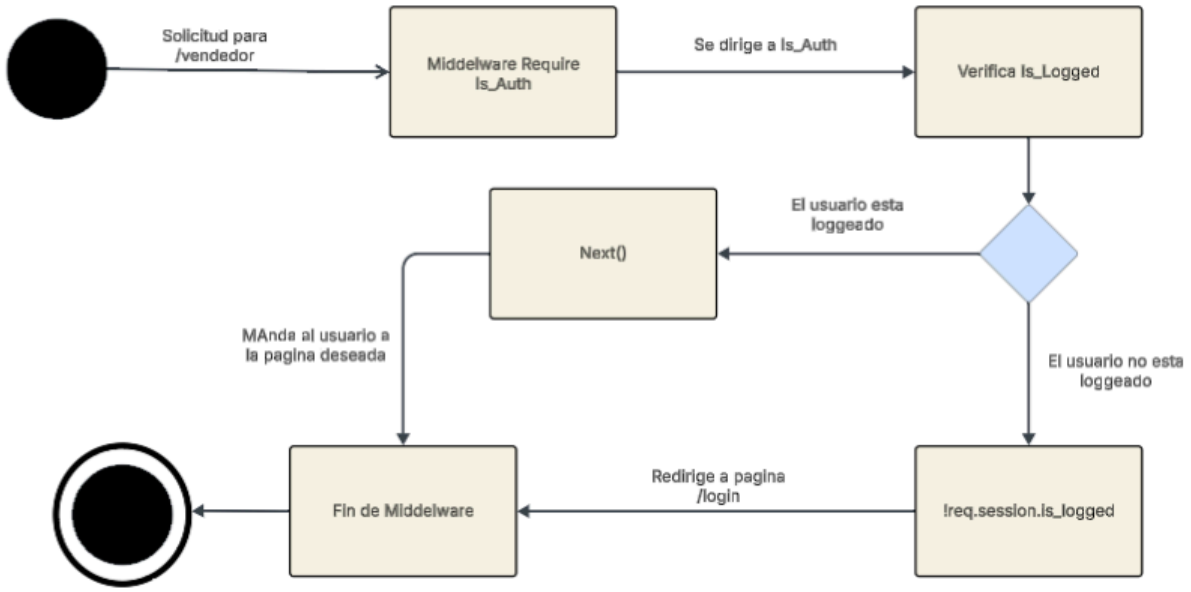
Material de referencia

- Buenas prácticas de manejo de contraseñas
- Guías de diseño de formularios de registro

Tabla de priorización de requisitos

No. de ficha	Prioridad
RF-001	Alta
RF-002	Alta
RF-003	Alta
RF-004	Alta
RF-005	Alta
RF-006	Alta
RF-007	Alta
RF-008	Alta
RF-009	Media-Baja
RF-010	Media-Baja
RF-011	Media-Baja
RNF-001	Alta
RNF-002	Alta
RNF-003	Alta
RNF-004	Alta
RNF-005	Media-Alta-
RNF-006	Media

Maquina de Estados



En los requerimientos de software a pesar de no ser mi sección de trabajo en el equipo. Ayuda a generar varios requerimientos y también ayuda a ajustar los objetivos y requerimientos funcionales en base a la practicidad del código.

STC0204

Esta evidencia se podrá ver de manera completa en el código dentro del repositorio de GitHub (no actualizado a dia actual): https://github.com/Goyeneche23/BigCaesar
De igual manera aqui hay varios ejemplos del trabajo:
Posicionamiento de archivos:

Scripts SQL para procedures:

```
CREATE DEFINER=`admin`@`%` PROCEDURE `bigcaesars`.`eliminarProductoByID`(IN
p_id_producto INT)
BEGIN
	DELETE FROM producto WHERE ID_Producto = p_id_producto;
END
```

Llamada a estos scripts desde models:

```
exports.CrearProductoConEmail = async (email, descripcion, tipo,
nombre, color) => {
    try {
      const connection = await pool.getConnection();
      const result = await connection.query(
        'CALL CrearProductoConEmail(?, ?, ?, ?, ?)',
        [email, descripcion, tipo, nombre, color]
      );
      connection.release();
      return result;
    } catch (error) {
      throw error;
    }
};
```

Usar esto deseo controller:

```
module.exports.ConsultarCantidadProductos = async (req, res) => {
    const email = req.session.email;

    if (!email) {
        console.error("No hay email en la sesión");
        return res.status(400).json({ error: "Email no proporcionado en
la sesión" });
    }

    try {
        console.log("Email en la sesión:", email);

        const cantidad = await model.ConsultarCantidadProductos(email);

        if (!cantidad || cantidad.length === 0) {
            return res.status(404).json({ error: "No se encontraron
piezas" });
        }

        return res.status(200).json({ cantidad: cantidad });

    } catch (err) {
        console.error('Error al obtener Productos:', err);
```

```
        return res.status(500).json({ error: 'Error al obtener piezas',
detalle: err.message });
    }
}
```

Manejo de Rutas para realizar tareas:

```
const express = require('express');
const router = express.Router();
const loginControllers = require('../controllers/login.controllers');
const isAuth = require("../config/is-Auth");
const pool = require("../config/database");

//const canCreate = require("../config/can-create");

router.get('/test_json',isAuth, (req, res)=>{
    res.status(200).json({code: 200, msg:"Ok"});
});

router.get('/loggear', loginControllers.render_login);

router.post('/loggear', loginControllers.do_login);

router.get('/registro', loginControllers.get_registro);

router.post('/registro', loginControllers.post_registro);

router.get('/logged', isAuth, loginControllers.get_logged);
```

Conexión con base de datos:

```
const pool = mariadb.createPool({
  host: "bigcaesars.cpemaks8ceth.us-east-2.rds.amazonaws.com",
  user: "admin",
  password: "salVAtore26#",
  connectionLimit: 5,
  database: "bigcaesars",
  port: 3306,
});
```

```
module.exports = pool;
```

Todo esto se puede ver en el github de manera completa, pero básicamente me encargue con ayuda de mis compañeros claro, de realizar el back end del proyecto el cual hace uso de todas las otras competencias para llevarlas a la realidad con un funcionamiento ya creado.