***Pérez Lautaro***

***Juarez ian***

***“Exequiel pacheco”***

***6to C***

***LABORATORIA DE PROGRAMACION III***

**ACTIVDAD:**

Kiosco Automatizado con Mercado Pago

Fase 1 – Investigación y Análisis (Semana 1 y 2)

| Tarea | Description | Responsible(s) | Plaza |
| --- | --- | --- | --- |
| Estudio de kioscos y máquinas expendedoras | Analizar modelos internacionales, características y funcionalidades. | Equip de investigation | dia 1 – dia 5 |
| Identification de components | Listar hardware y software más comunes (placas, sensores, motores, etc.). | Ticino de hardware + programador | Día 3 – Día 8 |
| Investigación de integración de pagos | Analizar API de Mercado Pago, protocolos de seguridad y ejemplos. | Programador | Día 5 – Día 10 |

***Resuelto:***

1 – Análisis del funcionamiento de kioscos automatizados y máquinas expendedoras en el mercado internacional  
Los kioscos automatizados y las máquinas expendedoras en el mercado internacional funcionan como sistemas integrados que combinan hardware y software para ofrecer productos o servicios sin la intervención directa de un operador humano.  
En general, su funcionamiento sigue el siguiente flujo:

1. Interfaz de usuario: pantalla táctil o botones físicos para seleccionar el producto/servicio.
2. Sistema de pago: admite monedas, billetes, tarjetas, códigos QR o pagos electrónicos.
3. Unidad de control: microcontrolador o mini-PC que gestiona la lógica de venta, el control de inventario y la comunicación con el sistema de pago.
4. Mecanismo de dispensado: motores, bandas transportadoras, solenoides u otros actuadores que liberan el producto.
5. Señalización: luces LED, sonidos o mensajes en pantalla para indicar estados como “Pago aceptado”, “Producto entregado” o “Error”.  
   A nivel global, se observan mejoras en rapidez de operación, compatibilidad con múltiples métodos de pago, sensores de stock, y conectividad IoT para gestión remota.

2 – Componentes de hardware y software más utilizados

Hardware más común:

* Controlador principal: Raspberry Pi, Arduino, ESP32 o PLC industriales.
* Pantalla: LCD táctil o TFT para interfaz.
* Mecanismos de dispensado: motores paso a paso, servomotores o solenoides.
* Sensores: finales de carrera, sensores ópticos para confirmar entrega, sensores de stock.
* Sistemas de pago: lectores de tarjetas, lectores de QR, módulos NFC, validadores de monedas/billetes.
* Conectividad: módulos Wi-Fi, Ethernet o 4G.
* Fuente de alimentación: adaptadores de corriente y fuentes reguladas.
* Estructura física: carcasa metálica o de policarbonato para seguridad.

Software más común:

* Firmware para controladores embebidos (C/C++ o MicroPython).
* Sistemas operativos embebidos (Raspberry Pi OS, Android Things, Linux embebido).
* Aplicaciones de control escritas en Python, Java, Node.js o C#.
* APIs de pago (REST/HTTP) para conexión con pasarelas como Mercado Pago, PayPal o Stripe.
* Bases de datos locales o en la nube para inventario y transacciones.

3 – Métodos de integración de pagos electrónicos y seguridad  
Los sistemas modernos se integran con pasarelas de pago electrónico a través de APIs seguras.

* Proceso de integración típico:
  1. El sistema genera una orden de pago a través de la API.
  2. El cliente paga usando QR, link de pago o NFC.
  3. La pasarela confirma el pago vía webhook o consulta periódica.
  4. El sistema habilita el dispensado al recibir confirmación.
* Métodos de pago soportados: tarjetas de crédito/débito, billeteras virtuales (Mercado Pago, PayPal), transferencias instantáneas y pagos por código QR.
* Medidas de seguridad:
  1. HTTPS/TLS para cifrado de la comunicación.
  2. Tokens de autenticación (OAuth2) para acceder a la API.
  3. Validación de firma digital en notificaciones de pago.
  4. Protección contra duplicados verificando ID de transacción.
  5. Aislamiento de control para que el mecanismo de entrega solo se active tras confirmación autenticada.

Fase 2 – Diseño del Sistema (Semana 3)

| Tarea | Descripción | Responsable(s) | Plazo |
| --- | --- | --- | --- |
| Definición de arquitectura | Elegir esquema general (hardware + software + conexión a internet). | Todo el equipo | Día 1 – Día 2 |
| Diagramas eléctricos y mecánicos | Crear planos de conexión y ensamblaje. | Técnico de hardware | Día 2 – Día 4 |
| Lista de materiales y herramientas | Adaptar diseño al stock disponible en laboratorio. | Responsable de materiales | Día 4 – Día 5 |

**Implementación del Sistema Físico**

Se desarrolló un prototipo de kiosco automatizado capaz de recibir pagos a través de la pasarela **Mercado Pago** y activar un mecanismo de habilitación/dispensado una vez acreditada la transacción.  
El sistema consta de los siguientes módulos:

1. **Módulo de pago**
   * Conexión a la API de Mercado Pago para generar órdenes y recibir confirmaciones.
   * Uso de un enlace de pago o código QR dinámico para que el cliente realice la transacción desde su dispositivo móvil.
2. **Unidad de control**
   * Microcontrolador ESP32 (o placa Raspberry Pi) encargado de manejar la lógica del sistema.
   * Recepción de la confirmación de pago mediante comunicación segura HTTPS.
3. **Mecanismo de habilitación/dispensado**
   * Control de un motor paso a paso o servomotor para liberar el producto.
   * Señalización con LED y/o buzzer para indicar estados (“Esperando pago”, “Pago aprobado”, “Producto entregado”).

**Adaptación del Diseño y Componentes al Stock del Laboratorio**

Se revisó el inventario del laboratorio para minimizar costos y tiempos:

* **Controlador**: reutilización de una placa ESP32 disponible.
* **Pantalla**: módulo LCD 16x2 reutilizado para mensajes simples en lugar de una pantalla táctil.
* **Mecanismo**: servomotor estándar recuperado de un proyecto anterior para accionar una compuerta de dispensado.
* **Fuente de alimentación**: adaptador de 12V reciclado, con conversor a 5V para electrónica de control.
* **Estructura**: fabricación en MDF y acrílico cortado en láser, materiales disponibles en el taller.

**Documentación del Proceso**

Durante todo el desarrollo se registró:

* **Diseño**: diagramas eléctricos, esquema mecánico y flujo de datos del sistema
* **Programación**: código documentado en GitHub con comentarios, dividido en módulos (conexión API, control mecánico, interfaz de usuario).
* **Ensamblaje**: fotografías paso a paso del armado físico.
* **Pruebas**: registro de test de pago, tiempos de respuesta y ajustes realizados.
* **Problemas y soluciones**:
  + *Problema*: retraso en la confirmación de pago.  
    *Solución*: optimización de consultas a la API y uso de webhooks.
  + *Problema*: interferencia en señal del servomotor.  
    *Solución*: filtrado y aislamiento eléctrico de la alimentacióN

Fase 3 – Desarrollo y Ensamblaje (Semana 4 y 5)

| Tarea | Description | Responsible(s) | Plaza |
| --- | --- | --- | --- |
| Montage fisIco del prototipo | Ensamblar estructura, componentes electrónicos y mecánicos. | Técnico de hardware | Día 1 – Día 5 |
| Programación de control | Desarrollar código para conexión con API de Mercado Pago y control de mecanismo. | Programador | Día 3 – Día 9 |
| Integración hardware-software | Probar comunicación entre pago y mecanismo de entrega. | Todo el equipo | Día 6 – Día 10 |

Fase 4 – Pruebas y Ajustes (Semana 6)

| Tarea | Descripción | Responsable(s) | Plazo |
| --- | --- | --- | --- |
| Pruebas de pago real | Simular transacciones en entorno de prueba de Mercado Pago. | Programador | Día 1 – Día 2 |
| Corrección de fallos | Resolver errores de conexión, tiempo de respuesta o mecánica. | Todo el equipo | Día 2 – Día 4 |
| Mejoras en interfaz y señalización | Ajustar luces, sonido o mensajes de estado. | Programador + diseñador | Día 4 – Día 5 |

Fase 5 – Documentación y Presentación (Semana 7)

| Tarea | Descripción | Responsable(s) | Plazo |
| --- | --- | --- | --- |
| Carpeta de campo digital | Redactar introducción, objetivos, investigación, diagramas, fotos y registro de problemas. | Responsable de documentación | Día 1 – Día 3 |
| Repositorio en GitHub | Subir código, diagramas y README.md. | Programador | Día 2 – Día 3 |
| Presentación en Canva | Preparar presentación visual para Expotécnica. | Diseñador | Día 3 – Día 4 |
| Ensayo de presentación | Practicar exposición con prototipo. | Todo el equipo | Día 5 |

Fase 6 – Presentación en Expotécnica (Semana 8)

| Tarea | Descripción | Responsable(s) | Plazo |
| --- | --- | --- | --- |
| Montaje en stand | Instalar prototipo y material visual. | Todo el equipo | Día del evento |
| Exposición | Explicar funcionamiento y responder preguntas. | Todo el equipo | Durante el evento |

Conexión código a mercado pago:

<?php

require \_\_DIR\_\_ . '/vendor/autoload.php';

use MercadoPago\MercadoPagoConfig;

use MercadoPago\Client\Preference\PreferenceClient;

// Configura tus credenciales

MercadoPagoConfig::setAccessToken('APP\_USR-8733978112569923-062512-93f26bd550a0cc81991b50b51fd786fc-1577049737');

// Crea una preferencia

$client = new PreferenceClient();

$preference = $client->create([

    "items" => [

        [

            "title" => "Mi producto",

            "quantity" => 1,

            "unit\_price" => 2000

        ]

    ],

    "back\_urls" => [

        "success" => "http://localhost/integramp/in.php",

        "failure" => "https://www.tu-sitio.com/failure",

        "pending" => "https://www.tu-sitio.com/pending"

    ],

    "auto\_return" => "approved"

]);

$preferenceId = $preference->id;

?>

<!DOCTYPE html>

<html>

<head>

  <title>Mi Integración con Checkout Pro</title>

</head>

<body>

  <h1>Botón de pago</h1>

  <div id="walletBrick\_container"></div>

  <!-- SDK de Mercado Pago -->

  <script src="https://sdk.mercadopago.com/js/v2"></script>

  <script>

    const mp = new MercadoPago('APP\_USR-d0d1b2d6-a42c-4504-8218-b2b052391fbb');

    const renderWalletBrick = async (bricksBuilder) => {

      await bricksBuilder.create("wallet", "walletBrick\_container", {

        initialization: {

          preferenceId: "<?php echo $preferenceId; ?>",

        }

      });

    };

    const bricksBuilder = mp.bricks();

    renderWalletBrick(bricksBuilder);

  </script>

</body>

</html>

***Accsess token***: APP\_USR-2298341657679130-082712-fa5b57bb0a7d9800b70bc3a6bf4d37c7-1577049737

***Public key***: APP\_USR-c4be6073-8d1c-4750-9e54-fa67e193e1fa

Autorizacion para el accsess token: http.addHeader("Authorization", "Bearer TU\_ACCESS\_TOKEN");

***Código de la maquina exprendedora:***

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Configuración de la pantalla LCD (dirección 0x27)

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Pinos de los botones para seleccionar productos

const int botonA = 2; // Producto A

const int botonB = 3; // Producto B

const int botonC = 4; // Producto C

// Pino del motor de paso o servomotor (depende del tipo de motor)

const int pinMotor = 9; // Puedo usar un motor de paso o servomotor

// Variables para la cantidad de monedas y el saldo

int monedas = 0;

int saldo = 0;

const int costoProductoA = 5; // Precio del Producto A

const int costoProductoB = 3; // Precio del Producto B

const int costoProductoC = 2; // Precio del Producto C

void setup() {

// Inicialización de la pantalla LCD

lcd.begin(16, 2);

lcd.print("Maquina Expendedora");

// Configuración de los pines de los botones

pinMode(botonA, INPUT\_PULLUP);

pinMode(botonB, INPUT\_PULLUP);

pinMode(botonC, INPUT\_PULLUP);

// Inicializar el motor

pinMode(pinMotor, OUTPUT);

// Mostrar el saldo inicial

lcd.setCursor(0, 1);

lcd.print("Saldo: $");

lcd.print(saldo);

}

void loop() {

// Comprobamos si se ha insertado una moneda

if (digitalRead(5) == LOW) { // Supón que el pin 5 es un sensor de monedas

monedas++;

saldo += 1; // Asumimos que cada moneda tiene valor de 1

delay(500); // Debemos dar tiempo para que el sensor se estabilice

// Actualizar la pantalla LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Insertando moneda");

lcd.setCursor(0, 1);

lcd.print("Saldo: $");

lcd.print(saldo);

delay(1000); // Mostrar por un momento el saldo

lcd.clear();

lcd.print("Maquina Expendedora");

}

// Verificar si el usuario presiona los botones para seleccionar productos

if (digitalRead(botonA) == LOW) {

if (saldo >= costoProductoA) {

// Despachar producto A

entregarProducto("Producto A");

saldo -= costoProductoA;

actualizarSaldo();

} else {

mostrarError("No suficiente saldo");

}

}

if (digitalRead(botonB) == LOW) {

if (saldo >= costoProductoB) {

// Despachar producto B

entregarProducto("Producto B");

saldo -= costoProductoB;

actualizarSaldo();

} else {

mostrarError("No suficiente saldo");

}

}

if (digitalRead(botonC) == LOW) {

if (saldo >= costoProductoC) {

// Despachar producto C

entregarProducto("Producto C");

saldo -= costoProductoC;

actualizarSaldo();

} else {

mostrarError("No suficiente saldo");

}

}

// Mantener la pantalla actualizada

delay(100);

}

// Función para entregar el producto

void entregarProducto(String producto) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Dispensando");

lcd.setCursor(0, 1);

lcd.print(producto);

// Aquí, activa el motor para dispensar el producto.

// Suponiendo que tienes un servomotor o motor de paso que gire

digitalWrite(pinMotor, HIGH);

delay(2000); // El motor estará encendido durante 2 segundos

digitalWrite(pinMotor, LOW);

delay(500); // Tiempo para que el producto salga

}

// Función para actualizar el saldo en la pantalla LCD

void actualizarSaldo() {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Saldo: $");

lcd.print(saldo);

}