Departamento de Ciencias de la Computación (DCCO)

Carrera de Ingeniería Electrónica y Automatización

Fundamentos de Programación

Perfil del Proyecto v5.

Presentado por: (Grupo 1)

- Alvear Alexander
- Campoverde Anthony
- Velecela Mateo

Tutor académico: Ing. Jenny A Ruiz R

Ciudad: Quito- Ecuador Fecha:

11/07/2025

Contenido

1. Introducción

El presente proyecto nace de la necesidad real de un pequeño negocio de víveres y frutos secos que carece de un sistema digital para la gestión de su inventario y proveedores, ya que actualmente, el control se realiza de forma manual, lo que inconscientemente genera errores y pérdidas de información, se propone el desarrollo de una herramienta simple, ejecutada desde consola y desarrollada en C, que permita organizar y monitorear productos y proveedores, contribuyendo a una mejor toma de decisiones y reposición eficiente del stock.

2. Planteamiento del Trabajo

2.1. Formulación del Problema

La gestión manual del inventario y proveedores dificulta el control de productos agotados, la planificación de compras y el seguimiento a los proveedores, el proyecto propone desarrollar una aplicación de consola en C que permita registrar, visualizar y mantener actualizado el inventario de productos y proveedores, ayudando a la dueña del negocio a tomar decisiones más acertadas basándose en algo de lo que tiene un registro sin inconsistencias.

2.2. Justificación

Este proyecto puede servir de modelo para muchos pequeños negocios que ya sea por, limitaciones económicas o técnicas, no pueden acceder a sistemas complejos de gestión, el implementarlo en C permite el aprendizaje práctico de estructuras de datos y lógica de programación, aportando tanto a nivel académico como profesional.

3. Sistema de Objetivos

3.1. Objetivo General

Desarrollar una aplicación de consola en lenguaje C que se ejecute en computadora, la cual permita gestionar de manera eficiente productos y proveedores para un negocio local de víveres y frutos secos, mediante el uso de estructuras de datos y archivos para el manejo interno del sistema, facilitando el control del inventario y el abastecimiento continuo, con el fin de optimizar los procesos de registro, consulta, modificación y seguimiento del stock disponible.

3.2. Objetivos Específicos

- Realizar pruebas de caja blanca (revisión lógica del código), verificando el correcto flujo de ejecución del programa, el funcionamiento de las estructuras condicionales y repetitivas, así como la integridad de los datos procesados

- Realizar pruebas de caja negra (verificación de entradas y salidas), evaluando el comportamiento del sistema ante diferentes datos ingresados por el usuario y verificando que las salidas sean correctas y coherentes con los requerimientos

- Desarrollar una funcionalidad que permita generar alertas automáticas cuando el stock de un producto sea igual o inferior al nivel mínimo definido, con el fin de facilitar una reposición oportuna y evitar la falta de productos en el micro emprendimiento.

- **Paquetes de trabajo:**

Análisis de requerimientos: Reunirse con el cliente, identificar su necesidad, ya sean productos, proveedores, alertas, ventas, etc. Redactar los requerimientos funcionales.

Diseño del Sistema: Definir la estructura de datos, es decir, una estructura funcional para el almacenamiento y archivado de datos de productos, y proveedores, planificar el flujo del programa (menús, operaciones dentro del mismo).

Implementación del Código: Codificar el sistema en C, funcionalidades para almacenamiento de datos para productos y proveedores, lógica de alertas de Stock Bajo.

Pruebas y Validación: Probar el sistema con diferentes casos, (stock bajo, producto inexistente, etc.), Verificar que los datos se gestionen correctamente, Ajustar errores que se puedan encontrar.

Documentación y entrega: Redacción de informe de proyecto, Documentación de cómo usar el sistema, errores, etc.

4. Alcance

El sistema permitirá registrar productos y proveedores, consultar y modificar sus datos, mostrar alertas de stock bajo, y realizar búsquedas por nombre o categoría. Este mismo no incluye interfaz gráfica ni integración con bases de datos externas, ya que se ejecuta únicamente en consola.

Específicamente, se desarrollarán las siguientes funcionalidades:

- Implementar un menú principal con opciones claras para gestionar productos y proveedores.

- Permitir el registro manual de productos y proveedores desde consola, incluyendo nombre, código, categoría, cantidad y datos de contacto.
- Habilitar la búsqueda de productos por nombre o categoría, mostrando sus detalles completos.
- Posibilitar la modificación de datos existentes de productos y proveedores.
- Incluir opción para eliminar productos o proveedores del sistema.
- Generar alertas cuando un producto tenga stock por debajo de un umbral definido.
- Documentar el funcionamiento del sistema y cómo se debe utilizar, junto con las limitaciones conocidas.

5. Marco Teórico

Lenguaje de programación C

```
/*
UNIVERSIDAD DE LAS
        FUERZAS ARMADAS
        ESPE
PROYECTO DE
        PROGRAMACION
CODIGO DE CONTROL DE
        INVENTARIO Y
        PROVEEDORES
GRUPO 1
ANTHONY CAMPOVERDE,
        MATEO VELECELA,
        ALEXANDER ALVEAR

*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#ifdef _WIN32
   #define CLEAR "cls"
#else
```

```
    #define CLEAR "clear"
#endif

#define MAX_PRODUCTOS 100
#define MAX_PROVEEDORES 50

typedef struct {
    char id[11];            // ID producto
        10 dígitos
    char nombre[50];
    float precio;           // Precio > 0,
        dos decimales
    int stock;
    int stockMinimo;
    char proveedorID[11];   // ID
        proveedor 10 dígitos
    int activo;
} Producto;

typedef struct {
    char id[11];            // ID proveedor
        10 dígitos
    char nombre[50];
    char telefono[20];
    char direccion[100];
    int activo;
} Proveedor;

Producto
        productos[MAX_PRODUCT
        OS];
Proveedor
        proveedores[MAX_PROVEE
        DORES];
int cantidadProductos = 0;
int cantidadProveedores = 0;
int ultimoIDProducto = 0;
int ultimoIDProveedor = 0;

// Prototipos
void limpiarBuffer();
void pausa();
void convertirMayusculas(char
        *texto);
void cargarDesdeArchivos();
void guardarEnArchivos();
void rellenarID(char *id);
void imprimirMenu();
```

```
void mostrarTablaProductos();
void mostrarTablaProveedores();
void agregarProducto();
void editarProducto();
void eliminarProducto();
void buscarProducto();
void agregarProveedor();
void editarProveedor();
void eliminarProveedor();
void buscarProveedor();

void limpiarBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c
            != EOF);
}

void pausa() {
    printf("\nPresione ENTER para
            continuar...");
    limpiarBuffer();
}

void convertirMayusculas(char
            *texto) {
    for (int i = 0; texto[i]; i++) texto[i]
            = toupper((unsigned
            char)texto[i]);
}

void rellenarID(char *id) {
    // Rellena con ceros a la izquierda
            para que id tenga 10
            caracteres
    int len = strlen(id);
    if (len > 10) id[10] = '\0'; // cortar
            si es más largo
    if (len < 10) {
        char temp[11];
        int diff = 10 - len;
        memset(temp, '0', diff);
        strcpy(temp + diff, id);
        temp[10] = '\0';
        strcpy(id, temp);
    }
}

char* generarID(int ultimoID) {
```

```
    static char nuevoID[11];
    sprintf(nuevoID, "%010d",
            ultimoID + 1);
    return nuevoID;
}

void imprimirMenu() {
    printf("\n========== SISTEMA
            DE INVENTARIO Y
            PROVEEDORES
            ==========\n");
    printf("---- PRODUCTOS ----\n");
    printf("1. Agregar producto\n");
    printf("2. Mostrar productos\n");
    printf("3. Buscar producto\n");
    printf("4. Editar producto\n");
    printf("5. Eliminar producto\n");
    printf("--------------------------------
            ------------------------\n");
    printf("---- PROVEEDORES ----
            \n");
    printf("6. Agregar proveedor\n");
    printf("7. Mostrar proveedores\n");
    printf("8. Buscar proveedor\n");
    printf("9. Editar proveedor\n");
    printf("10. Eliminar proveedor\n");
    printf("--------------------------------
            ------------------------\n");
    printf("11. Salir\n");
    printf("Seleccione una opcion: ");
}

void mostrarTablaProductos() {
    printf("\n%-12s%-20s%-10s%-
            8s%-14s%-12s\n", "ID",
            "NOMBRE", "PRECIO",
            "STOCK", "STOCK
            MINIMO",
            "PROVEEDOR");
    printf("--------------------------------
            ------------------------------
            ----------\n");
    for (int i = 0; i <
            cantidadProductos; i++) {
        if (productos[i].activo) {
            printf("%-12s%-
            20s%10.2f%-8d%-14d%-
            12s",
```

```
                productos[i].id,
                productos[i].nombre,
                productos[i].precio,
                productos[i].stock,

            productos[i].stockMinimo,

            productos[i].proveedorID);
            if (productos[i].stock == 0)
                printf(" <- SIN
            EXISTENCIA");
            else if (productos[i].stock <=
            productos[i].stockMinimo)
                printf(" <- STOCK
            BAJO");
            printf("\n");
        }
    }
    pausa();
}

void mostrarTablaProveedores() {
    printf("\n%-12s%-20s%-20s%-
            30s\n", "ID", "NOMBRE",
            "TELEFONO",
            "DIRECCION");
    printf("----------------------------------
            ----------------------------------
            ----------\n");
    for (int i = 0; i <
            cantidadProveedores; i++) {
        if (proveedores[i].activo) {
            printf("%-12s%-20s%-20s%-
            30s\n",
                proveedores[i].id,
                proveedores[i].nombre,
                proveedores[i].telefono,

            proveedores[i].direccion);
        }
    }
    pausa();
}

void agregarProducto() {
    if (cantidadProductos >=
            MAX_PRODUCTOS) {
        printf("No se pueden agregar
```

```
        mas productos.\n");
    pausa();
    return;
}
Producto p;
p.activo = 1;

ultimoIDProducto++;
strcpy(p.id,
        generarID(ultimoIDProducto
        - 1)); // Generar nuevo ID
printf("ID generado para producto:
        %s\n", p.id);

printf("Ingrese nombre del
        producto: ");
fgets(p.nombre, sizeof(p.nombre),
        stdin);
p.nombre[strcspn(p.nombre, "\n")]
        = '\0';
convertirMayusculas(p.nombre);

// Precio: validar > 0, con 2
        decimales
do {
    printf("Ingrese precio (> 0): ");
    if (scanf("%f", &p.precio) != 1)
        {
        printf("Entrada invalida.
        Intente de nuevo.\n");
        limpiarBuffer();
        p.precio = 0;
        continue;
    }
    if (p.precio <= 0) {
        printf("El precio debe ser
        mayor a cero.\n");
    }
    limpiarBuffer();
} while (p.precio <= 0);

// Stock >= 0
do {
    printf("Ingrese stock (0-1000):
        ");
    if (scanf("%d", &p.stock) != 1)
        {
        printf("Entrada invalida.
```

```
        Intente de nuevo.\n");
        limpiarBuffer();
        p.stock = -1;
        continue;
    }
    if (p.stock < 0 || p.stock > 1000)
        {
        printf("Stock invalido. Debe
        estar entre 0 y 1000.\n");
    }
    limpiarBuffer();
} while (p.stock < 0 || p.stock >
        1000);

// Stock minimo >= 0 y <= stock
do {
    printf("Ingrese stock minimo (0-
        %d): ", p.stock);
    if (scanf("%d",
        &p.stockMinimo) != 1) {
        printf("Entrada invalida.
        Intente de nuevo.\n");
        limpiarBuffer();
        p.stockMinimo = -1;
        continue;
    }
    if (p.stockMinimo < 0 ||
        p.stockMinimo > p.stock) {
        printf("Stock minimo
        invalido. Debe estar entre 0 y
        stock actual.\n");
    }
    limpiarBuffer();
} while (p.stockMinimo < 0 ||
        p.stockMinimo > p.stock);

// ID proveedor: debe existir y se
        autorellena
do {
    printf("Ingrese ID proveedor
        (solo números, se rellenara a
        10 digitos): ");
    char proveedorIDtemp[20];
    fgets(proveedorIDtemp,
        sizeof(proveedorIDtemp),
        stdin);

        proveedorIDtemp[strcspn(pro
```

```
            veedorIDtemp, "\n")] = '\0';

        rellenarID(proveedorIDtemp);

        // Verificar existencia
        int encontrado = 0;
        for (int i = 0; i <
            cantidadProveedores; i++) {
            if (strcmp(proveedores[i].id,
            proveedorIDtemp) == 0 &&
            proveedores[i].activo) {
                encontrado = 1;
                break;
            }
        }
        if (!encontrado) {
            printf("Proveedor no existe.
            Intente de nuevo.\n");
        } else {
            strcpy(p.proveedorID,
            proveedorIDtemp);
            break;
        }
    } while (1);

    productos[cantidadProductos++] =
            p;
    guardarEnArchivos();
    printf("Producto agregado
            exitosamente.\n");
    pausa();
}

void editarProducto() {
    if (cantidadProductos == 0) {
        printf("No hay productos para
            editar.\n");
        pausa();
        return;
    }
    char id[20];
    printf("Ingrese ID del producto a
            editar (numeros, se rellenara a
            10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);
```

```
for (int i = 0; i <
        cantidadProductos; i++) {
    if (strcmp(productos[i].id, id)
        == 0 && productos[i].activo)
        {
        printf("\n--- Editar Producto
        %s ---\n", productos[i].id);

        printf("Nuevo nombre (%s):
        ", productos[i].nombre);
        fgets(productos[i].nombre,
        sizeof(productos[i].nombre),
        stdin);

        productos[i].nombre[strcspn(
        productos[i].nombre, "\n")] =
        '\0';

        convertirMayusculas(product
        os[i].nombre);

        // Precio
        float precioNuevo;
        do {
            printf("Nuevo precio
        (%.2f): ",
        productos[i].precio);
            if (scanf("%f",
        &precioNuevo) != 1) {
                printf("Entrada invalida.
        Intente de nuevo.\n");
                limpiarBuffer();
                continue;
            }
            if (precioNuevo <= 0) {
                printf("El precio debe
        ser mayor a cero.\n");
                limpiarBuffer();
                continue;
            }
            limpiarBuffer();
            break;
        } while (1);
        productos[i].precio =
        precioNuevo;

        // Stock
        int stockNuevo;
```

```
do {
    printf("Nuevo stock (%d):
", productos[i].stock);
    if (scanf("%d",
&stockNuevo) != 1) {
        printf("Entrada invalida.
Intente de nuevo.\n");
        limpiarBuffer();
        continue;
    }
    if (stockNuevo < 0 ||
stockNuevo > 1000) {
        printf("Stock invalido.
Debe estar entre 0 y
1000.\n");
        limpiarBuffer();
        continue;
    }
    limpiarBuffer();
    break;
} while (1);
productos[i].stock =
stockNuevo;

// Stock mínimo
int stockMinNuevo;
do {
    printf("Nuevo stock
mínimo (%d): ",
productos[i].stockMinimo);
    if (scanf("%d",
&stockMinNuevo) != 1) {
        printf("Entrada invalida.
Intente de nuevo.\n");
        limpiarBuffer();
        continue;
    }
    if (stockMinNuevo < 0 ||
stockMinNuevo >
productos[i].stock) {
        printf("Stock minimo
invalido. Debe estar entre 0 y
stock actual.\n");
        limpiarBuffer();
        continue;
    }
    limpiarBuffer();
    break;
```

```
} while (1);
productos[i].stockMinimo =
stockMinNuevo;

// Proveedor ID
char provIDtemp[20];
do {
   printf("Nuevo proveedor
ID (%s): ",
productos[i].proveedorID);
   fgets(provIDtemp,
sizeof(provIDtemp), stdin);

provIDtemp[strcspn(provIDte
mp, "\n")] = '\0';

   if(strlen(provIDtemp) ==
0) {
      // Si no cambia, dejar
igual
      break;
   }

   rellenarID(provIDtemp);

   int encontrado = 0;
   for (int j = 0; j <
cantidadProveedores; j++) {
      if
(strcmp(proveedores[j].id,
provIDtemp) == 0 &&
proveedores[j].activo) {
         encontrado = 1;
         break;
      }
   }
   if (!encontrado) {
      printf("Proveedor no
existe. Intente de nuevo.\n");
   } else {

strcpy(productos[i].proveedor
ID, provIDtemp);
      break;
   }
} while (1);

guardarEnArchivos();
```

```
            printf("Producto
            actualizado.\n");
            pausa();
            return;
        }
    }
    printf("Producto no
        encontrado.\n");
    pausa();
}

void eliminarProducto() {
    if (cantidadProductos == 0) {
        printf("No hay productos para
            eliminar.\n");
        pausa();
        return;
    }
    char id[20];
    printf("Ingrese ID del producto a
        eliminar (numeros, se
        rellenara a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i <
        cantidadProductos; i++) {
        if (strcmp(productos[i].id, id)
            == 0 && productos[i].activo)
            {
            productos[i].activo = 0;
            guardarEnArchivos();
            printf("Producto eliminado
            lógicamente.\n");
            pausa();
            return;
        }
    }
    printf("Producto no
        encontrado.\n");
    pausa();
}

void agregarProveedor() {
    if (cantidadProveedores >=
        MAX_PROVEEDORES) {
        printf("No se pueden agregar
```

```
            mas proveedores.\n");
        pausa();
        return;
    }
    Proveedor p;
    p.activo = 1;

    ultimoIDProveedor++;
    strcpy(p.id,
            generarID(ultimoIDProveedo
            r - 1));
    printf("ID generado para
            proveedor: %s\n", p.id);

    printf("Ingrese nombre del
            proveedor: ");
    fgets(p.nombre, sizeof(p.nombre),
            stdin);
    p.nombre[strcspn(p.nombre, "\n")]
            = '\0';
    convertirMayusculas(p.nombre);

    printf("Ingrese telefono: ");
    fgets(p.telefono,
            sizeof(p.telefono), stdin);
    p.telefono[strcspn(p.telefono,
            "\n")] = '\0';

    printf("Ingrese direccion: ");
    fgets(p.direccion,
            sizeof(p.direccion), stdin);
    p.direccion[strcspn(p.direccion,
            "\n")] = '\0';


            proveedores[cantidadProveed
            ores++] = p;
    guardarEnArchivos();
    printf("Proveedor agregado
            exitosamente.\n");
    pausa();
}

void editarProveedor() {
    if (cantidadProveedores == 0) {
        printf("No hay proveedores para
            editar.\n");
        pausa();
```

```
    return;
}
char id[20];
printf("Ingrese ID del proveedor a
        editar (numeros, se rellenara a
        10 digitos): ");
fgets(id, sizeof(id), stdin);
id[strcspn(id, "\n")] = '\0';
rellenarID(id);

for (int i = 0; i <
        cantidadProveedores; i++) {
    if (strcmp(proveedores[i].id, id)
        == 0 &&
        proveedores[i].activo) {
        printf("\n--- Editar Proveedor
        %s ---\n", proveedores[i].id);

        printf("Nuevo nombre (%s):
        ", proveedores[i].nombre);
        fgets(proveedores[i].nombre,
        sizeof(proveedores[i].nombre
        ), stdin);

        proveedores[i].nombre[strcsp
        n(proveedores[i].nombre,
        "\n")] = '\0';

        convertirMayusculas(proveed
        ores[i].nombre);

        printf("Nuevo telefono (%s):
        ", proveedores[i].telefono);
        fgets(proveedores[i].telefono,
        sizeof(proveedores[i].telefono
        ), stdin);

        proveedores[i].telefono[strcsp
        n(proveedores[i].telefono,
        "\n")] = '\0';

        printf("Nueva direccion (%s):
        ", proveedores[i].direccion);

        fgets(proveedores[i].direccion
        ,
        sizeof(proveedores[i].direccio
        n), stdin);
```

```
            proveedores[i].direccion[strcs
            pn(proveedores[i].direccion,
            "\n")] = '\0';

            guardarEnArchivos();
            printf("Proveedor
            actualizado.\n");
            pausa();
            return;
        }
    }
    printf("Proveedor no
            encontrado.\n");
    pausa();
}

void eliminarProveedor() {
    if (cantidadProveedores == 0) {
        printf("No hay proveedores para
            eliminar.\n");
        pausa();
        return;
    }
    if (cantidadProveedores == 1) {
        printf("No puede eliminar al
            unico proveedor
            existente.\n");
        pausa();
        return;
    }

    char id[20];
    printf("Ingrese ID del proveedor a
            eliminar (numeros, se
            rellenara a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i <
            cantidadProveedores; i++) {
        if (strcmp(proveedores[i].id, id)
            == 0 &&
            proveedores[i].activo) {
            proveedores[i].activo = 0;
            guardarEnArchivos();
            printf("Proveedor eliminado
```

```
            logicamente.\n");
            pausa();
            return;
        }
    }
    printf("Proveedor no
            encontrado.\n");
    pausa();
}

void buscarProducto() {
    if (cantidadProductos == 0) {
        printf("No hay productos para
            buscar.\n");
        pausa();
        return;
    }
    char id[20];
    printf("Ingrese ID del producto a
            buscar (numeros, se rellenara
            a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i <
            cantidadProductos; i++) {
        if (strcmp(productos[i].id, id)
            == 0 && productos[i].activo)
            {
            printf("\nProducto
            encontrado:\n");
            printf("ID: %s\n",
            productos[i].id);
            printf("Nombre: %s\n",
            productos[i].nombre);
            printf("Precio: %.2f\n",
            productos[i].precio);
            printf("Stock: %d\n",
            productos[i].stock);
            printf("Stock minimo: %d\n",
            productos[i].stockMinimo);
            printf("Proveedor ID: %s\n",
            productos[i].proveedorID);
            pausa();
            return;
        }
    }
```

```
    printf("Producto no
        encontrado.\n");
    pausa();
}

void buscarProveedor() {
    if (cantidadProveedores == 0) {
        printf("No hay proveedores para
            buscar.\n");
        pausa();
        return;
    }
    char id[20];
    printf("Ingrese ID del proveedor a
        buscar (numeros, se rellenara
        a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i <
        cantidadProveedores; i++) {
        if (strcmp(proveedores[i].id, id)
            == 0 &&
            proveedores[i].activo) {
            printf("\nProveedor
            encontrado:\n");
            printf("ID: %s\n",
            proveedores[i].id);
            printf("Nombre: %s\n",
            proveedores[i].nombre);
            printf("Telefono: %s\n",
            proveedores[i].telefono);
            printf("Direccion: %s\n",
            proveedores[i].direccion);
            pausa();
            return;
        }
    }
    printf("Proveedor no
        encontrado.\n");
    pausa();
}

void guardarEnArchivos() {
    FILE *fp;

    // Guardar productos
```

```
    fp = fopen("productos.txt", "w");
    if (fp == NULL) {
        printf("Error al abrir archivo
            productos.txt\n");
        return;
    }
    for (int i = 0; i <
            cantidadProductos; i++) {
        if (productos[i].activo) {
            fprintf(fp,
            "%s|%s|%.2f|%d|%d|%s\n",
                    productos[i].id,
                    productos[i].nombre,
                    productos[i].precio,
                    productos[i].stock,

            productos[i].stockMinimo,

            productos[i].proveedorID);
        }
    }
    fclose(fp);

    // Guardar proveedores
    fp = fopen("proveedores.txt",
            "w");
    if (fp == NULL) {
        printf("Error al abrir archivo
            proveedores.txt\n");
        return;
    }
    for (int i = 0; i <
            cantidadProveedores; i++) {
        if (proveedores[i].activo) {
            fprintf(fp, "%s|%s|%s|%s\n",
                    proveedores[i].id,
                    proveedores[i].nombre,
                    proveedores[i].telefono,

            proveedores[i].direccion);
        }
    }
    fclose(fp);
}

void cargarDesdeArchivos() {
    FILE *fp;
    char linea[256];
```

```
cantidadProductos = 0;
cantidadProveedores = 0;

// Cargar proveedores primero
      (para validar al agregar
      productos)
fp = fopen("proveedores.txt", "r");
if (fp != NULL) {
   while (fgets(linea, sizeof(linea),
      fp)) {
      Proveedor p;
      p.activo = 1;
      linea[strcspn(linea, "\n")] =
      '\0';
      char *token = strtok(linea,
      "|");
      if (!token) continue;
      strcpy(p.id, token);
      token = strtok(NULL, "|");
      if (!token) continue;
      strcpy(p.nombre, token);
      token = strtok(NULL, "|");
      if (!token) continue;
      strcpy(p.telefono, token);
      token = strtok(NULL, "|");
      if (!token) continue;
      strcpy(p.direccion, token);

      // Actualizar
      ultimoIDProveedor
      int idInt = atoi(p.id);
      if (idInt >
      ultimoIDProveedor)
      ultimoIDProveedor = idInt;


      proveedores[cantidadProveed
      ores++] = p;
      if (cantidadProveedores >=
      MAX_PROVEEDORES)
      break;
   }
   fclose(fp);
}

// Cargar productos
fp = fopen("productos.txt", "r");
```

```
    if (fp != NULL) {
        while (fgets(linea, sizeof(linea),
            fp)) {
            Producto p;
            p.activo = 1;
            linea[strcspn(linea, "\n")] =
            '\0';
            char *token = strtok(linea,
            "|");
            if (!token) continue;
            strcpy(p.id, token);
            token = strtok(NULL, "|");
            if (!token) continue;
            strcpy(p.nombre, token);
            token = strtok(NULL, "|");
            if (!token) continue;
            p.precio = atof(token);
            token = strtok(NULL, "|");
            if (!token) continue;
            p.stock = atoi(token);
            token = strtok(NULL, "|");
            if (!token) continue;
            p.stockMinimo = atoi(token);
            token = strtok(NULL, "|");
            if (!token) continue;
            strcpy(p.proveedorID, token);

            // Actualizar
            ultimoIDProducto
            int idInt = atoi(p.id);
            if (idInt > ultimoIDProducto)
            ultimoIDProducto = idInt;


            productos[cantidadProductos
            ++] = p;
            if (cantidadProductos >=
            MAX_PRODUCTOS) break;
        }
        fclose(fp);
    }
}

int main() {
    cargarDesdeArchivos();
    int opcion;
    do {
        system(CLEAR);
```

```
        imprimirMenu();
        if (scanf("%d", &opcion) != 1) {
            printf("Entrada invalida.\n");
            limpiarBuffer();
            opcion = 0;
            continue;
        }
        limpiarBuffer();

        switch (opcion) {
            case 1: agregarProducto();
            break;
            case 2:
            mostrarTablaProductos();
            break;
            case 3: buscarProducto();
            break;
            case 4: editarProducto();
            break;
            case 5: eliminarProducto();
            break;
            case 6: agregarProveedor();
            break;
            case 7:
            mostrarTablaProveedores();
            break;
            case 8: buscarProveedor();
            break;
            case 9: editarProveedor();
            break;
            case 10: eliminarProveedor();
            break;
            case 11:
            printf("Saliendo...\n"); break;
            default: printf("Opcion
            invalida.\n"); pausa(); break;
        }
    } while (opcion != 11);

    return 0;
}
```

5.1. Metodología

¿QUÉ?	¿CÓMO?	¿QUIÉN?	¿CUÁNDO?	¿POR QUÉ?	% CUMPLIMIENTO
Desarrollo de una aplicación en C para gestionar productos y proveedores.	Mediante programación estructurada, uso de arreglos/listas, funciones y archivos.	Alvear Alexander, Campoverde Anthony, Velecela Mateo.	Desde el 28 de mayo hasta julio/agosto de 2025.	Para resolver problemas del inventario manual y aplicar conocimientos de programación.	90% (en desarrollo)
Implementación de alertas de stock mínimo.	Programación de lógica condicional para comparar niveles de stock con umbrales.		Durante la fase de implementación funcional.	Para evitar quiebres de stock y mejorar la toma de decisiones.	100%
Validación y pruebas del sistema.	Aplicación de pruebas de caja blanca y caja negra.		Finales de julio 2025.	Para garantizar la funcionalidad y confiabilidad del sistema.	90%
Documentación del proyecto.	Redacción de informes técnicos y manual de usuario.	Estudiantes.	Durante y al final del desarrollo.	Para entregar un proyecto completo y comprensible.	70%

6. Ideas a Defender

"Los pequeños negocios pueden beneficiarse de soluciones tecnológicas simples."

"La programación en C sigue siendo útil para construir herramientas prácticas."

"Es posible aplicar conceptos teóricos de programación en problemas reales de negocios."

7. Resultados Esperados

Se espera entregar una herramienta funcional en C que permita a la dueña del negocio registrar y gestionar sus productos y proveedores de forma eficiente, reduciendo errores humanos y mejorando el control del inventario.

8. Viabilidad

(Todavía en desarrollo)

8.1. Humana

8.1.1. Tutor Empresarial: (Hilda Guajala)

Responsable de: Aportar Contexto al Proyecto.

8.1.2. Tutor Académico: (Ing. Jenny A Ruiz R)

Responsable de: Guía Técnica del Proyecto.

8.1.3. Estudiantes: (Alvear Alexander, Campoverde Anthony, Velecela Mateo)

Responsables de: Programación, pruebas y documentación.

8.2. Tecnológica

(Todavía en desarrollo)

9. Conclusiones y Recomendaciones

(Todavía en desarrollo)

10. Planificación para el Cronograma

#	TAREA	INICIO	FIN
1	Desarrollo de Introducción (v1 de Perfil P.)	28/05/2025	29/05/2025
2	Desarrollo de Alc.,Objeti. (Definido) (v2 de Perfil P.)	18/06/2025	20/06/2025
3	Desarrollo de la metodología (v3 de Perfil p.)	10/07/2025	10/07/2025
4	Cambio de Objetivos Específicos y Alcance (v4 de Perfil p.)	17/07/2025	17/07/2025
5	Marco teórico (v5 de Perfil p.)	24/07/2025	30/07/2025
6			
7			
8			

Tabla 1- Cronograma del Proyecto

11. Referencias

(Todavía en desarrollo)