

Coordinador de réplicas
Máquina 3

Recursos de interés:
• Transmisión de archivos a través de sockets, en Java.
• Pasar un fichero por socket en java.



Coordinador
+ main(args: String[]): void

También es un ServerSocket para recibir las peticiones del servidor web

Backer
- serverSocket: ServerConnection
+ backUpInventoryFile(): void
+ sendInventoryFile(): void

Restorer
- serverSocket: ServerConnection
+ restoreInventoryFile(): void
+ receiveInventoryFile(): void

ServerConnection
- serverSocket: Socket
+ ServerConnection(server: int): ServerConnection
+ getServerSocket(): Socket
+ sendRequestToSocketServer(message: String): String

Servidor de réplicas / restauración
Máquinas 1 y 2

ReplicaRestoreServer
+ main(args: String[]): void
+ selectResponse(): String

Backer
- coordinatorSocket: Socket
+ Backer(socket: Socket): Backer
+ backUpInventoryFile(): void
+ receiveInventoryFile(): void

Restorer
- coordinatorSocket: Socket
+ Restorer(socket: Socket): Restorer
+ restoreInventoryFile(): void
+ sendInventoryFile(): void

inventory.json

Tarro (Monitor)
Máquina 3

Servidor RMI

Legenda:
@method() -> synchronized method

Servidor web
Máquina 3

Cliente RMI

Recursos de interés:
• Cómo crear un API Rest con Spring Boot.
• ¿Qué es la arquitectura orientada a los servicios (SOA)?
• Arquitectura Orientada a Servicios (SOA).

CRASH API REST CON SPRING
• Como API REST con Spring, Springboot, Docker y Redis

Red Hat
• ¿Qué es la arquitectura orientada a los servicios?

ApiApplication
+ main(args: String[]): void

Paquete Controller
ConsumerController
+ consumeProduct(request: JSONObject): void
+ checkInventory(): JSONArray
+ checkInventoryExists(): JSONObject
ProducerController
+ fillJar(request: JSONObject): void
+ backUpInventory(): void
+ checkInventoryExists(): JSONObject

Paquete Service
IConsumerService
+ consumeProduct(product: String, quantity: int): void
+ checkInventory(): JSONArray
IClientService
+ checkInventoryExists(): boolean
IProducerService
+ fillJar(product: String, quantity: int): void
+ backUpInventory(): void
+ action(): void
ConsumerService
+ consumeProduct(product: String, quantity: int): void
+ checkInventory(): JSONArray
+ checkInventoryExists(): boolean
ProducerService
+ fillJar(product: String, quantity: int): void
+ backUpInventory(): void
+ checkInventoryExists(): boolean
+ action(): void
+ connectWithCoordinator(): void

Paquete DAO
IConsumerDAO
+ consumeProduct(product: String, quantity: int): void
+ checkInventory(): JSONArray
IClientDAO
+ checkInventoryExists(): boolean
IProducerDAO
+ fillJar(product: String, quantity: int): void
+ backUpInventory(): void
+ action(): void
ConsumerDAO
+ consumeProduct(product: String, quantity: int): void
+ checkInventory(): JSONArray
+ checkInventoryExists(): boolean
ProducerDAO
+ fillJar(product: String, quantity: int): void
+ backUpInventory(): void
+ checkInventoryExists(): boolean
+ action(): void

Paquete RMCommon
IClient
+ setJar(jar: Jar): void
+ setProduct(product: Product): void
+ setProductQuantity(quantity: int): void
+ updateInventory(): void
+ checkInventory(): JSONObject

Paquete RMIServer
ConsumerRMIServer
+ ConsumerRMIServer(jar: Jar): ConsumerRMIServer
+ updateInventory(): void
ClientRMIServer
product: Product
quantity: int
jar: Jar
client: Client
+ ClientRMIServer(jar: Jar, client: Client): ClientRMIServer
+ run(): void
+ setProduct(product: Product): void
+ setProductQuantity(quantity: int): void
ProducerRMIServer
+ ProducerRMIServer(jar: Jar): ProducerRMIServer
+ updateInventory(): void

Remote
Registry
RMIServer
- PORT = 1100: int
- consumer: Remote
- producer: Remote
- registry: Registry
+ main(args: String[]): void
- consumerBinding(): void
- producerBinding(): void

Paquete Inventory
InventoryFileManager
- instance: Jar
- inventory: ArrayList<Transaction>
+ Jar(): Jar
+ getInstance(): Jar
+ getInventory(): JSONArray
+ @updateInventory(client: Client, product: Product, quantity: int): void
+ getLastTransaction(inventory: ArrayList<Transaction>): Transaction
+ updateInventoryIfConsumer(productStocks: ArrayList<ProductStock>, product: Product, quantity: int): ArrayList<ProductStock>
+ updateInventoryIfProducer(productStocks: ArrayList<ProductStock>, product: Product, quantity: int): ArrayList<ProductStock>

InventoryFileManager
- INVENTORY_FILE_PATH: String
+ getInventory(): ArrayList<Transaction>
+ getJSONInventory(): JSONArray
+ setInventory(inventory: ArrayList<Transaction>): void

ProductStock
- product: Product
- quantity: int
+ add(quantity: int): ProductStock
+ subtract(quantity: int): ProductStock

Transaction
- quantity: int
- product: String
- actor: String
- actionDateTime: String
- currentStock: ArrayList<ProductStock>
+ getJSONTransaction(): JSONObject

Thread
+ run(): void
+ start(): void

Paquete Common
Product
+ A
+ B
+ getProductName(): String
+ getProduct(name: String): Product
+ isProductA(): boolean
+ isProductB(): boolean
+ isEqualTo(productStock: ProductStock): boolean
Client
+ CONSUMER
+ PRODUCER
+ getClientName(): String
+ getClient(name: String): Client
+ isConsumer(): boolean
+ isProducer(): boolean



Recursos de interés:
• Escribir datos JSON en un archivo en Java.
• ¿Cómo trabajar con JSON fácilmente en Java?
• Leer y escribir JSON en Java.
• Leer archivo JSON en Java.



Crear archivo JSON fácilmente en Java

