Universidad de la Cuenca del Plata

Autorización Definitiva. Decreto Poder Ejecutivo Nacional N°091/2006

Trabajo Integrador

CARRERA: Ingeniería en Sistemas de la Información.

MATERIA: Paradigmas y Lenguajes de Programación III.

COMISIÓN: “U” (única).

PROFESOR: Mgter. Encina Agustín.

ESTUDIANTE: Tabarez, Kiara.

FECHA: 18-11-2025.

TQComponents – Tienda Online de Componentes Arduino

Descripción de la Idea

El proyecto consiste en el desarrollo de una aplicación web de comercio electrónico destinada a la venta de componentes electrónicos para proyectos de Arduino.

La plataforma permitirá a los usuarios navegar por un catálogo de productos (placas, drivers, motores, sensores, cables, LEDs), agregar artículos al carrito de compras y realizar pedidos.

Asimismo, contará con un módulo administrativo para la gestión de productos, stock y pedidos.

Objetivos Generales

- Desarrollar una aplicación web full stack que implemente un modelo cliente-servidor.
- Permitir la compra y gestión de productos electrónicos de manera virtual.
- Incorporar una base de datos que garantice la persistencia de la información.
- Presentar un diseño web intuitivo, accesible y atractivo para los usuarios.

Objetivos Específicos

1. Diseñar una interfaz web responsiva con HTML, CSS y JavaScript para el frontend.
2. Implementar un backend que procese las solicitudes del cliente y gestione la comunicación con la base de datos.
3. Construir una base de datos relacional para almacenar productos, usuarios y pedidos.
4. Incluir un sistema de gestión de usuarios (clientes y administradores).
5. Incorporar funciones de búsqueda y filtrado de productos.

6. Generar la documentación técnica y los diagramas UML (casos de uso, clases, entidad-relación).

Requisitos del Proyecto

★ Requisitos Funcionales:

- RF1: El sistema deberá permitir a los usuarios registrarse e iniciar sesión.
- RF2: El usuario podrá visualizar los productos disponibles en el catálogo.
- RF3: El usuario podrá agregar y quitar productos del carrito de compras.
- RF4: El sistema permitirá confirmar un pedido mediante un formulario de compra.
- RF5: El administrador podrá dar de alta, modificar y eliminar productos.
- RF6: El administrador podrá visualizar y gestionar los pedidos realizados.

★ Requisitos No Funcionales:

- RNF1: El sistema deberá ser accesible desde navegadores web.
- RNF2: La interfaz será intuitiva y responsiva (adaptada para móviles 320px y PC 1024px).
- RNF3: La aplicación deberá garantizar la integridad de los datos almacenados.
- RNF4: El backend estará implementado en JavaScript o PHP.
- RNF5: La base de datos será MySQL Workbench.
- RNF6: El código deberá estar versionado en GitHub y documentado con README.
- RNF7: Autenticación, Las contraseñas deben tener mínimo 8 caracteres con al menos una mayúscula, una minúscula y un número.

- RNF8: Separación clara de roles: usuarios clientes y administradores. Los usuarios solo pueden acceder a sus propios pedidos y datos; los administradores tienen acceso completo al panel de gestión.

Arquitectura del Sistema

Frontend

- HTML5: Estructura semántica de las páginas web
- CSS3: Estilos y diseño responsivo
- JavaScript: Interactividad y validaciones del lado cliente

Backend

- Lenguaje: PHP
- Servidor: Apache

Base de Datos

- MySQL Workbench: Herramienta de diseño y administración

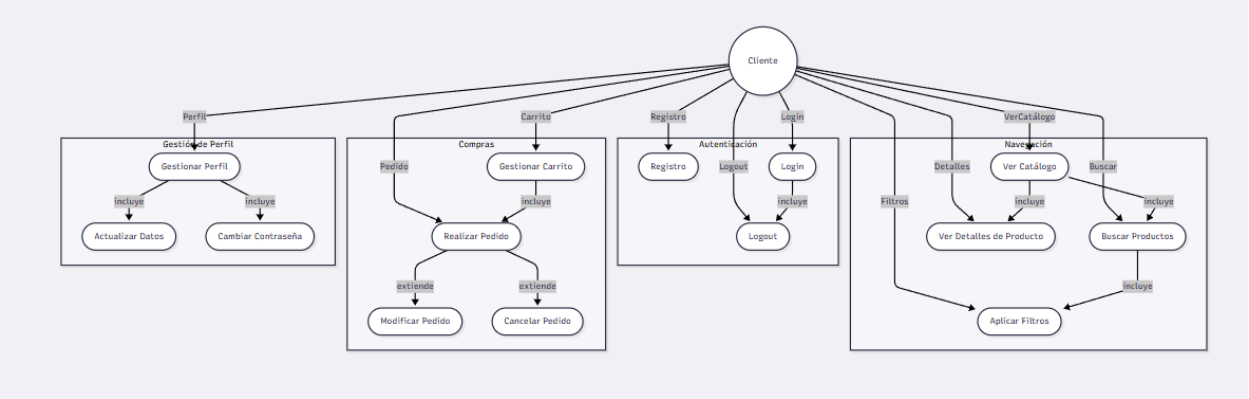
Control de Versiones

- Git: Sistema de control de versiones
- GitHub: Repositorio remoto y colaboración

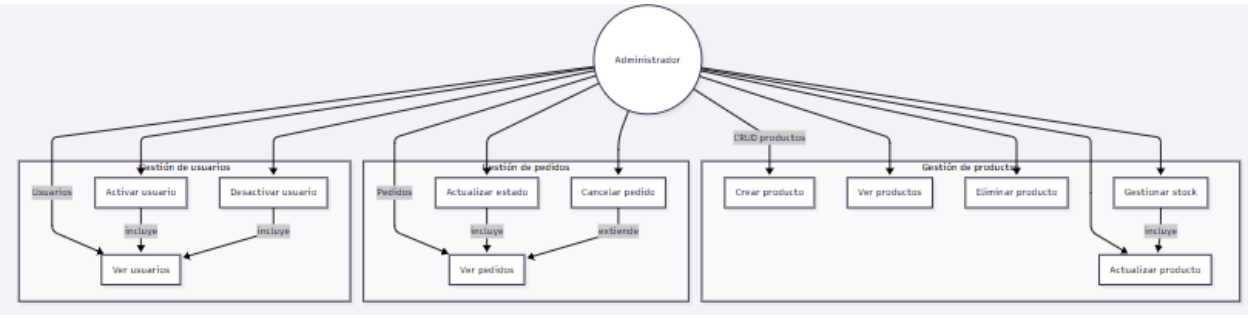
Diagramas del Sistema

Diagrama de Caso de uso

- Cliente



- Administrador



Mejoras Incorporadas mediante Js

1. Sistema de Autenticación y Gestión de Usuarios

Implementación de Registro e Inicio de Sesión Dinámico

Se desarrolló un sistema completo de autenticación del lado del cliente utilizando JavaScript vanilla y localStorage:

- **Validación en tiempo real**: El formulario de registro valida instantáneamente:
    - Disponibilidad de username (verifica duplicados)
    - Formato de email con expresiones regulares
    - Fortaleza de contraseña con indicador visual progresivo
    - Coincidencia de contraseñas

```
function validatePasswordStrength() {
        const password = regPasswordInput.value;
        let strength = 0;
        if (password.length > 5) strength++;
        if (password.match(/[a-z]/) && password.match(/[A-Z]/)) strength++;
        if (password.match(/\d/)) strength++; if (password.match(/[^a-zA-Z\d]/)) strength++;


        if (strength < 2) { text = 'Débil'; className = 'strength-weak'; }
        else if (strength < 4) { text = 'Media'; className = 'strength-medium'; } else { text = 'Fuerte';
        className = 'strength-strong';
        }
}
```

- **Persistencia de sesión**: Implementación de sistema de sesiones con expiración automática de 24 horas
- **Función "Recordarme"**: Almacenamiento seguro de credenciales en localStorage
- **Protección de rutas**: Redirección automática a login para usuarios no autenticados que intenten acceder al carrito o checkout

2. **Gestión Dinámica del Carrito de Compras**

Funcionalidades Implementadas:

a. **Agregar productos sin recargar página**
    - Sistema anti-doble click con throttling temporal
    - Actualización instantánea del contador en el header
    - Notificaciones toast no intrusivas

```
function agregarAlCarrito(id) {
  const now = Date.now();
  if (!window._ultimoAdd) window._ultimoAdd = {};
  if (window._ultimoAdd[id] && (now - window._ultimoAdd[id] < 500)) {
    return;
  }
  window._ultimoAdd[id] = now;

  const producto = productos.find(p => p.id === id);
  const productoExistente = carrito.findIndex(p => p.id === id);
```

```
    if (productoExistente !== -1) {
        carrito[productoExistente].cantidad++;
    } else {
        carrito.push({...producto, cantidad: 1});
    }

    guardarCarrito();
    mostrarNotificacion(`"${producto.nombre}" agregado al carrito`);
}
```

b. Modificación de cantidades en tiempo real

- Botones de incremento/decremento
- Input manual con validación
- Recálculo automático de subtotales y total

c. Eliminación individual o masiva

- Modal de confirmación para vaciar carrito
- Actualización instantánea de la interfaz

d. Cálculo dinámico de costos

- Subtotal automático
- Envío gratis para compras > $5000
- Total general actualizado en tiempo real

3. Sistema de Filtrado y Navegación de Productos

Características Implementadas:

a. **Filtrado por categorías**

```
function obtenerCategoriaDeURL() {
    const urlParams = new URLSearchParams(window.location.search);
    return urlParams.get('categoria');
}
function cargarProductosPorCategoria(categoria) {
    let productosFiltrados = [...productos];


    if (categoria && categoria !== 'todos') {
        productosFiltrados = productosFiltrados.filter(
            producto => producto.categoria === categoria
        );
    }
    renderizarProductos(productosFiltrados);
}
```

b. **Dos vistas de productos**:

- **Vista Grid**: Tarjetas con imagen, descripción y precio
- **Vista Tabla**: Listado con filtros y ordenamiento alfabético

c. **Menú dropdown responsivo**

- Navegación por categorías
- Adaptación automática a dispositivo móvil (hamburguesa)

4. Proceso de Checkout Interactivo

Validaciones Implementadas en Tiempo Real:

a. **Información Personal**

- Validación de formato de email: /^[^\s@]+@[^\s@]+\.[^\s@]+$/
- Validación de teléfono: /^[+]?[(]?[0-9]{1,4}[)]?[-\s./0-9]*$/

b. **Información de Tarjeta**

- Formateo automático del número (XXXX-XXXX-XXXX-XXXX)
- Validación de fecha de expiración con verificación contra fecha actual
- CVV de 3-4 dígitos

```
numeroTarjeta.addEventListener('input', function(e) {
    let value = e.target.value.replace(/\s+/g, '').replace(/[^0-9]/gi, '');
    let formattedValue = '';

    for (let i = 0; i < value.length; i++) {
        if (i > 0 && i % 4 === 0) {
            formattedValue += ' ';
        }
        formattedValue += value[i];
    }

    e.target.value = formattedValue;
});
```

c. **Mostrar/Ocultar campos según método de pago**

- Campos de tarjeta aparecen solo si se selecciona "Tarjeta" o "Débito"
- Transiciones con CSS

d. **Resumen dinámico del pedido**

- Actualización en tiempo real de productos, cantidades y precios

- Cálculo automático de subtotal, envío y total

5 Sistema de Notificaciones y Feedback Visual

Implementaciones:

a. Notificaciones Toast

```
}
  function mostrarNotificacion(mensaje, esError = false) {
  const notificacion = document.createElement('div');
  notificacion.className = `notificacion ${esError ? 'error' : ''}`;
  notificacion.textContent = mensaje;

  document.body.appendChild(notificacion);
    setTimeout(() => {
    notificacion.classList.add('mostrar');
  }, 10);

  setTimeout(() => {
    notificacion.classList.remove('mostrar');
    setTimeout(() => notificacion.remove(), 300);
  }, 3000);
}
```

b. **Modales de Confirmación**

i. Modal para vaciar carrito

ii. Modal de éxito al finalizar compra

iii. Animaciones de entrada/salida

c. **Indicadores Visuales**

i. Contador de carrito siempre visible

ii. Estados hover en botones y productos

iii. Transiciones suaves en todos los elementos interactivos

6 Gestión de Estado y Persistencia

LocalStorage como Sistema de Persistencia:

a. **Datos almacenados:**

- users: Array de usuarios registrados
- carrito: Productos en el carrito
- isLoggedIn, username, loginTime: Estado de sesión
- historialCompras: Registro completo de pedidos
- rememberedUsername/Password: Credenciales guardadas

b. **Sincronización en tiempo real**

- Actualización del localStorage en cada acción
- Lectura automática al cargar páginas
- Limpieza de datos obsoletos (sesiones expiradas)

7 Adaptación Responsive con JavaScript

Funcionalidades Móviles:

a. Menú Hamburguesa

```
const menuToggle = document.querySelector(".menu-toggle");
const navMenu = document.querySelector("nav ul");

if (menuToggle && navMenu) {
  menuToggle.addEventListener("click", () => {
    navMenu.classList.toggle("active");
  });
}
```

b. Detección de contexto

    a. Botón "Volver" visible solo en páginas internas

    b. Ajuste de dropdowns en dispositivos táctiles

c. Optimización de tablas en móviles

    a. Conversión automática de tablas a cards mediante CSS

    b. Atributos data-label para etiquetas responsivas

8 Funciones Auxiliares y Optimizaciones

a. Normalización de rutas de imágenes

```
productosFiltrados = productosFiltrados.map(p => {
  let imagenLimpia = p.imagen
    .normalize("NFD")
    .replace(/[\u0300-\u036f]/g, "")
    .replace(/\s+/g, "")
    .replace(/[°°]/g, "");
```

```
  return { ...p, imagen: imagenLimpia };
});
```

b. Gestión de errores de carga de imágenes

```
<img src="${producto.imagen}"
  alt="${producto.nombre}"
  onerror="this.src='imagenes/iconos/no-image.png'">
```

3. Limpieza automática de parámetros URL
    - Eliminación de parámetros después de login exitoso

Link del Repositorio GitHub

Repositorio del Proyecto:

https://github.com/Kiaratabarez/Kiaratabarez-TQcomponentes.git

Link del Proyecto en Línea (Hosting)

URL del Sitio Web:

https://tqcomponents.netlify.app/

Plataforma de Hosting: Netlify

Implementación CSS Responsivo

5.1 Breakpoints Definidos

Dispositivo	Ancho	Características
Móvil pequeño	< 480px	1 columna, menú hamburguesa
Móvil grande	481px - 768px	2 columnas en productos
Tablet	769px - 1024px	2-3 columnas, menú horizontal
Desktop	> 1025px	3-4 columnas, layout completo

5.2 Técnicas Responsive Aplicadas

1. CSS Grid Adaptativo

```
.productos-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 25px;
}
```

5.3 Flexbox para Layouts Dinámicos

```
.header-container {
    display: flex;
```

```
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap;
}
```

5.4 Unidades Relativas

- Uso de rem y em para tipografía
- vh y vw para secciones hero
- Porcentajes para anchos de contenedores

5.5 Imágenes Responsivas

```
.producto img {
  max-width: 100%;
  height: auto;
  object-fit: contain;
}
```

5.6 Componentes Adaptativos

Navegación Responsiva:

- Desktop: Menú horizontal completo
- Móvil: Menú hamburguesa con animación
- Tablas Adaptativas:

```
@media (max-width: 600px) {
  .tabla-productos thead {
    display: none;
  }

  .tabla-productos tr {
    display: block;
    margin-bottom: 15px;
    border: 1px solid #ddd;
  }

  .tabla-productos td {
    display: flex;
    justify-content: space-between;
  }

  .tabla-productos td::before {
    content: attr(data-label);
    font-weight: bold;
  }
```

- Formularios Responsivos:

```
.form-group.doble {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 15px;
}
```

```
@media (max-width: 768px) {
    .form-group.doble {
        grid-template-columns: 1fr;
    }
}
```

- Variables CSS para Consistencia

```
:root {
    --color-primary: #2c3e50;
    --color-secondary: #e74c3c;
    --color-accent: #3498db;
    --shadow: 0 3px 10px rgba(0,0,0,0.1);
    --transition: all 0.3s ease;
}
```

Animaciones y Transiciones

- Transiciones Suaves:

```
.producto {
    transition: var(--transition);
}

.producto:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 20px rgba(0,0,0,0.15);
```

```
}
```

Animaciones de Entrada:

```
@keyframes fadeIn {
   from {
      opacity: 0;
      transform: translateY(-20px);
   }
   to {
      opacity: 1;
      transform: translateY(0);
   }
}

.auth-card {
   animation: fadeIn 0.4s ease;
}
```

Tecnologías Utilizadas

Categoría	Tecnología
Frontend	HTML5, CSS3, JavaScript
Estilos	CSS Grid, Flexbox, Media Queries
Fuentes	Google Fonts (Poppins)
Hosting	Netlify
Control de Versiones	Git / GitHub
Persistencia	LocalStorage

Base de Datos

El proyecto utiliza una base de datos relacional para gestionar la información de manera persistente y consistente.

A. Tecnología y Diseño Conceptual

Característica	Detalle	Fuente en el Informe
Motor DB	MySQL	Requisito No Funcional (RNF5)
Herramienta	MySQL Workbench	RNF5 / Arquitectura del Sistema
Objetivo	Construir una base de datos relacional para almacenar productos, usuarios y pedidos.	Objetivo Específico 3
Documentación	Generar el diagrama Entidad-Relación (UML).	Objetivo Específico 6

B. Modelo Entidad-Relación Conceptual

El diseño conceptual se basa en la necesidad de gestionar tres entidades principales y sus relaciones transaccionales:

1. **Entidad Usuario:** Almacena la información de acceso y perfil, implementando la **separación clara de roles** (clientes y administradores).
    - **Campos Clave:** id_usuario, nombre, email, password (encriptada, debe cumplir la política de fortaleza de RNF7: 8 caracteres, mayúscula, minúscula y número).
    - **Atributo de Control:** rol (define permisos de acceso y gestión, RNF8).
2. **Entidad Producto:** Representa los componentes de Arduino disponibles en el catálogo.
    - **Campos Clave:** id_producto, nombre, descripcion, precio, stock (esencial para la gestión de stock), categoria.

3. **Entidad Pedido:** Registra la orden de compra confirmada por el cliente (RF4)[13].
    - **Relación:** Vinculado con Usuario (1:N), ya que un usuario puede tener múltiples pedidos.
    - **Campos Clave:** id_pedido, id_usuario (FK), fecha, total, estado (para seguimiento y gestión administrativa).
4. **Relación Detalle_Pedido:** Actúa como una tabla de unión (N:M) entre Pedido y Producto, registrando las cantidades de cada artículo vendido.

Backend PHP

El backend está diseñado para operar bajo un modelo cliente-servidor utilizando **PHP** y **Apache** , siendo responsable de procesar la lógica de negocio y las transacciones con la base de datos.

A. Funciones Clave del Servidor

La capa PHP debe gestionar todas las solicitudes que requieren persistencia de datos o acceso privilegiado:

1. **Módulo de Autenticación (auth.php):**
    - Gestiona el registro e inicio de sesión.
    - Aplica la validación de contraseñas (RNF7) y verifica el rol del usuario.
2. **Módulo de Catálogo (products.php):**
    - Responde a solicitudes GET para obtener el catálogo de productos disponible (RF2).
    - Ejecuta las funciones de búsqueda y filtrado de productos.
3. **Módulo Transaccional (checkout.php):**
    - Recibe los datos del formulario de compra y del carrito desde el frontend (RF4).
    - Crea un registro en la tabla Pedidos y actualiza el stock en la tabla Productos.

Implementación de la Gestión del Administrador

El sistema requiere un panel administrativo para controlar el catálogo y los pedidos (RF5 y RF6) , con acceso restringido (RNF8).

A. Funcionalidades del Administrador

Los archivos PHP del backend deben implementar la lógica para los casos de uso definidos en el diagrama:

Módulo de Gestión	Funciones (Casos de Uso)	Requisito Funcional (RF)	Impacto en la Base de Datos
Gestión de Productos	Crear, Modificar y Eliminar productos.	RF5	CRUD sobre la tabla Productos.
Gestión de Stock	Actualizar stock de los productos.	RF5 (Implícito)	UPDATE sobre el campo stock en Productos.
Gestión de Pedidos	Visualizar y Actualizar el estado de los pedidos.	RF6	LECTURA de la tabla Pedidos y UPDATE del campo estado.
Gestión de Usuarios	Ver, Activar/Desactivar y Eliminar usuarios.	RNF8 (Control de Roles)	CRUD sobre la tabla Usuarios.

B. Interfaz del Administrador

- **Protección de Rutas:** JavaScript del lado del cliente y PHP del lado del servidor garantizarían que solo los usuarios con el rol Administrador puedan acceder a estas vistas (RNF8).
- **Manejo de Datos:** El panel utilizaría solicitudes **asíncronas (AJAX)** para comunicarse con los *scripts PHP* especializados (admin_products.php, admin_orders.php), actualizando la interfaz dinámicamente sin recarga.
- **Componentes Adaptativos:** Las tablas de gestión (productos, pedidos) emplearían técnicas CSS Responsive para asegurar la accesibilidad, como la conversión de tablas a *cards* en dispositivos móviles.

Referencias

- MDN Web Docs - JavaScript: https://developer.mozilla.org/es/docs/Web/JavaScript
- CSS Tricks - Complete Guide to Grid: https://css-tricks.com/snippets/css/complete-guide-grid/
- W3C - Web Storage API: https://www.w3.org/TR/webstorage/
- Netlify Documentation: https://docs.netlify.com/