

**Tema**

INTERCEPTORS, TIMERS Y GENERACIÓN DE SESSION BEANS EN JAVA

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU.CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

17/07/2023

INTERCEPTORS, TIMERS Y GENERACIÓN DE SESSION BEANS EN JAVA

[1. MARCO TEÓRICO 4](#_Toc125408924)

[1.1. INTRODUCCIÓN 4](#_Toc125408925)

[1.2. OBJETIVOS 4](#_Toc125408926)

[**1.2.1.** **OBJETIVO GENERAL** 4](#_Toc125408927)

[**1.2.2.** **OBJETIVOS ESPECÍFICOS** 4](#_Toc125408928)

[1.3. APLICACIÓNES WEB EN IDE NETBEANS 4](#_Toc125408929)

[**1.3.1.** **PAYARA SERVER** 5](#_Toc125408930)

[1.4. JAVA WEB 5](#_Toc125408931)

[1.5. JAVABEANS 7](#_Toc125408932)

[1.6. MANAGED BEAN 7](#_Toc125408933)

[1.7. JAVA TRANSACTION API 8](#_Toc125408934)

[1.7.1 TRANSACCIONES 8](#_Toc125408935)

[1.7.2 OAP (PROGRAMACIÓN ORIENTADA A ASPECTOS) 10](#_Toc125408936)

[1.7.3 ENTERPRISE JAVA BEANS 13](#_Toc125408937)

[1.7.4 ENTIDADES JPA PERSISTENTES 16](#_Toc125408938)

[1.8. PALETA DE COMPONENTES 17](#_Toc125408939)

[1.9. ETIQUETAS HTML STANDARD 21](#_Toc125408940)

[1.10. PATRONES DE DISEÑO 22](#_Toc125408941)

[2. PARTE PRÁCTICA 24](#_Toc125408942)

[2.1. CREACIÓN DEL PROYECTO 24](#_Toc125408943)

[**2.1.1.** **CODIFICACIÓN DEL ARCHIVO REPETIR.JAVA** 27](#_Toc125408944)

[**2.1.2.** **CODIFICACIÓN DEL ARCHIVO LOGGINGINTERCEPTOR.JAVA** 32](#_Toc125408945)

[2.2. CODIFICACIÓN DEL ARCHIVO EJBTIMERSDEMO.JAVA 35](#_Toc125408946)

[2.3. CODIFICACIÓN DEL ARCHIVO MAIN.JAVA 38](#_Toc125408947)

[2.4 DESEMPEÑO DE LA APLICACIÓN 40](#_Toc125408948)

[**2.4.1** **INTERCEPTORS\_TIMERS\_SESSIONBEAN\_GRUPO1** 40](#_Toc125408949)

[3. CONCLUSIONES 44](#_Toc125408950)

[4. RECOMENDACIONES 44](#_Toc125408951)

[5. REFERENCIAS 44](#_Toc125408952)

**INDICE DE IMÁGENES**

[Figura 1. Arquitectura de aplicación web 5](#_Toc125410774)

[Figura 2. Dominio Payara Server 6](#_Toc125410775)

[Figura 3. Representación estructura de un Java Transaction API 9](#_Toc125410776)

[Figura 4. Presentación del video referente a transacciones 10](#_Toc125410777)

[Figura 5. Ejemplo de codigo empleando transacciones 10](#_Toc125410778)

[Figura 6. Ejemplo de codigó 12](#_Toc125410779)

[Figura 7. Presentación del video referente al tema de OAP e interceptores 13](#_Toc125410780)

[Figura 8. Interceptores explicados dentro del ejercicio práctico 13](#_Toc125410781)

[Figura 9. Ejemplo de código aplicando interceptores 14](#_Toc125410782)

[Figura 10. Ejemplo de aplicación de SessionBeans 15](#_Toc125410783)

[Figura 11. Presentación del video referente al tema de timers 16](#_Toc125410784)

[Figura 12. Ejemplo de aplicación de javax.swing.Timer 16](#_Toc125410785)

[Figura 13. Ejemplo de aplicacion de java.util.Timer 18](#_Toc125410786)

[Figura 14.Estructura referente a DAOs 19](#_Toc125410787)

[Figura 15. Pantalla de inicio del IDE NetBeans 20](#_Toc125410788)

[Figura 16. Menú desplegable 20](#_Toc125410789)

[Figura 17. Submenú de la opcion IDE Tools 21](#_Toc125410790)

[Figura 18. PAntalla de inicio de paleta del IDE 21](#_Toc125410791)

[Figura 19. Paleta de componentes de NetBeans 22](#_Toc125410792)

[Figura 20. Opciones del menú File 25](#_Toc125410793)

[Figura 21. Pantalla para selección del tipo de proyecto a crear 26](#_Toc125410794)

[Figura 22. Colocar el nombre del proyecto principal 26](#_Toc125410795)

[Figura 23. Ventana donde se seleccionará la versión de JEE y el servidor 27](#_Toc125410796)

[Figura 24. Selección del proyecto tipo Client 27](#_Toc125410797)

[Figura 25. Selección de la versión del JEE y adjuntando al proyecto principal 28](#_Toc125410798)

[Figura 26. Estructura final del proyecto 28](#_Toc125410799)

[Figura 27. Selección del archivo tipo Session Beans 29](#_Toc125410800)

[Figura 28. Asignar nombre al archivo Session Bean y seleccionando las opciones del archivo 29](#_Toc125410801)

[Figura 29. Menú de opciones del archivo Session Beans 30](#_Toc125410802)

[Figura 30. Ventana de códigos a seleccionar 30](#_Toc125410803)

[Figura 31. Ventana en la cual se añadirá el método a usar en el programa 31](#_Toc125410804)

[Figura 32. Vista del código del archivo Repetir.java 31](#_Toc125410805)

[Figura 33. Selección del tipo de archivo Session Beans 33](#_Toc125410806)

[Figura 34. Asignar nombre al archivo que se creará 33](#_Toc125410807)

[Figura 35. Vista del archivo resultante LoggingInterceptor.java 34](#_Toc125410808)

[Figura 36. Selección del tipo de archivo a crear en el proyecto 36](#_Toc125410809)

[Figura 37. Asognar el nombre respectivo al archivo creado 36](#_Toc125410810)

[Figura 38. Vista final del archivo EJBTimersDemo.java 37](#_Toc125410811)

[Figura 39. Actualización del nombre del paquete creado automáticamente 39](#_Toc125410812)

[Figura 40. Vista final del archivo Main.java 39](#_Toc125410813)

[Figura 41. Menú de opciones de un proyecto 41](#_Toc125410814)

[Figura 42. Seleccionando la opción de Run ejecutando el proyecto 42](#_Toc125410815)

[Figura 43. Opciones del menú de herramientas 42](#_Toc125410816)

[Figura 44. Pestaña de Services 43](#_Toc125410817)

[Figura 45. Menú de opciones del servidor de payara server 43](#_Toc125410818)

[Figura 46. Output con información de la payara iniciándose 44](#_Toc125410819)

[Figura 47. Resultado del inicio del payara server 44](#_Toc125410820)

[Figura 48. Vista de como se ve el payara server iniciado 45](#_Toc125410821)

[Figura 49. Ventana resultante del método repetir 45](#_Toc125410822)

[Figura 50. Resultado de los interceptores por medio de consola 45](#_Toc125410823)

**ÍNDICE DE TABLAS**

[Tabla 1. Operadores usados en Java 6](#_Toc125410764)

[Tabla 2. Tabla de valores del atributo transacción 10](#_Toc125410765)

[Tabla 3. Etiquetas HTML Standard 22](#_Toc125410766)

[Tabla 4. Código del archivo Repetir.java 31](file:///C:\Users\DETPC\Desktop\ESPE\Sexto-Septimo%20Semestre\APLICACIONES%20DISTRIBUIDAS\SEGUNDO%20PARCIAL\TRABAJOS%20GRUPALES\17.%20T1.14%20INTERCEPTORS_TIMERS_GENERACIÓN_SESSIONBEANS_GRUPO1\DOCUMENTACION\Interceptors_Timers_Generación_SessionBeans_Java_Grupo1.docx#_Toc125410767)

[Tabla 5. Código del archivo LoggingInterceptor.java 34](file:///C:\Users\DETPC\Desktop\ESPE\Sexto-Septimo%20Semestre\APLICACIONES%20DISTRIBUIDAS\SEGUNDO%20PARCIAL\TRABAJOS%20GRUPALES\17.%20T1.14%20INTERCEPTORS_TIMERS_GENERACIÓN_SESSIONBEANS_GRUPO1\DOCUMENTACION\Interceptors_Timers_Generación_SessionBeans_Java_Grupo1.docx#_Toc125410768)

[Tabla 6.Código del archivo EJBTimersDemo.java 37](file:///C:\Users\DETPC\Desktop\ESPE\Sexto-Septimo%20Semestre\APLICACIONES%20DISTRIBUIDAS\SEGUNDO%20PARCIAL\TRABAJOS%20GRUPALES\17.%20T1.14%20INTERCEPTORS_TIMERS_GENERACIÓN_SESSIONBEANS_GRUPO1\DOCUMENTACION\Interceptors_Timers_Generación_SessionBeans_Java_Grupo1.docx#_Toc125410769)

[Tabla 7. Código del archivo Main.java 39](file:///C:\Users\DETPC\Desktop\ESPE\Sexto-Septimo%20Semestre\APLICACIONES%20DISTRIBUIDAS\SEGUNDO%20PARCIAL\TRABAJOS%20GRUPALES\17.%20T1.14%20INTERCEPTORS_TIMERS_GENERACIÓN_SESSIONBEANS_GRUPO1\DOCUMENTACION\Interceptors_Timers_Generación_SessionBeans_Java_Grupo1.docx#_Toc125410770)

# MARCO TEÓRICO

## INTRODUCCIÓN

En el presente informe se recopilará información tanto teórica como ejercicios prácticos relacionados al uso de interceptores, timers y generación de sesión beans, por lo que se abordará tópicos relacionados a los temas principales, de esta manera complementando la información impartida en el video relacionado a la actual práctica. A su vez aplicando todos los conceptos en la parte práctica que se desarrollará posteriormente.

Se manejará los conceptos previamente vistos en el curso, pero a su vez se adjuntarán muchos más, de esta manera enriqueciendo y mejorando los conocimientos teóricos y prácticos para poder desempeñar un papel de correcta manera en futuros proyectos alrededor del entorno de JAVA Enterprise Edition.

## OBJETIVOS

### **OBJETIVO GENERAL**

Comprender la importancia y la utilización de interceptores, timers y generación de sesión beans en el lenguaje de JEE, desempeñando habilidades tanto en la documentación como en el desarrollo de un aplicativo que aborde los temas en cuestión.

### **OBJETIVOS ESPECÍFICOS**

* Realizar un estudio del estado del arte en cuanto a los diversos componentes de IDE Netbeans brindados a partir de Enterprise Java Beans en cuanto a sitios web se refiere.
* Identificar cuáles son las diferencias y las utilidades que están aportas a una aplicación.
* Investigar sobre las posibilidades que ofrece el IDE Netbeans para la implementación de los puntos anteriores.
* Aplicar de manera correcta los elementos que ofrece Session beans para el desarrollo de aplicaciones tanto de escritorio como web.

## APLICACIÓNES WEB EN IDE NETBEANS

Java API for XML Web Services (JAX-WS), JSR 224, es una parte importante de la plataforma Java EE. Como continuación del lanzamiento de la API de Java para RPC basado en XML 1.1 (JAX-RPC), JAX-WS simplifica la tarea de desarrollar servicios web con tecnología Java. Aborda algunos de los problemas de JAX-RPC 1.1 proporcionando soporte para múltiples protocolos como SOAP 1.1, SOAP 1.2, XML, y proporcionando una facilidad para soportar protocolos adicionales junto con HTTP. JAX-WS utiliza JAXB 2.0 para la vinculación de datos y admite personalizaciones para controlar las interfaces de punto final de servicio generadas. Con su soporte para anotaciones, JAX-WS simplifica el desarrollo de servicios web y reduce el tamaño de los archivos JAR en tiempo de ejecución [2].

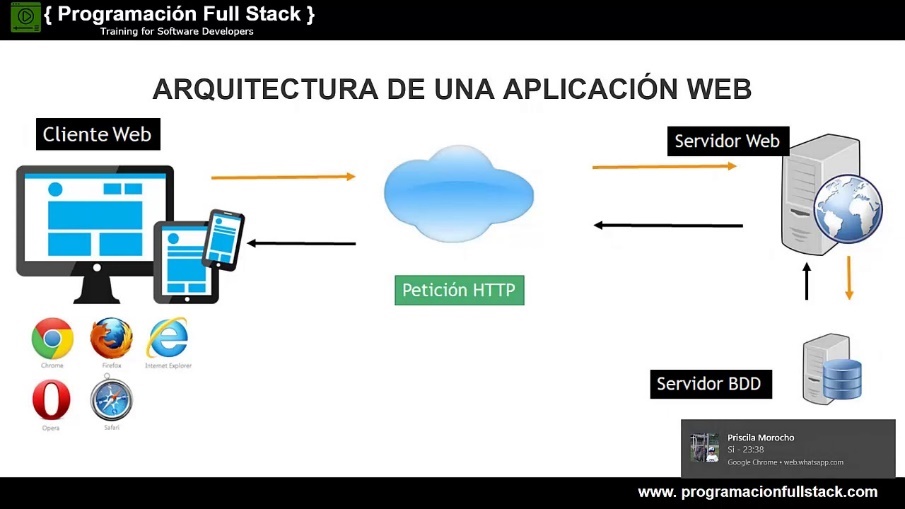


Figura . Arquitectura de aplicación web

### **PAYARA SERVER**

Payara server es un servidor de aplicaciones JAVA de código abierto que es el reemplazo de GlassFish open sourse edition quien carece de soporte por parte de Oracle desde el 2014, payara recibe constantes actualizaciones, cuenta con un amplio soporte por la gran comunidad que lo respalda [3].

Payara server es una plataforma de middleware nativa de nube que brinda a sus aplicaciones seguridad y confiabilidad, es un remplazo directo a GlassFish por ser un derivado del mismo. Payara está respaldado por una comunidad que garantiza ser la mejor opción para el desarrollo de aplicaciones JAVA. Payara server está construida a partir del código fuete de GlassFish para ser su reemplazo directo, esto permite que las empresas migren fácilmente entre las plataformas para aprovechar todo lo que puede ofrecer payara [3].



Figura . Dominio Payara Server

## JAVA WEB

Es una plataforma de programación—parte de la [Plataforma Java](https://es.wikipedia.org/wiki/Plataforma_Java)—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)). Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un [servidor de aplicaciones](https://es.wikipedia.org/wiki/Servidor_de_aplicaciones). La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del [Java Community Process](https://es.wikipedia.org/wiki/Java_Community_Process), Java EE es también considerado informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE [5].

Java EE tiene varias especificaciones de [API](https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones), tales como [JDBC](https://es.wikipedia.org/wiki/JDBC), [RMI](https://es.wikipedia.org/wiki/Java_Remote_Method_Invocation), [e-mail](https://es.wikipedia.org/wiki/JavaMail), [JMS](https://es.wikipedia.org/wiki/JMS), [Servicios Web](https://es.wikipedia.org/wiki/Servicios_Web), [XML](https://es.wikipedia.org/wiki/XML), etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes [5].

Estas incluyen [Enterprise JavaBeans](https://es.wikipedia.org/wiki/Enterprise_JavaBeans), [servlets](https://es.wikipedia.org/wiki/Servlet), [portlets](https://es.wikipedia.org/wiki/Portlet) (siguiendo la especificación de Portlets Java), [JavaServer Pages](https://es.wikipedia.org/wiki/JavaServer_Pages) y varias tecnologías de servicios web. Ello permite al desarrollador crear una [Aplicación de Empresa](https://es.wikipedia.org/w/index.php?title=Aplicaci%C3%B3n_de_Empresa&action=edit&redlink=1) portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel [5].

#### Expresiones Java y Operadores Java

Una expresión es una combinación de variables, operadores y llamadas de métodos construida de acuerdo a la sintaxis del lenguaje que devuelve un valor. El tipo de dato del valor regresado por una expresión depende de los elementos usados en la expresión [6].

Los operadores son palabras claves del lenguaje que permiten la ejecución de operaciones en el contenido de ciertos elementos, en general variables, constantes, valores literales, o retornos de funciones. La combinación de uno o varios operadores y elementos en los cuales los operadores van a apoyarse se llama una expresión. Estas expresiones se evalúan en el momento de la ejecución en función de los operadores y de los valores asociados [6].

Tabla . Operadores usados en Java

|  |  |
| --- | --- |
| Operador | Acción |
| **-** | Valor Negativo |
| **~** | Complemento a uno |
| **++** | Incremento |
| **--** | Decremento |
| **!** | Negación |
| **+** | Suma |
| **-** | Resta |
| **\*** | Multiplicación |
| **/** | División |
| **%** | Módulo (resto entero de la división) |
| **&&** | YY binario |
| **||** | O Binario |
| **^** | O exclusivo |
| **<<** | Desplazamiento hacia la derecha (división por 2) |
| **>>** | Desplazamiento hacia la izquierda (división por 2) |
| **==** | Igualdad |
| **!=** | Desigualdad |
| **<** | Inferior a |
| **>** | Mayor a |
| **<=** | Inferior o igual a |
| **>=** | Mayor o igual a |

*Javax*

La API javax.xml es una colección de APIs de documentos XML que parte del JDK de Sun Microsystems. Consta de las siguientes APIS:

* Java API for XML Processing, o JAXP: permite parsear documentos XML. Ofrece tres interfaces de parseo básicas: DOM, SAX y StAX.
* Java API for XML Messaging, o JAXM: posibilita a las aplicaciones distribuidas comunicarse usando XML y SOAP. Soporta mensajería síncrona y asíncrona.
* Java API for XML-based RPC, o JAX-RPC: permite invocar un servicio web desde una aplicación Java. Puede considerarse como protocolo RMI sobre servicios web.
* Java API for XML Registries, o JAXR: define una API estándar para que las aplicaciones puedan acceder e interactuar con distintos tipos metadatos basados en XML.
* Java Architecture for XML Binding, o JAXB: permite a los desarrolladores exportar clases Java a representaciones en XML.

*Clases utilizadas en Java Web*

Las clases en Java son plantillas para la creación de objetos. Como tal, la clase forma la base para la programación orientada a objetos en Java, la cual es una de los principales paradigmas de desarrollo de software en la actualidad.

Unas clases muy utilizadas en la programación web son los beans de sesión representan sesiones interactivas con uno o más clientes. Los bean de sesión pueden mantener un estado, pero sólo durante el tiempo que el cliente interactúa con el bean. Esto significa que los beans de sesión no almacenan sus datos en una base de datos después de que el cliente termine el proceso.

## JAVABEANS

Los JavaBeans son un modelo de componentes creado por [Sun Microsystems](https://es.wikipedia.org/wiki/Sun_Microsystems) para la construcción de aplicaciones en [Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) [7]. Se usan para encapsular varios objetos en un único objeto (la vaina o [Bean](https://es.wikipedia.org/wiki/Bean_(inform%C3%A1tica)) en inglés), para hacer uso de un solo objeto en lugar de varios más simples [7]. La especificación de JavaBeans de [Sun Microsystems](https://es.wikipedia.org/wiki/Sun_Microsystems) los define como "componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción" [7]. Para funcionar como una [clase](https://es.wikipedia.org/wiki/Clase_(programaci%C3%B3n_orientada_a_objetos)) JavaBean, una clase debe obedecer ciertas convenciones sobre nomenclatura de métodos, construcción y comportamiento [7].

Estas convenciones permiten tener herramientas que puedan utilizar, reutilizar, sustituir y conectar JavaBeans [7].

Las convenciones requeridas son:

* Debe tener un constructor sin argumentos.
* Sus atributos de clase deben ser privados.
* Sus propiedades deben ser accesibles mediante métodos get y set que siguen una convención de nomenclatura estándar.

## MANAGED BEAN

Los managed bean son clases java que se registran en el framework de JSF, lo que permite que puedan ser consumidos desde las páginas dinámicas. Desde la versión 2.0 de JSF este registro se puede llevar a cabo mediante la anotación *@javax.faces.bean.ManagedBean*.

Los managed bean contienen métodos get y set para poder guardar y recoger datos de la sesión y métodos de lógica de negocio, aunque si la aplicación es compleja la lógica de negocio recomiendo que recaiga en componentes EJB [10].

## JAVA TRANSACTION API

Antes de empezar se debe de dejar en claro que JTA solo funciona en ambientes de Java EE, pues es el Application Server quien administra las transacciones a través de JPA. En la siguiente sección se hablará de cómo administrar las transacciones en Ambientes Java SE, por lo que por ahora se dejará hasta aquí. Para configurar JPA para que utilice JPA, se tendrá que regresar al archivo [persistence.xml](https://www.oscarblancarteblog.com/2016/10/26/entitymanager-archivo-persistence-xml/) y ajustar la línea:

<persistence-unit name=”JPA\_PU” transaction-type=”JTA”>

Si bien JTA administrara las transacciones es importante decirle como, es por eso que aquí entra en juego la anotación @TransactionAttribute , la cual pude ser declarada en cada método de negocio o a nivel de clase. Existen 6 posibles valores, los cuales harán que el EJB Container funcione de forma diferente. Los valores son los siguientes:

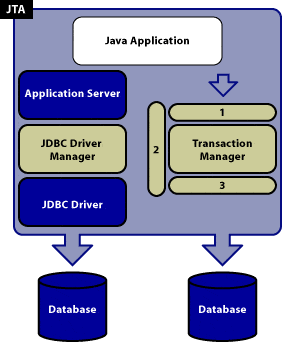


Figura . Representación estructura de un Java Transaction API

## 1.7.1 TRANSACCIONES

Una transacción es un conjunto de operaciones sobre una base de datos que se deben ejecutar como una unidad.

Hay ocasiones en las que es necesario que varias operaciones sobre la base de datos se realicen en bloque, es decir, que se ejecuten o todas o ninguna, pero no que se realicen unas sí y otras no.

Si se ejecutan parcialmente hasta que una da error, el estado de la base de datos puede quedar inconsistente. En este caso se necesitará un mecanismo para devolverla a su estado anterior, pudiendo deshacer todas las operaciones realizadas.

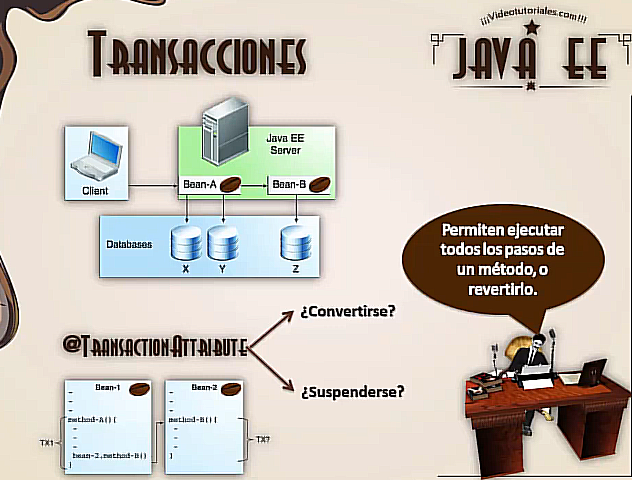


Figura . Presentación del video referente a transacciones

Como ejemplo, se supondrá que se va a reservar un vuelo desde Alicante hasta Sydney. Para llegar a Sydney para realizar varias escalas. Se tendrá que reservar un vuelo de Alicante a Madrid, otro de Madrid a Dubai y otro desde Dubai a Sydney. Si se presenta cada reserva como un insert en la base de datos se tendrá las siguientes instrucciones:

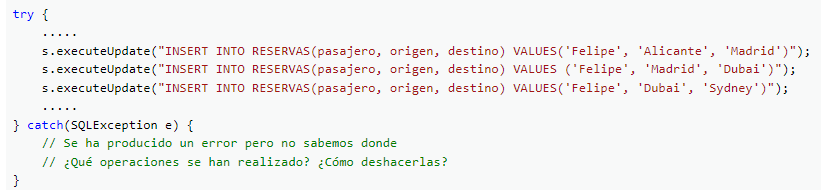


Figura . Ejemplo de codigo empleando transacciones

Tabla . Tabla de valores del atributo transacción

|  |  |  |
| --- | --- | --- |
| @TransactionAttribute value | Método transacción en progreso | Método sin transacción en progreso |
| **TransactionAttributeType MANDATORY** | Se convierte en parte de la existente transacción | Es ejecutado sin ningún soporte de transacción |
| **TransactionAttributeType NEVER** | RemoteException es thrown | Es ejecutado sin ningún soporte de transacción |
| **TransactionAttributeType**  **NOT\_SUPPORTED** | Client transaction esta temporalmente suspendida, se ejecuta sin soporte de transacción, y luego la transacción, y luego la transacción cliente es resumida | Es ejecutado sin ningún soporte de transacción. |
| **TransactionAttributeType**  **REQUIRED** | Se convierte en parte de la existente transacción. | Una nueva transacción es creada por el método |
| **TransactionAttributeType**  **REQUIRES\_NEW** | La transacción cliente es suspendida temporalmente, una nueva transaccipon es creada por el método y luego la transacción cliente es resumida | Una nueva transacción es creada por el método. |
| **TransactionAttributeType**  **SUPPORTS** | El método se convierte en parte de la transacción existente. | Método es ejecutado sin ningún soporte de transacción. |

## 1.7.2 OAP (PROGRAMACIÓN ORIENTADA A ASPECTOS)

La [programación orientada a aspectos](https://en.wikipedia.org/wiki/Aspect-oriented_programming) es una técnica de programación que permite extraer este código transversal y aplicarlo en aquellos puntos de la aplicación donde sea necesario sin estar mezclado con el código al que se aplica. Un aspecto es el código transversal de utilidad aplicable a varios puntos de la aplicación. Esto facilita la legibilidad del código, su mantenimiento y la separación de conceptos.

La programación orientada a aspectos se usa mucho en las aplicaciones que usan [Spring](https://spring.io/) pero hay otras librerías que lo permiten, incluso el propio JDK tiene alguna clase sin necesitar de dependencias adicionales.

La programación define varios términos:

* Aspect: es una funcionalidad genérica aplicable a múltiples objetos. Cada aspecto trata una sola funcionalidad.
* Join point: es el punto de ejecución donde se puede aplicar un aspecto como la llamada a un método, su retorno o el acceso a una propiedad.
* Advice: es la acción que se realiza en un pointcut.
* Pointcut: es una expresión que busca joint points, tiene un advice asociado que se ejecuta en todos los joint points que concuerdan con la expresión.
* weaving: proceso que aplica los aspectos a las clases, puede ser en tiempo de compilación o en tiempo de ejecución.

Esta es una clase normal con un método en la que a modo de ejemplo en la llamada al método se le apliquen dos aspectos, uno para añadir una traza cuando se llame al método y su valor de retorno y otro aspecto para medir cuando tiempo tarda en ejecutarse. La clase Foo desconoce los aspectos que se van a aplicar, no hay que hacer ninguna modificación en ella ni para añadirle los aspectos ni para quitarselos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Ejemplo de codigó

#### INTERCEPTORES

Los interceptores (interceptors) son objetos que son capaces de interponerse en las llamadas a los métodos en los eventos de ciclo de vida de los beans de sesión y de mensaje. Permiten encapsular conductas comunes a distintas partes de la aplicación que normalmente no tienen que ver con la lógica de negocio. Los interceptores son una característica avanzada de la especificación EJB que permite modularizar la aplicación o incluso extender el funcionamiento del contenedor EJB. En esta sesión se verá una introducción a la definición y al funcionamiento de los interceptores para interponerse en las llamadas a los métodos de negocio.

Los interceptores proporcionan un mecanismo para encapsular este tipo de código de una forma sencilla y aplicarlo a los métodos sin interferir directamente. Los interceptores proporcionan una estructura para este tipo de conducta de forma que puedan ser ampliados y extendidos fácilmente en una clase. Por último, proporcionan un mecanismo simple y configurable para aplicar la conducta en el lugar que se requiera.

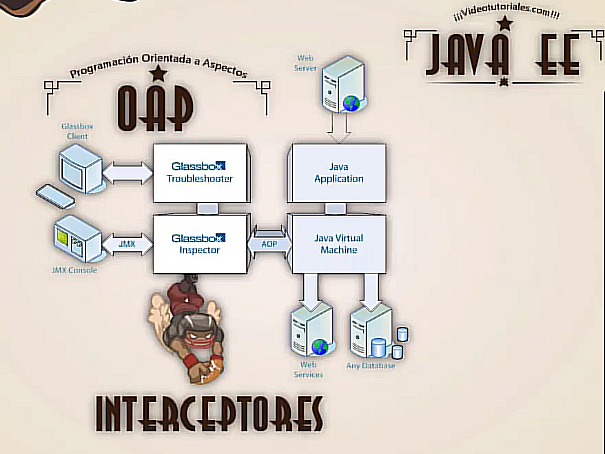


Figura . Presentación del video referente al tema de OAP e interceptores

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Figura . Interceptores explicados dentro del ejercicio práctico

Este tipo de código que envuelve los métodos de la aplicación se suele denominar también aspecto y es la base de una técnica de programación denominada AOP (Aspect Oriented Programming). Un framework muy popular de programación dirigida por aspectos en Java es AspectJ.

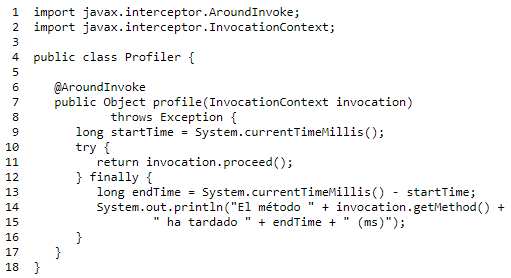


Figura . Ejemplo de código aplicando interceptores

## 1.7.3 ENTERPRISE JAVA BEANS

Las ***Enterprise JavaBeans*** (también conocidas por sus siglas **EJB**) son una de las [interfaces de programación de aplicaciones](https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones) (API) que forman parte del estándar de construcción de aplicaciones empresariales [J2EE](https://es.wikipedia.org/wiki/J2EE) (ahora JEE) de [Oracle Corporation](https://es.wikipedia.org/wiki/Oracle_Corporation) (inicialmente desarrollado por [Sun Microsystems](https://es.wikipedia.org/wiki/Sun_Microsystems)).

Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor, que son precisamente los EJB:

* Comunicación remota utilizando [CORBA](https://es.wikipedia.org/wiki/CORBA).
* Transacciones.
* Control de la concurrencia.
* Eventos utilizando [JMS](https://es.wikipedia.org/wiki/JMS) (*Java Messaging Service*).
* Servicios de nombres y de directorio.
* Seguridad.
* Ubicación de componentes en un [servidor de aplicaciones](https://es.wikipedia.org/wiki/Servidor_de_aplicaciones).

La especificación de EJB define los papeles jugados por el contenedor de EJB y los EJB, además de disponer los EJB en un contenedor.

#### SESSION BEANS STATELESS

Un *Stateless bean* es una clase java sin estado, es decir, no debería tener atributos, sólo métodos a los que se pueda llamar de forma independiente. Para que el contenedor de aplicaciones lo maneje, únicamente hay que ponerle la anotación *@Stateless*, como en el siguiente ejemplo

Texto

Descripción generada automáticamente

Figura . Ejemplo de aplicación de SessionBeans

Es una clase con un método al que se le pasas un texto y lo saca por pantalla. La anotación *@Stateless* es la que lo convierte en un *EJB* y le indica al contenedor de aplicaciones que debe encargarse de manejarlo. La anotación *@LocalBean* indica que a esta clase no se va a acceder de forma remota (desde fuera de la aplicación). La anotación es opcional, puesto que por defecto se entiende que el *bean* no va a ser accedido de forma remota.

¿Qué quiere decir que lo manejará el contenedor de aplicaciones? En el caso de *Stateless beann*, el contenedor de aplicaciones creará varias instancias de estas clases (se encargará de hacer los *new*) y le pasará estas instancias a quien las necesite. Como son clases sin estado, el contenedor dará cualquiera de las instancias que tiene al que la necesite y no se preocupará de entregar siempre la misma al mismo. Se verá más adelante, con la inyección de dependencias, como se entrega una instancia a otra clase que la necesite.

#### TIMERS

Al programar es posible que se verá con la necesidad de realizar una tarea cada cierto tiempo. Por ejemplo, si se quiere pintar un reloj en pantalla, cada segundo se deberá mover o repintar el segundero. Quizás se quiera ver cuándo se crea un fichero, con lo que se puede, por ejemplo, cada diez segundos ver si existe. En fin, hay un montón de posibles aplicaciones en las que se puede necesitar realizar tareas periódicamente.

Diagrama, Esquemático

Descripción generada automáticamente

Figura . Presentación del video referente al tema de timers

Java proporciona dos clases distintas que ayudan a esta tarea. Son las clases **javax.swing.Timer** y **java.util.Timer**. A ambas clases les indica cada cuánto se quiere el aviso (por ejemplo, un aviso cada segundo en el caso del reloj) y ellas se encargan de llamar a un método que se haya implementado. El resultado es que ese método (el que pinta el segundero en la pantalla), se llamará cada cierto tiempo (una vez por segundo en el caso del reloj).

**javax.swing.Timer**

Esta clase es más sencilla de usar.  Basta con instanciarla pasándole cada cuánto tiempo (en milisegundos) se quiere que avise y un **ActionListener**, cuyo método **actionPerformed()** se ejecutará periódicamente. Luego sólo hay que llamar al método **start()** cuando se desee que el **Timer** empiece a contar.  El código puede ser similar a este:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Ejemplo de aplicación de javax.swing.Timer

**java.util.Timer**

Esta clase es más general, tiene más opciones, pero es algo más compleja de usar. Para arrancarlo hay que llamar a alguno de los métodos **schedule()**. No voy a detallarlos todos porque para eso está la API, pero dan bastantes posibilidades. Se puede decirle cosas complejas, como, por ejemplo, que "a partir del 22 de febrero de 2007, a las 17:00 empieza a darme avisos cada 5 minutos", o bien se podrá decirle cosas sencillas, como "avísame una sola vez dentro de 5 minutos".

Dentro de los avisos repetitivos hay incluso dos opciones.

Con los métodos **schedule()**, si por ejemplo, el aviso es repetitivo cada segundo, es posible que el ordenador esté bastante ocupado haciendo otras cosas, con lo que el aviso puede llegar con un cierto retraso. Con esta opción el retraso se acumula de una llamada a otra. Si el ordenador está muy atareado y da avisos cada 1.1 segundos en vez de cada 1, el primer aviso lo recibirá en el segundo 1.1, el segundo en el 2.2, el tercero en el 3.3, etc, etc. Si que el reloj de esta forma, cogerá adelantos o retrasos importantes en poco tiempo.

La otra opción son los métodos **scheduleAtFixedRate()**. Con estos métodos los avisos son relativos al primer aviso, de esta forma, si hay retraso en un aviso, no influye en cuando se produce el siguiente. Igual que antes, si el ordenador está muy ocupado y da avisos cada 1.1 segundos en vez de cada segundo, el primer aviso se recibirá en el segundo 1.1, el segundo en el 2.1, el tercero en el 3.1, etc. El retraso no se va acumulando. Está claro que para hacer un reloj, como es el ejemplo, deberá usar la segunda forma (métodos **scheduleAtFixedRate()** en vez de **schedule()**).

Para el ejemplo del reloj, se crea el **Timer** instanciándolo con un constructor sin parámetros. Luego se llamará al método **scheduleAtFixedRate()**. A este método hay que pasarle tres parámetros:

Una clase **TimerTask**. Dicha clase tiene un método **run()** (Puesto que **TimerTask** es una clase abstracta, deberá hacer una clase hija de ella e implementar el método **run()**) y que es al que se llamará cada cierto tiempo. Ahí deberá poner el código.

Se usará la versión del método en el que el segundo parámetro es un **long**, que es a partir de cuántos milisegundos se quiere que empiece a avisar. Si se pone cero, empezará a avisar inmediatamente.

Como tercer parámetro lleva otro **long**, que es cada cuántos milisegundos se quiere recibir el aviso.

El código de ejemplo para el reloj puede ser como este

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Ejemplo de aplicacion de java.util.Timer

## 1.7.4 ENTIDADES JPA PERSISTENTES

Como cualquier objeto Java, una entidad tiene una identidad que lo distingue de otros objetos, pero cuando existen en el almacén de datos también tiene una identidad persistente. La identidad persistente, definida por el identificador (id) de la entidad, es la clave única que identifica a una instancia entidad y la diferencia de otras instancias del mismo tipo de entidad.

Una entidad tiene una identidad persistente cuando existe una representación de ella en el almacén de datos, esto es, una fila en una tabla de una base de datos. Si no está en la base de datos, entonces incluso aunque en memoria tenga su campo identidad definido, no tienen una identidad persistente. El identificador de la entidad es, por tanto, equivalente a la clave primaria de la tabla de la base de datos en la que se almacena la entidad.

#### DAO (DATA ACCESS OBJECT)

En [software](https://es.wikipedia.org/wiki/Software) de computadores, un **objeto de acceso a datos** (en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s), *data access object*, abreviado **DAO**) es un componente de software que suministra una [interfaz](https://es.wikipedia.org/wiki/Interfaz) común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una [Base de datos](https://es.wikipedia.org/wiki/Base_de_datos) o un [archivo](https://es.wikipedia.org/wiki/Archivo_(inform%C3%A1tica)). El término se aplica frecuentemente al [Patrón de diseño](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o) Object.

**VENTAJAS**

Los Objetos de Acceso a Datos son un Patrón de los subordinados de Diseño Core [J2EE](https://es.wikipedia.org/wiki/J2EE) y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier [objeto de negocio](https://es.wikipedia.org/wiki/Objeto_de_negocio) (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

Los Objetos de Acceso a Datos pueden usarse en [Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) para aislar a una aplicación de la tecnología de persistencia Java subyacente (API de Persistencia Java), la cual podría ser [JDBC](https://es.wikipedia.org/wiki/JDBC), [JDO](https://es.wikipedia.org/w/index.php?title=JDO&action=edit&redlink=1), [Enterprise JavaBeans](https://es.wikipedia.org/wiki/Enterprise_JavaBeans), [TopLink](https://es.wikipedia.org/wiki/TopLink), [EclipseLink](https://es.wikipedia.org/w/index.php?title=EclipseLink&action=edit&redlink=1), [Hibernate](https://es.wikipedia.org/wiki/Hibernate_(Java)), [iBATIS](https://es.wikipedia.org/wiki/IBATIS), o cualquier otra tecnología de persistencia. Usar Objetos de Acceso de Datos significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

**DESVENTAJAS**

La flexibilidad tiene un precio. Cuando se añaden DAOs a una aplicación, la complejidad adicional de usar otra capa de persistencia incrementa la cantidad de código ejecutado durante tiempo de ejecución. La configuración de las capas de persistencia requiere en la mayoría de los casos mucho trabajo.

Las aplicaciones críticas con el rendimiento no deberían usar DAOs.

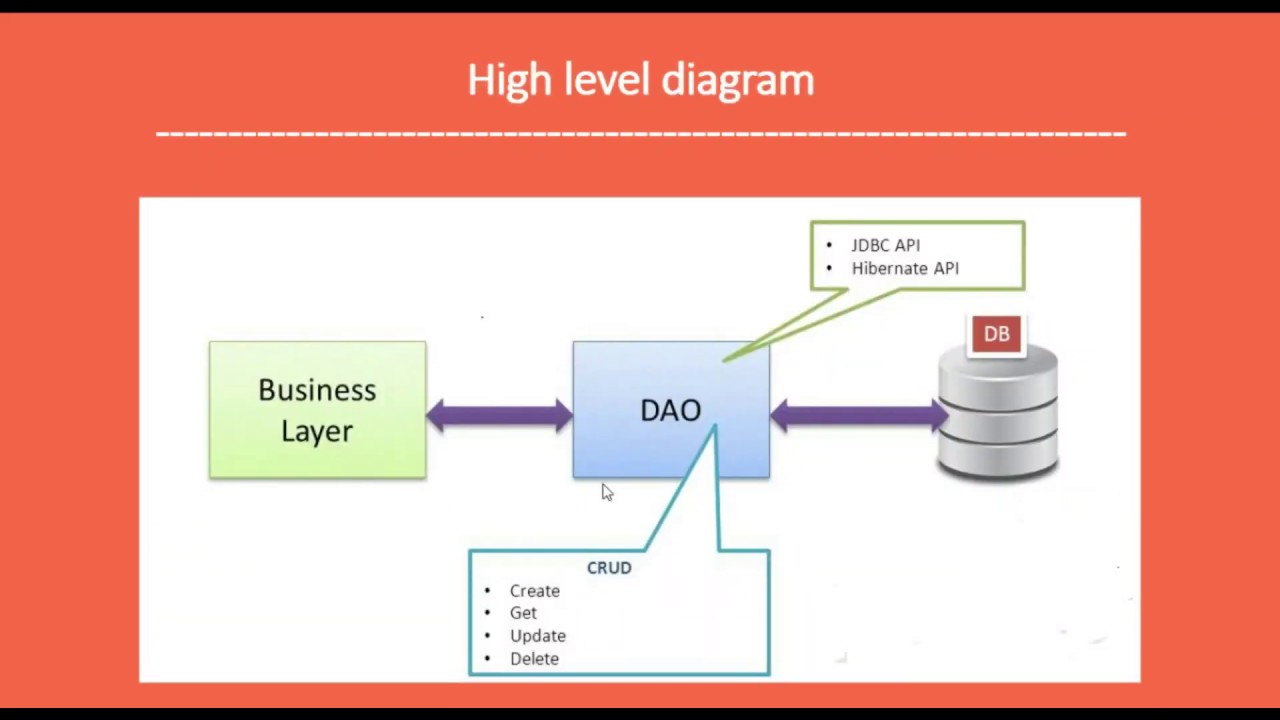


Figura .Estructura referente a DAOs

## PALETA DE COMPONENTES

La paleta de componentes de NetBeans es una forma más práctica de poder agregar componentes para realizar tus diferentes interfaces, en este caso se usará componentes para realizar un respectivo formulario por lo que se mostrará no solo los elementos usados si no también se hará un repaso completo de lo que ofrece esta paleta de componentes en su totalidad [13].

Pero primero se dará un breve instructivo de como habilitar la paleta ya que por defecto no viene activada dentro del IDE NetBeans.

#### HABILITAR LA PALETA DE COMPONENTES

Existen dos formas de poder habilitar la paleta una de ellas es presionar las siguientes teclas seguidas Crtl+Shift+8 así con ese comando de teclas se podrá activar la paleta, si no se usa el comando anteriormente mencionado se deberá seguir los siguientes pasos:

PASO 1.

Una vez abierto el IDE NetBeans se dirige a la pestaña de Window y la selecciona.



Figura . Pantalla de inicio del IDE NetBeans

PASO 2.

Una vez seleccionada la pestaña Window se desplegará un menú de opciones, se ira hasta abajo y seleccionará la opción de “IDE Tools”.

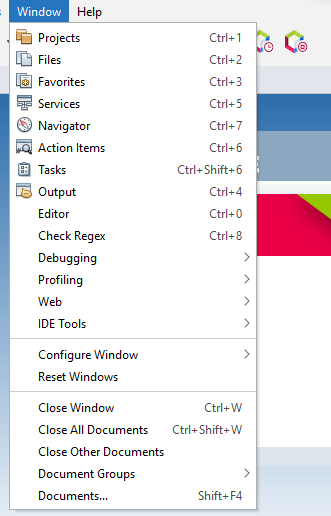


Figura . Menú desplegable

PASO 3.

Una vez seleccionada la opción de IDE Tools se desplegará un submenú en el cual seleccionará la opción de “Palette”.

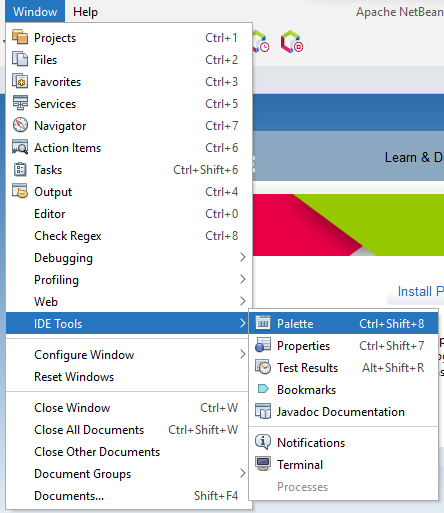


Figura . Submenú de la opcion IDE Tools

Una vez realizado todos los pasos demostrados se mostrará la paleta de la siguiente manera en la pantalla.

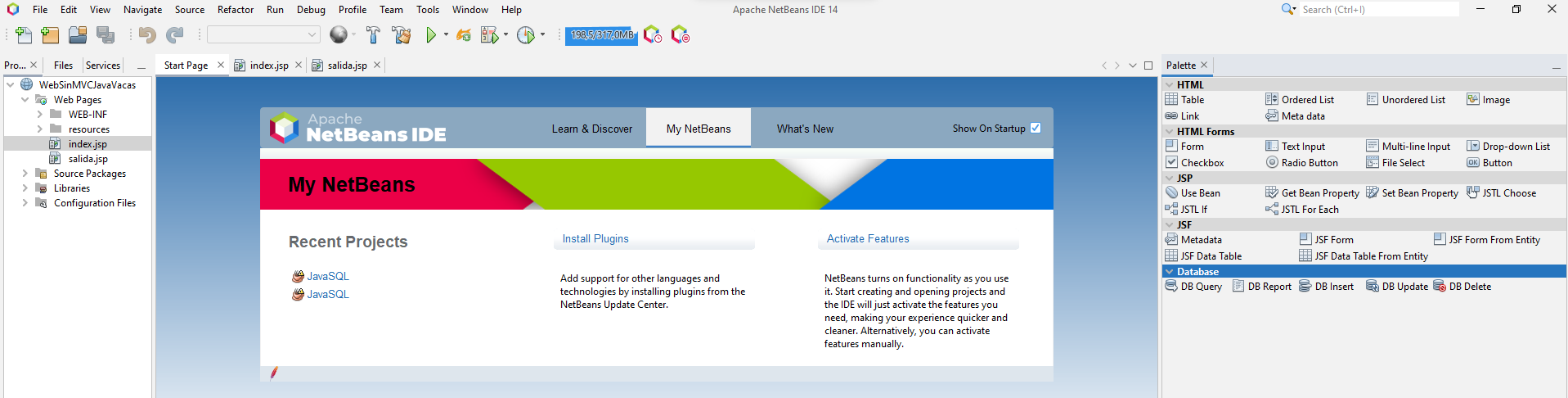


Figura . PAntalla de inicio de paleta del IDE

**ELEMENTOS DE LA PALETA DE COMPONENTES**

La paleta se divide en 5 diferentes pestañas eso quiere decir que en cada una de estas se encontrará elementos o componentes que tienen diferentes funcionalidades que se puede usar para realizar diversas tareas y desarrollar aplicaciones.

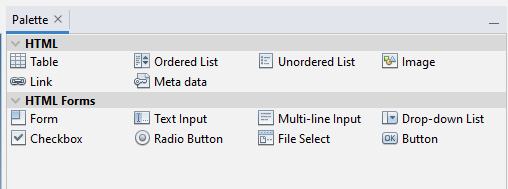


Figura . Paleta de componentes de NetBeans

Se explicarán los componentes utilizados en el desarrollo de la práctica del formulario del presente proyecto, detallando su estructura y su sintaxis para su utilización.

1. Etiqueta Form: es la que contendrá todos los componentes necesarios para que el formulario funcione correctamente, en este caso se deberá elegir el archivo con el que interactuara cuando se realice el formulario por medio del action.

<form action="salida.jsp"></form>

1. Etiqueta table: esta brindará la estructura de una tabla para poder organizar los datos o en este caso los campos que tendrá en el formulario.

<table border="1">

<thead>

</thead>

<tbody>

</tbody>

</table>

1. Etiqueta Text Input: es un cuadro de texto en el cual se podrá ingresar cualquier tipo de texto sea éste letras o números, el uso que se le de dependerá del uso que se le dará.

<input type="text" name="txtNombre" value="" />

1. Etiqueta CheckBox: por medio de esta se podrá colocar cuadros de selección y así poder tener varias opciones a escoger de igual manera será muy útil dependiendo del uso que se le de.

<input type="checkbox" name="chkProgLeng" value="Java" />

1. Etiqueta File: por medio de esta podrá cargar diferente tipo de archivos en el formulario, para este caso se carga un archivo de imagen a su vez se puede dar las dimensiones que tendrá la imagen cuando sea cargada.

<input type="file" name="Foto" value="" width="250">

1. Etiqueta Button: permitirá generar un botón el cual es usado para enviar la información que se llene por medio del formulario, pero no solo sirve para realizar envío de información de igual manera por medio del botón se pueden hacer varias acciones o eventos.

<input type="submit" name="btnEnviar" value="Enviar">

## ETIQUETAS HTML STANDARD

En la programación web se usan principalmente etiquetas HTML, un lenguaje informático de hipertexto que permite utilizar estructuras de párrafos, enlaces, elementos, etiquetas y atributos. Actualmente existen alrededor de 142 etiquetas HTML disponibles, sin embargo, a continuación, se presentarán las etiquetas standard utilizadas regularmente para la creación de sitios web [14].

Tabla . Etiquetas HTML Standard

|  |  |
| --- | --- |
| Operador | Acción |
| <html> | Define el inicio del documento HTML, indica al navegador que lo que viene a continuación debe ser interpretado como código HTML. |
| <script> | Incrusta un script en una web a partir de una src. |
| <head> | Cabecera del documento HTML, contiene información sobre lo que tratará el documento. |
| <title> | Define el título de la página, aparece en la ventana del browser. |
| <link> | Vincula el sitio web con hojas de estilo o íconos. |
| <style> | Colecta el estilo interno de la página a partir de CSS. |
| <body> | Define el contenido principal o el cuerpo del documento que se mostrará en el navegador. |
| <h1> a <h6> | Encabezados del documento, únicamente puede existir un h1, sin embargo, de los otros tipos si pueden existir más usos. |
| <table> | Define una tabla en el documento HTML. |
| <tr> | Define la fila de una tabla creada en el documento HTML. |
| <td> | Corresponde a la celda de una fila de la tabla creada en el documento HTML. |
| <a> | Hipervínculo o enlace de redirección fuera o dentro del mismo sitio web. |
| <div> | División de la página. |
| <img> | Indica la ruta de especificación de una imagen, viene acompañada de src. |
| <li> <ol> <ul> | Etiquetas utilizadas para crear listas en el sitio web. |
| <strong> | Texto en negrita |
| <em> | Texto en cursiva |
| <del> | Texto tachado |

## PATRONES DE DISEÑO

Los patrones de diseño o design patterns, son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error [16].

Los patrones de diseño más utilizados se clasifican en tres categorías principales que son:

* Patrones creacionales
* Patrones estructurales

#### Patrones de comportamiento

##### Modelo-Vista-Controlador (MVC)

MVC (Modelo-Vista-Controlador) es un patrón en el diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocios y su visualización. Esta "separación de preocupaciones" proporciona una mejor división del trabajo y una mejora de mantenimiento. Algunos otros patrones de diseño se basan en MVC, como MVVM (Modelo-Vista-modelo de vista), MVP (Modelo-Vista-Presentador) y MVW (Modelo-Vista-Whatever) [17].

Las tres partes del patrón de diseño de software MVC se pueden describir de la siguiente manera [18]:

1. Modelo: Maneja datos y lógica de negocios.
2. Vista: Se encarga del diseño y presentación.
3. Controlador: Enruta comandos a los modelos y vistas.

# PARTE PRÁCTICA

## CREACIÓN DEL PROYECTO

Para crear el siguiente proyecto se deberá de ir al menú inicial la opción de FILE y luego seleccionar New Project

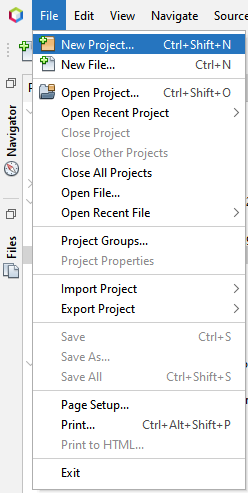


Figura . Opciones del menú File

Posteriormente se seleccionará la opción de Java Enterprise y Enterprise Aplicaction.

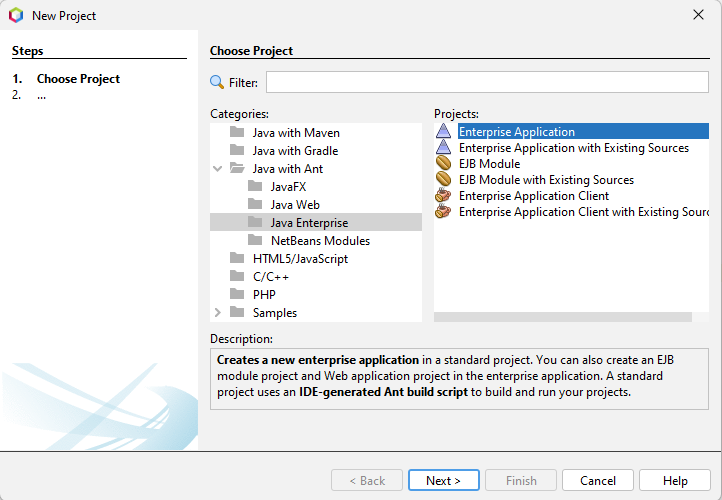


Figura . Pantalla para selección del tipo de proyecto a crear

Se colocará el nombre con el cual se identificará el aplicativo, luego seleccionará la versión de JEE la cual se recomienda de la 7 en adelante y dejar las cajas de selección marcadas como se muestra en la imagen.

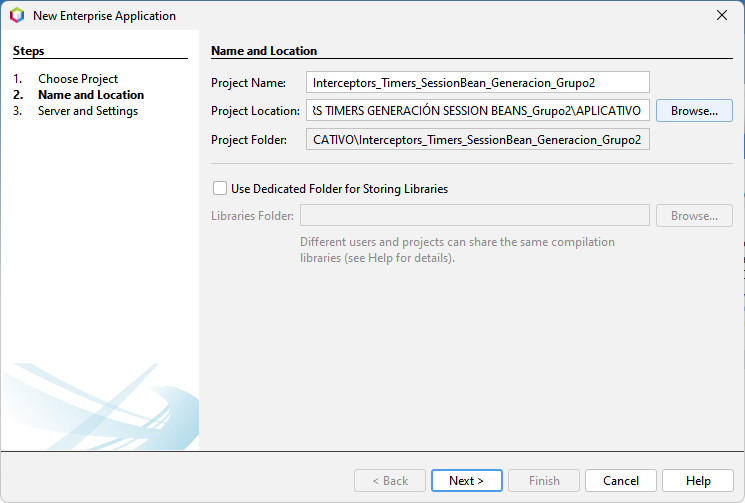


Figura . Colocar el nombre del proyecto principal

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Ventana donde se seleccionará la versión de JEE y el servidor

Como siguiente paso se creará un proyecto de Enterprise Application Client como se observa en la siguiente imagen.

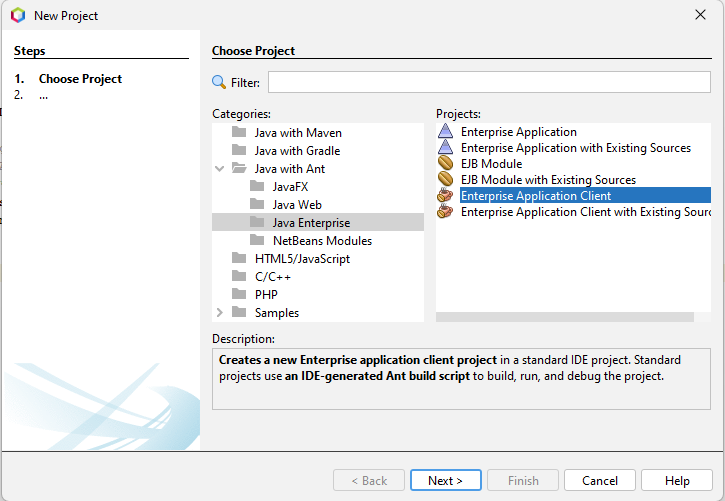


Figura . Selección del proyecto tipo Client

Se colocará el nombre con el cual se identificará el aplicativo, luego seleccionará la versión de JEE la cual se recomienda de la 7 en adelante y dejar las cajas de selección marcadas como se muestra en la imagen, cabe recalcar que en la siguiente pantalla aparecerá el nombre del proyecto que se creó con anterioridad y se seleccionará ya que así ambos proyectos estarán vinculados.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Selección de la versión del JEE y adjuntando al proyecto principal

Una vez creado todos los proyectos y archivos que se generan se obtendrá una estructura de proyecto como la siguiente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura . Estructura final del proyecto

### **CODIFICACIÓN DEL ARCHIVO REPETIR.JAVA**

Continuando con la codificación realizada en el informe anterior, se procederá a crear el archivo repetir dentro del último proyecto de los 4 previamente mencionados el que posee la terminología de ejb, por lo que se hará click derecho sobre el proyecto, se selecciona New y posteriormente se escoge la opción de Session Bean como se muestra en la siguiente imagen.

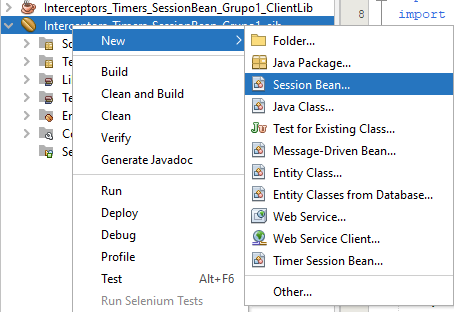


Figura . Selección del archivo tipo Session Beans

Para continuar se pondrá un nombre al archivo el cual será Repetir y se colocará a su vez el nombre del paquete en el cual se ubicará, cabe recalcar que las opciones a elegir serán las marcadas como se muestra en la siguiente imagen, y por último se deberá de seleccionar la librería que se creó al inicio del proyecto para continuar con el desarrollo de la práctica.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Asignar nombre al archivo Session Bean y seleccionando las opciones del archivo

Una vez que se generó el archivo lo siguiente es colocar el siguiente trozo de código e insertar un código de la siguiente manera:

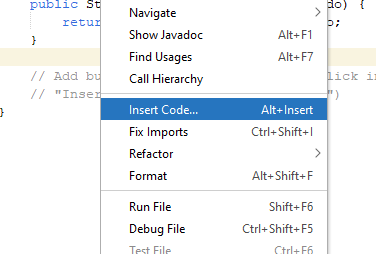


Figura . Menú de opciones del archivo Session Beans

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Ventana de códigos a seleccionar

Una vez que se ha seguido estos pasos se dirigirá a la siguiente pantalla en la cual se tendrá que colocar información del método que se creará y lo que devolverá por lo que se verá de la siguiente manera al final:

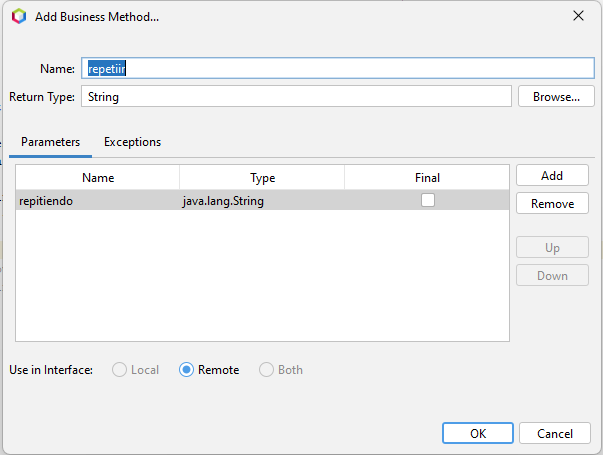


Figura . Ventana en la cual se añadirá el método a usar en el programa

Al final el archivo completo se visualizará como en la siguiente imagen:

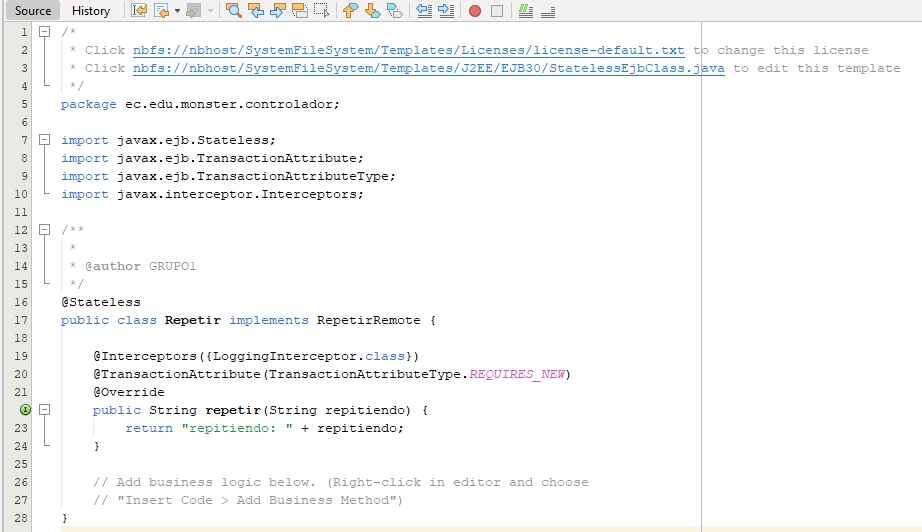


Figura . Vista del código del archivo Repetir.java

Tabla . Código del archivo Repetir.java

package ec.edu.monster.controlador;

import javax.ejb.Stateless;

import javax.ejb.TransactionAttribute;

import javax.ejb.TransactionAttributeType;

import javax.interceptor.Interceptors;

@Stateless

public class Repetir implements RepetirRemote {

@Interceptors({LoggingInterceptor.class})

@TransactionAttribute(TransactionAttributeType.REQUIRES\_NEW)

@Override

public String repetir(String repitiendo) {

return "repitiendo: " + repitiendo;

}

// Add business logic below. (Right-click in editor and choose

// "Insert Code > Add Business Method")

}

### **CODIFICACIÓN DEL ARCHIVO LOGGINGINTERCEPTOR.JAVA**

Continuando con la codificación realizada en el informe anterior, se procederá a crear el archivo repetir dentro del último proyecto de los 4 previamente mencionados el que posee la terminología de ejb, por lo que se hará click derecho sobre el proyecto, se selecciona New y posteriormente se escoge la opción de Session Bean como se muestra en la siguiente imagen.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Selección del tipo de archivo Session Beans

Para continuar se pondrá un nombre al archivo el cual será Repetir y se colocará a su vez el nombre del paquete en el cual se ubicará, cabe recalcar que las opciones a elegir serán las marcadas como se muestra en la siguiente imagen, y por último se deberá de seleccionar la librería que se creó al inicio del proyecto para continuar con el desarrollo de la práctica.

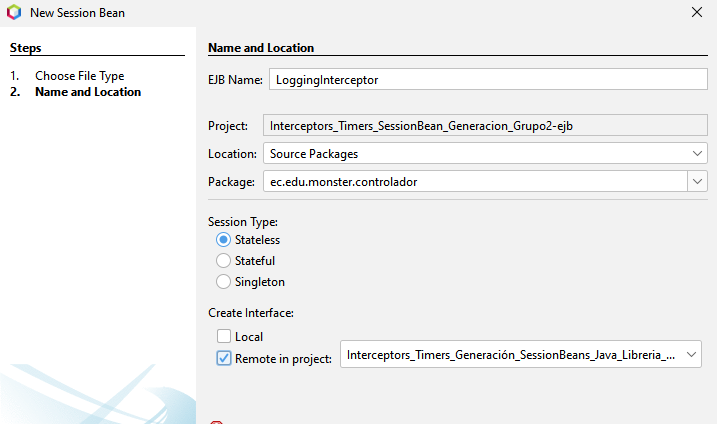


Figura . Asignar nombre al archivo que se creará

Una vez se genere el archivo de LoggingInterceptor.java se colocará el siguiente pedazo de código con el cual se el archivo finalmente se vera de esta manera:

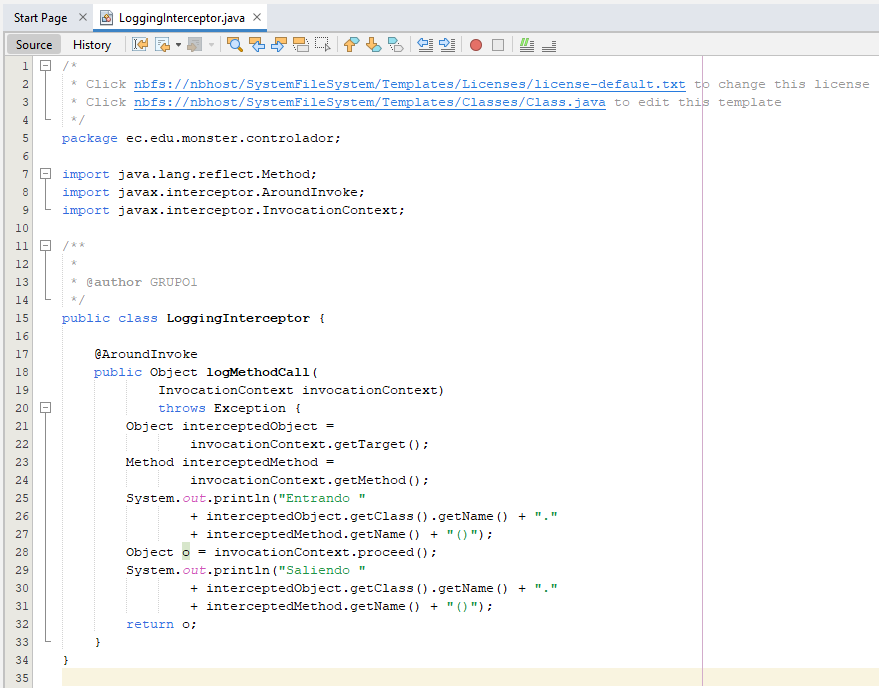


Figura . Vista del archivo resultante LoggingInterceptor.java

Tabla . Código del archivo LoggingInterceptor.java

package ec.edu.monster.controlador;

import java.lang.reflect.Method;

import javax.interceptor.AroundInvoke;

import javax.interceptor.InvocationContext;

public class LoggingInterceptor {

@AroundInvoke

public Object logMethodCall(

InvocationContext invocationContext)

throws Exception {

Object interceptedObject =

invocationContext.getTarget();

Method interceptedMethod =

invocationContext.getMethod();

System.out.println("Entrando "

+ interceptedObject.getClass().getName() + "."

+ interceptedMethod.getName() + "()");

Object o = invocationContext.proceed();

System.out.println("Saliendo "

+ interceptedObject.getClass().getName() + "."

+ interceptedMethod.getName() + "()");

return o;

}

}

## 

## CODIFICACIÓN DEL ARCHIVO EJBTIMERSDEMO.JAVA

Continuando con la codificación realizada en el informe anterior, se procederá a crear el archivo repetir dentro del último proyecto de los 4 previamente mencionados el que posee la terminología de ejb, por lo que se hará click derecho sobre el proyecto, se selecciona New y posteriormente se escoge la opción de Session Bean como se muestra en la siguiente imagen.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Selección del tipo de archivo a crear en el proyecto

Para continuar se pondrá un nombre al archivo el cual será Repetir y se colocará a su vez el nombre del paquete en el cual se ubicará, cabe recalcar que las opciones a elegir serán las marcadas como se muestra en la siguiente imagen, y por último se deberá de seleccionar la librería que se creó al inicio del proyecto para continuar con el desarrollo de la práctica.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Asignar el nombre respectivo al archivo creado

Una vez se genere el archivo de EJBTimersDemo.java se colocará el siguiente pedazo de código con el cual se el archivo finalmente se vera de esta manera:

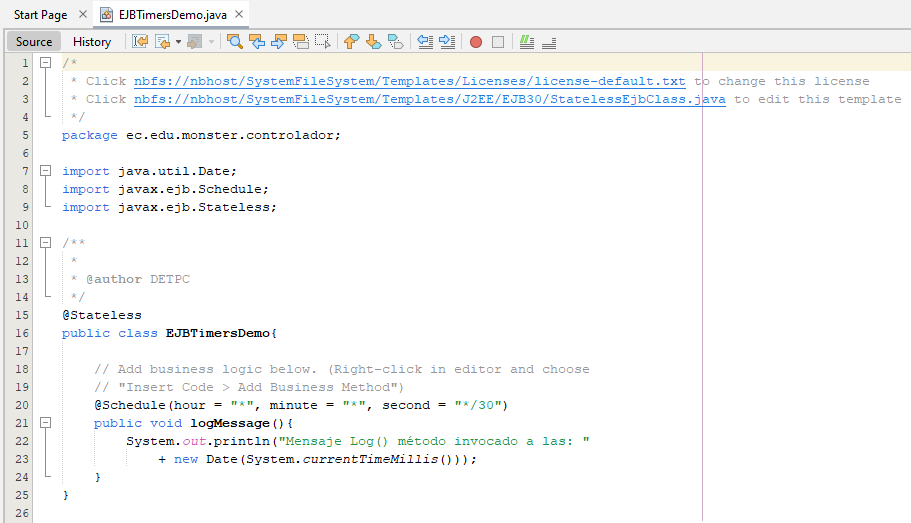


Figura . Vista final del archivo EJBTimersDemo.java

package ec.edu.monster.controlador;

import java.util.Date;

import javax.ejb.Schedule;

import javax.ejb.Stateless;

/\*\*

\*

\* @author GRUPO1

\*/

@Stateless

public class EJBTimersDemo{

// Add business logic below. (Right-click in editor and choose

// "Insert Code > Add Business Method")

@Schedule(hour = "\*", minute = "\*", second = "\*/30")

public void logMessage(){

System.out.println("Mensaje Log() método invocado a las: "

+ new Date(System.currentTimeMillis()));

}

}

Tabla .Código del archivo EJBTimersDemo.java

## CODIFICACIÓN DEL ARCHIVO MAIN.JAVA

El presente archivo se generará automáticamente cuando se creó el archivo de Enterprise Application Client por lo que único que se deberá de hacer es cambiar el nombre del paquete ya que se pone uno predeterminado con el nombre del proyecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Actualización del nombre del paquete creado automáticamente

Una vez realizado este proceso se colocará el siguiente pedazo de código y se con eso se podrá correr el proyecto.

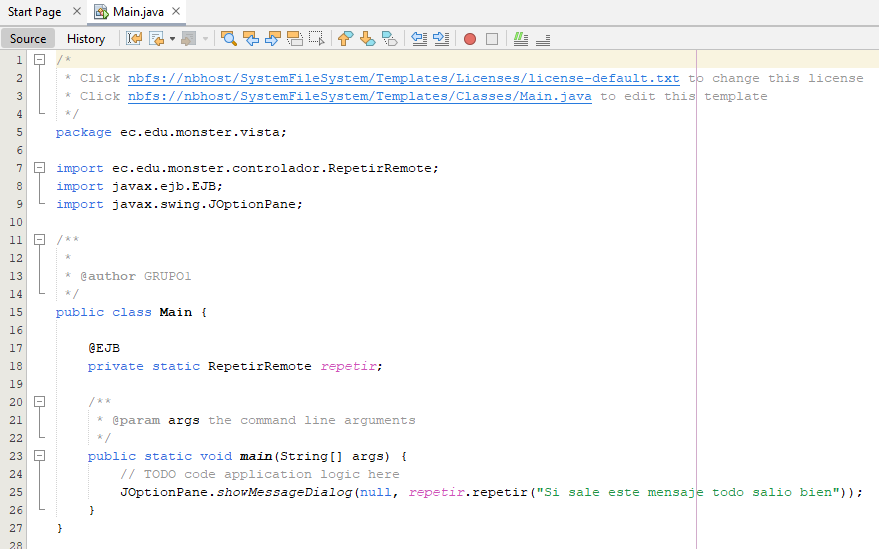


Figura . Vista final del archivo Main.java

package ec.edu.monster.vista;

import ec.edu.monster.controlador.RepetirRemote;

import javax.ejb.EJB;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author GRUPO1

\*/

public class Main {

@EJB

private static RepetirRemote repetir;

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

JOptionPane.showMessageDialog(null, repetir.repetir("Si sale este mensaje todo salio bien"));

}

}

Tabla . Código del archivo Main.java

## DESEMPEÑO DE LA APLICACIÓN

### **INTERCEPTORS\_TIMERS\_SESSIONBEAN\_GRUPO2**

Para la ejecución de la aplicación se tiene que guardar todos los cambios que se han hecho en los archivos respectivos del proyecto y se podrá ejecutar de dos maneras

La primera manera es dando click derecho sobre el paquete y realizar un deploy, una vez hecho el deploy se deberá de hacer click derecho de nuevo sobre el proyecto principal y seleccionar la opción de Run.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Menú de opciones de un proyecto

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Seleccionando la opción de Run ejecutando el proyecto

La segunda forma en la que se puede correr o ejecutar el programa es realizando click sobre el símbolo de play de color verde en el menú de herramientas.



Figura . Opciones del menú de herramientas

Cabe recalcar que se debe de tener el servidor corriendo es decir se deberá de iniciar la payara sever para que se pueda ejecutar la aplicación de manera correcta, por lo que si no se tiene el servidor iniciado el proceso para hacerlo es el siguiente:

Primero se hará click sobre la pestaña de Services, y se deberá de desplegar la pestaña de servers y se dará click derecho sobre Payara Server. Una vez que se haya iniciado en la ventana de output se notificará de que este ya está iniciado para trabajar.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Pestaña de Services

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Menú de opciones del servidor de payara server

Texto

Descripción generada automáticamente

Figura . Output con información de la payara iniciándose

Tabla

Descripción generada automáticamente

Figura . Resultado del inicio del payara server

Para verificar si la payara server se inició adecuadamente se revisará en la pestaña de services y el símbolo de payara se deberá de ver de la siguiente manera.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Vista de como se ve el payara server iniciado

Si se ha verificado todos los pasos anteriores al momento de correr el aplicativo el resultado del proyecto será el siguiente:

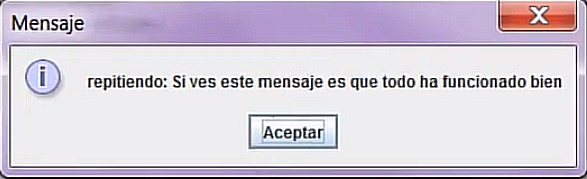


Figura . Ventana resultante del método repetir

Esto demuestra de que el método repetir funciono de manera indicada ya que si no lo hiciera no se mostraría el respectivo mensaje y por lo contrario en la ventana de output informará cual es el error que se produjo en el aplicativo.

Como segundo punto de funcionalidad del aplicativo se tiene el resultado del logMethodCall por lo que mediante consola se mostrará un mensaje de entrando y saliendo, haciendo referencia a la funcionalidad de los interceptores con el nombre del método repetir que es el que se ha ejecutado y mostrando la ventana de mensaje que se indicó con anterioridad.

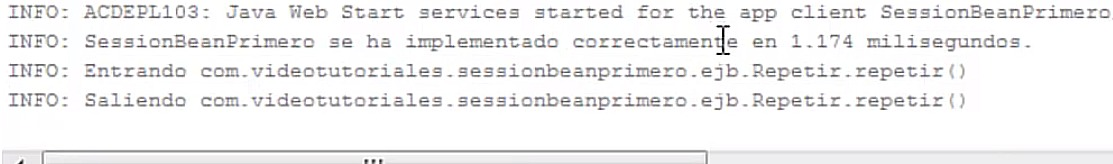


Figura . Resultado de los interceptores por medio de consola

# CONCLUSIONES

* En conclusión, los interceptores, timers y sesión vean muestran que se pueden de usar de diferentes maneras, ofreciendo al programador realizar diferentes tareas o métodos que se ejecutaran dando diferentes resultados por lo que es necesario y muy útil poder implementar todos estos conceptos ya en ejercicios prácticos pero cabe tener en cuenta de que se debe de verificar el versionamiento del JEE con el que se trabajará y que los proyectos siempre deben de estar muy bien relacionados para obtener los resultados esperados.
* Se comprobó la importancia que tiene en una aplicación tener separaciones al momento de querer resolver un problema, estas separaciones se las realizo con éxito mediante el patrón de diseño MVC, separando cada componente en modelo, vista y controlador.

# RECOMENDACIONES

* Se recomienda actualizar a la versión correspondiente cada cierto tiempo. Desde el propio Panel de Control pudiendo habilitar la opción para que se haga de manera automática mediante un mensaje de aviso.
* Se recomienda que se sigan revisando más a profundidad los contenidos de la clase referentes a JAVA Enterprise Edition, de esta manera reforzando conceptos en los cuales hay que mejorar, ya que todos estos conceptos abordados serán de mucha utilidad para futuros proyectos.
* Se recomienda revisar mucho más a profundidad la documentación respectiva a los tópicos vistos y abordados alrededor de la presente práctica.

# REFERENCIAS

[1] "Informática Básica: ¿Qué son las aplicaciones web?" GCFGlobal.org. <https://edu.gcfglobal.org/es/informatica-basica/que-son-las-aplicaciones-web/1/> (accedido el 30 de diciembre de 2022).

[2] "Crear aplicacion web java netbeans - Todo sobre JAVA". Todo sobre JAVA. <https://quejava.com/crear-aplicacion-web-java-netbeans/> (accedido el 30 de diciembre de 2022).

[3] "Home". Payara Services Ltd. [https://www.payara.fish](https://www.payara.fish/) (accedido el 30 de diciembre de 2022).

[4] "Los mejores frameworks de Java en 2022 para desarrollo web". Profile Software Services. <https://profile.es/blog/frameworks-java-desarrollo-web/> (accedido el 30 de diciembre de 2022).

[5] Colaboradores de los proyectos Wikimedia. "Java EE - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=Java_EE&amp;oldid=146032828> (accedido el 7 de noviembre de 2022).

[6] Java. "Expresiones y Operadores en Java - Auribox Training". Auribox Training. https://blog.auriboxtraining.com/java/el-lenguaje-java-expresiones-y-operadores/ (accedido el 11 de noviembre de 2022).

[7] Colaboradores de los proyectos Wikimedia. "JavaBean - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=JavaBean&amp;oldid=125248574> (accedido el 7 de noviembre de 2022).

[8] Colaboradores de los proyectos Wikimedia. "Facelets - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Facelets#:~:text=En%20informática,%20Facelets%20es%20un,de%20entrada%20válidos%20para%20trabajar.](https://es.wikipedia.org/wiki/Facelets#:~:text=En%20inform%C3%A1tica,%20Facelets%20es%20un,de%20entrada%20v%C3%A1lidos%20para%20trabajar.) (accedido el 30 de diciembre de 2022).

[9] "JSF Etiquetas (tags) - jcodepoint". jcodepoint. <https://jcodepoint.com/jsf/jsf-etiquetas/> (accedido el 30 de diciembre de 2022).

[10] "Managed Bean – Tecnicomio". Tecnicomio. <https://tecnicomio.com/tag/managed-bean/#:~:text=Los%20managed%20bean%20son%20clases,faces.> (accedido el 30 de diciembre de 2022).

[11] "J2EE: JSF (9) – Composite Components – inmensia". inmensia – Haiku Coder's Blog. <https://inmensia.com/articulos/j2ee/jsf_components.html> (accedido el 30 de diciembre de 2022).

[12] "Métodos de petición HTTP - HTTP | MDN". MDN Web Docs. <https://developer.mozilla.org/es/docs/Web/HTTP/Methods> (accedido el 30 de diciembre de 2022).

[13] "Agregar nuevos componentes a la paleta de NetBeans - Sitio Web de Javier García Escobedo (javiergarciaescobedo.es)". Inicio - Sitio Web de Javier García Escobedo (javiergarciaescobedo.es). [https://javiergarciaescobedo.es/programacion-en-java/2-clases-y-objetos/23-agregar-nuevos-qbeansq-java-a-la-paleta-de-netbeans#:~:text=La%20paleta%20de%20componentes%20que,,%20áreas%20de%20texto,%20etc.](https://javiergarciaescobedo.es/programacion-en-java/2-clases-y-objetos/23-agregar-nuevos-qbeansq-java-a-la-paleta-de-netbeans#:~:text=La%20paleta%20de%20componentes%20que,,%20%C3%A1reas%20de%20texto,%20etc.) (accedido el 30 de diciembre de 2022).

[14] "HTML: Lenguaje de etiquetas de hipertexto | MDN". MDN Web Docs. <https://developer.mozilla.org/es/docs/Web/HTML> (accedido el 30 de diciembre de 2022).

[15] "JavaServer Faces(JSF) | Marco de Desarrollo de la Junta de Andalucía". Junta de Andalucía - Portal oficial. <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/101> (accedido el 30 de diciembre de 2022).

[16] MDN Web Docs. "MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN". MDN Web Docs. https://developer.mozilla.org/es/docs/Glossary/MVC#:~:text=MVC%20(Modelo-Vista-Controlador,de%20negocios%20y%20su%20visualización. (accedido el 14 de noviembre de 2022).

[17] M. Martínez. "Qué son los Patrones de Diseño de software / Design Patterns". Profile Software Services. https://profile.es/blog/patrones-de-diseno-de-software/ (accedido el 13 de noviembre de 2022).