Nombre del Alumno: ***Vania Natali Santizo Morales***

Nombre del tema: ***Trabajo Plataforma 2***

Parcial: ***2° Parcial***

Nombre de la Materia: ***Ingeniería en Software***

Nombre del profesor: ***Andres Alejandro Reyes Molina***

Nombre de la Licenciatura: ***Ingeniería en Sistemas Computacionales***

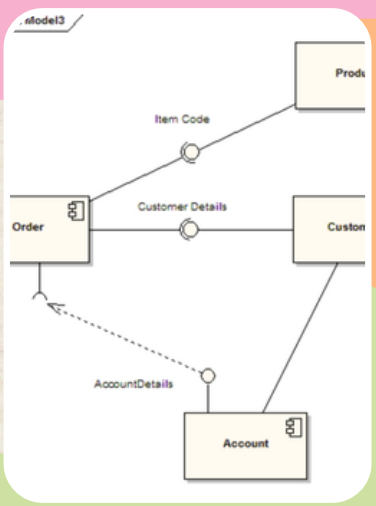
Cuatrimestre: ***8°***

SUPERNOTA

4.1 MODELOS DE IMPLEMENTACIÓN

Los modelos de implementación describen cómo los componentes del software se integran y se ejecutan en una arquitectura concreta.

Tipos principales:
Monolítico: Todo el código en un solo bloque ejecutable. Fácil de desplegar, difícil de mantener.
Cliente-servidor: Separación entre cliente (interfaz) y servidor (lógica y datos).
Basado en servicios (SOA): Usa servicios independientes que interactúan entre sí.
Microservicios: Cada funcionalidad es un servicio autónomo. Escalable, ideal para DevOps.
En la nube (Cloud-native): Adaptado para ambientes cloud, con escalabilidad dinámica.



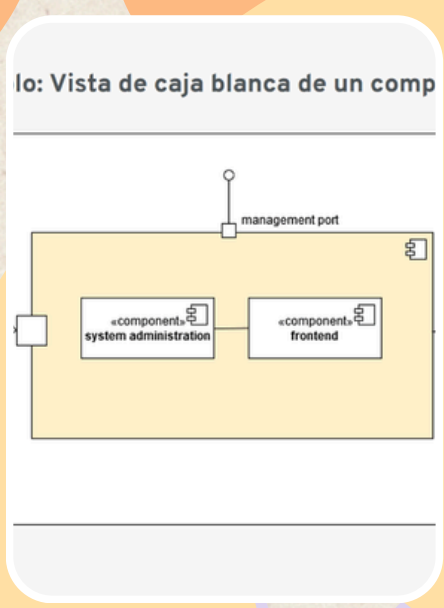
4.2 DIAGRAMAS DE COMPONENTES

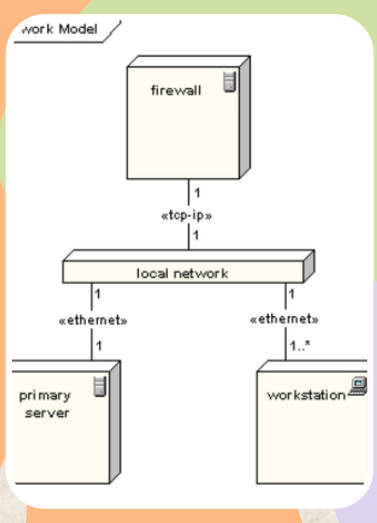
Son diagramas UML que muestran cómo el sistema se divide en componentes físicos (módulos de código, bibliotecas, ejecutables, etc.).

Objetivo: Visualizar la organización y dependencias entre componentes de software.

4.3 ELEMENTOS DEL DIAGRAMA DE COMPONENTES

Componentes: Representan partes reutilizables del sistema (ej. clases, librerías).
Interfaces: Indican qué servicios ofrece o necesita un componente.
Dependencias: Muestran relaciones de uso entre componentes.
Artefactos: Archivos físicos (JAR, DLL, scripts).
Conectores: Representan canales de comunicación entre componentes.





4.4 DIAGRAMAS DE DESPLIEGUE

Representan la infraestructura del sistema, indicando cómo los componentes se distribuyen en el hardware.
Elementos principales:
Nodos: Dispositivos físicos o virtuales (servidores, móviles, etc.).
Artefactos: Programas que se ejecutan en los nodos.
Conexiones: Comunicación entre nodos (redes, protocolos).

4.6 IMPLEMENTACIÓN EN JAVA DE LOS DIAGRAMAS DE CLASE

Ejemplo básico:

```
public class Persona {
private String nombre;
private int edad;
public Persona(String nombre,
int edad) {
this.nombre = nombre;
this.edad = edad;
}
public void saludar() {
System.out.println("Hola, soy
" + nombre);
}
}
```

4.7 CONSTRUCTORES Y DESTRUCTORES: DECLARACIÓN, USO Y APLICACIONES

Constructor: Método especial que se ejecuta al crear un objeto.

```
public Clase() {
// inicialización
}
```

Destructor: Java no tiene destructores como C++; se usa finalize() (obsoleto) y se prefiere el recolector de basura.

Uso: Inicializar objetos, asignar recursos (constructores).
Liberar memoria (destructores).