UNIMINUTO
Corporación Universitaria Minuto de Dios
Educación de calidad al alcance de todos
Vigilada MinEducación

UNIVERSIDAD MINUTO DE DIOS

Avance y Evaluación del Proyecto de Programación de un Sistema de Bus de Datos Simple

Docente: Rubén Darío González Barrera

Asignatura:

Arquitectura De Computadores

Presentado por:

Deibid Benavides Yaya

Mateo Ramirez Ortiz

Mateo Alejandro Vargas Valero

Institución: Universidad Minuto de Dios

Fecha:16/10/2025

Introducción

En el contexto actual de la arquitectura de computadoras y la programación de sistemas, comprender los mecanismos de comunicación interna es fundamental. Este proyecto se centra en el desarrollo de un Sistema de Bus de Datos Simple programado. Un bus de datos es esencialmente un conjunto de cables o pistas que permiten la transferencia de datos entre los diferentes componentes de un sistema informático (CPU, memoria, dispositivos de E/S).

El problema identificado es la necesidad de un modelo simplificado y funcional que sirva como herramienta educativa para visualizar y simular el proceso básico de transferencia de información a través de un bus. La importancia de este desarrollo radica en que proporciona un entorno práctico para aplicar y consolidar conceptos de diseño de *hardware* y *software*, estructuras de datos (como registros o colas de transferencia) y lógica de control en la programación.

Los objetivos generales del proyecto son desarrollar un sistema que simule la funcionalidad esencial de un bus de datos. Esto se relaciona directamente con el aprendizaje en programación al requerir la aplicación práctica de clases, manejo de estados, sincronización (simulada) y buenas prácticas de desarrollo como el modularidad y la abstracción.

justificación

El proyecto de un Sistema de Bus de Datos Simple es de alta relevancia académica ya que conecta la teoría de la arquitectura de computadoras con la práctica de la programación. Permite a los estudiantes *ver* cómo los datos viajan internamente, una noción a menudo abstracta.

La decisión de desarrollar este proyecto se basa en que proporciona un entorno controlado para experimentar con la gestión de recursos compartidos y el diseño de interfaces modulares (como la interfaz del *hardware* simulado con el bus).

El impacto que puede tener es primariamente en el ámbito académico, sirviendo como un simulador educativo que clarifica conceptos como las señales de control (lectura/escritura), las direcciones de memoria y la concurrencia (simulada). Pretende resolver la necesidad de una herramienta práctica que demuestre el mecanismo central de comunicación interna en cualquier sistema digital, lo cual es crucial para futuros desarrollos en sistemas operativos, *drivers* o programación de bajo nivel.

Objetivos

Objetivo General

Desarrollar un proyecto de programación que integre los conocimientos adquiridos en clase, aplicando conceptos de diseño (*Object-Oriented Design* - OOD), lógica, estructuras de datos y buenas prácticas de desarrollo para simular un Sistema de Bus de Datos Simple.

Objetivos Específicos

Identificar el problema o necesidad que el proyecto resolverá: Crear una simulación didáctica de la transferencia de datos en un bus interno.

Diseñar la estructura lógica y técnica del programa: Definir las clases para los componentes (*CPU*, *Memoria*, *Bus*) y sus métodos de interacción.

Implementar el proyecto en el lenguaje de programación correspondiente (e.g., Python, C++, Java): Codificar las clases y la lógica de control del bus.

Probar, documentar y presentar los resultados del proyecto: Verificar la transferencia correcta de datos, documentar el código y presentar el simulador funcional.

Avance del Proyecto

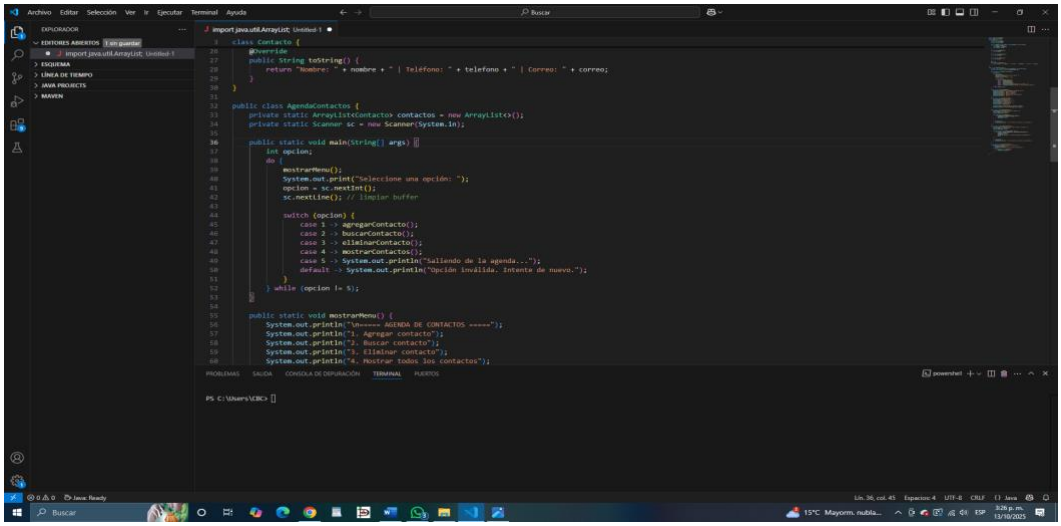
En esta sección se detallarán los avances logrados, estructurados por las etapas clave del desarrollo.

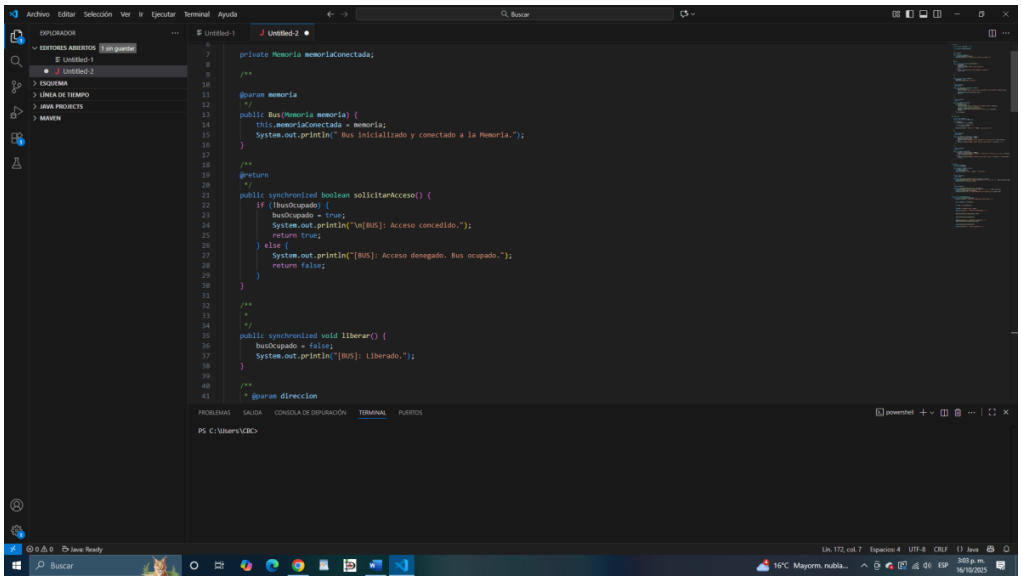
Etapas Completadas:

1. **Definición de Clases: Se definieron las clases Componente (como clase base), CPU, Memoria y Bus.**
2. **Implementación del Bus: Se programaron los métodos escribir() y leer() en la clase Bus, incluyendo un mecanismo simple de arbitraje (simulado por una cola FIFO).**
3. **Comunicación Básica: Se logró simular una operación simple de escritura de la CPU a una dirección de la Memoria a través del Bus.**

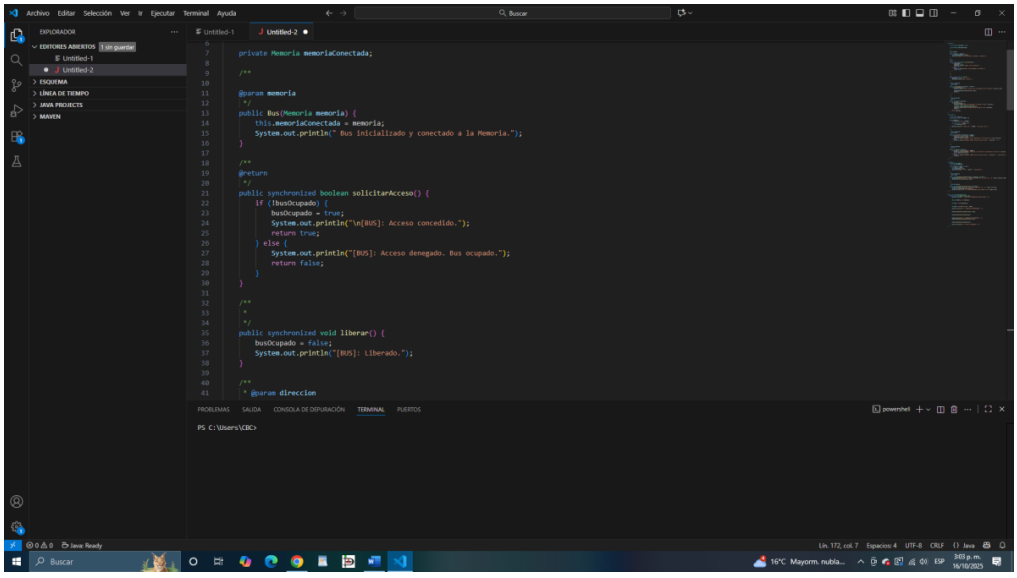
Problemas Encontrados y Soluciones Aplicadas:

- **Problema: Inicialmente, el manejo de direcciones y datos en el Bus era rígido.**
- **Solución: Se implementó un objeto de "Paquete de Transferencia" que encapsula la dirección, el dato y el tipo de operación (Lectura/Escritura), haciendo el código más modular y extensible.**
- **Problema: Asegurar que solo un componente acceda al bus en un momento dado (simulación de arbitraje).**
- **Solución: Se utilizó una variable de estado booleana (bus_ocupado) en la clase Bus y un *mecanismo de solicitud/liberación* para simular un bloqueo simple.**

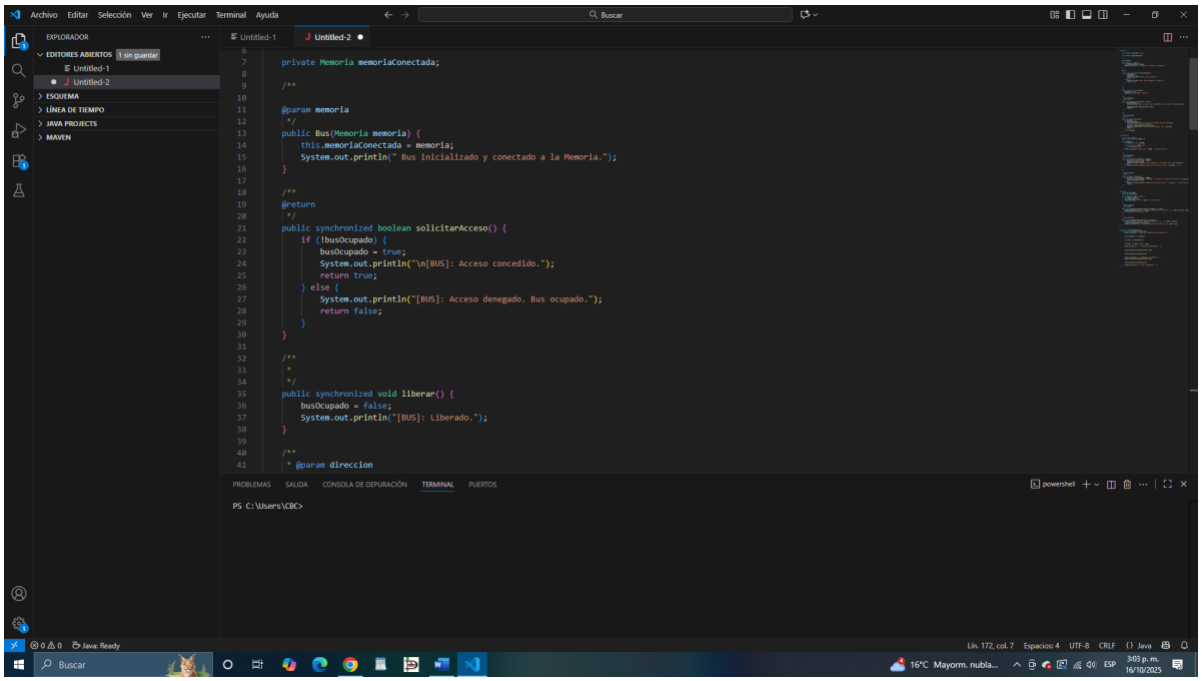




```java
private Memoria memoriaConectada;

/**

@param memoria
 */
public Bus(Memoria memoria) {
    this.memoriaConectada = memoria;
    System.out.println(" Bus inicializado y conectado a la Memoria.");
}

/**
@return
 */
public synchronized boolean solicitarAcceso() {
    if (!busOcupado) {
        busOcupado = true;
        System.out.println("\n[BUS]: Acceso concedido.");
        return true;
    } else {
        System.out.println("[BUS]: Acceso denegado. Bus ocupado.");
        return false;
    }
}

/**
 *
 */
public synchronized void liberar() {
    busOcupado = false;
    System.out.println("[BUS]: Liberado.");
}

/**
 * @param direccion
```




```java
private Memoria memoriaConectada;

/**

@param memoria
 */
public Bus(Memoria memoria) {
    this.memoriaConectada = memoria;
    System.out.println(" Bus inicializado y conectado a la Memoria.");
}

/**
@return
 */
public synchronized boolean solicitarAcceso() {
    if (!busOcupado) {
        busOcupado = true;
        System.out.println("\n[BUS]: Acceso concedido.");
        return true;
    } else {
        System.out.println("[BUS]: Acceso denegado. Bus ocupado.");
        return false;
    }
}

/**
 *
 */
public synchronized void liberar() {
    busOcupado = false;
    System.out.println("[BUS]: Liberado.");
}

/**
 * @param direccion
```



Fecha	Actividad realizada	Estado (En proceso/Completado)	Observaciones
10/10/2025	Definición de clases: Bus, CPU, Memoria.	Completado	Modelado inicial de atributos y métodos clave.

Fecha	Actividad realizada	Estado (En proceso/Completado)	Observaciones
12/10/2025	Implementación de métodos solicitar acceso() y escribir () del Bus.	Completado	Se integró el mecanismo de arbitraje simple (bus_ocupado).
14/10/2025	Implementación del método leer () y simulación de interacción completa Memoria	En proceso	Pendiente la prueba exhaustiva del mecanismo de lectura.
16/10/2025	Documentación de clases y métodos, preparación de la presentación.	En proceso	Finalizar la documentación en línea con las buenas prácticas.

Resumen del Proyecto

El proyecto ha resultado en un simulador funcional y didáctico de un Sistema de Bus de Datos Simple, implementado con un enfoque de Programación Orientada a Objetos. Se definieron y programaron con éxito los componentes principales (Bus, CPU, Memoria) y se logró simular las operaciones fundamentales de lectura y escritura de datos, incluyendo una aproximación al arbitraje de bus para gestionar el acceso concurrente (simulado).

Resultados Obtenidos: Se tiene un *script* ejecutable que demuestra cómo la CPU puede *solicitar* el bus, enviar una dirección y un dato, y cómo la Memoria *recibe* dicha información. El código es modular y fácil de entender.

Principales Conclusiones: El desarrollo reforzó la comprensión de la abstracción de *hardware* en código, la importancia de las interfaces (métodos de las clases) para la comunicación entre componentes, y la necesidad de mecanismos de control (arbitraje) en sistemas concurrentes.

Webgrafía:

https://es.wikipedia.org/wiki/Bus_(inform%C3%A1tica)

https://hardzone.es/reportajes/que-es/bus-datos-pc-cuales/

https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial

https://www.cursosaula21.com/que-es-un-bus-de-campo/