**TECNOLÓGICO NACIONAL DE MÉXICO**

**Instituto Tecnológico de León**



**INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**PROGRAMACIÓN WEB AVANZADA**

**7:00 – 8:40, martes, jueves, viernes**

**“Introducción a la Programación Web**

**con Java: Ejercicios 11 y 12”**

**PRESENTA:**

**JÉSSICA RAMÍREZ SÁNCHEZ**



**CON LA ASESORÍA DE:**

**LIC. ANGÉLICA MARÍA ORTIZ GAUCIN**

**ÍNDICE**

[INTRODUCCIÓN 3](#_Toc39022407)

[PRÁCTICAS 3](#_Toc39022408)

[Ejercicio 11. Proyecto JSF con acceso a base de datos 3](#_Toc39022409)

[Base de datos 3](#_Toc39022410)

[Creación de proyecto 4](#_Toc39022411)

[Bean Uea 5](#_Toc39022412)

[Conexión a base de datos 5](#_Toc39022413)

[Leer 8](#_Toc39022414)

[Crear 9](#_Toc39022415)

[Borrar 12](#_Toc39022416)

[Modificar 14](#_Toc39022417)

[Resultados 18](#_Toc39022418)

[Ejercicio 12. Validadores y existencia de librerías adicionales 24](#_Toc39022419)

[Campo obligatorio 24](#_Toc39022420)

[Validador de longitud 25](#_Toc39022421)

[Validador de rango 27](#_Toc39022422)

[Métodos validadores 29](#_Toc39022423)

[CONCLUSIÓN 31](#_Toc39022424)

# INTRODUCCIÓN

Realizar un CRUD de registros en una base de datos necesita de definir la lógica de cada una de esas operaciones, agregando sus respectivos modelos usados de acuerdo a las tablas creadas en nuestra base de datos y poder persistir todos nuestros datos.

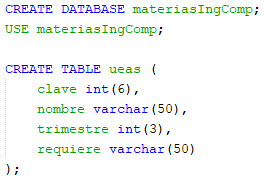
# PRÁCTICAS

## Ejercicio 11. Proyecto JSF con acceso a base de datos

En este ejercicio se creará una aplicación web con el framework JSF y su respectivo acceso a base de datos. Se contará con las aplicaciones básicas: crear, leer, actualizar y borrar registros de la base de datos a crear. Nuestra aplicación se encargará de registrar materias llamadas “UEAS”.

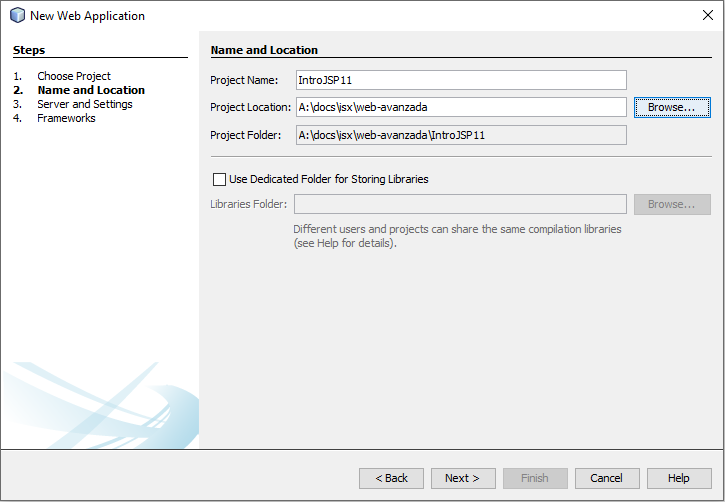
### Base de datos

Empezaremos por crear la base de datos, con los siguientes campos dentro de la tabla ueas: clave, nombre, trimestre, requiere.

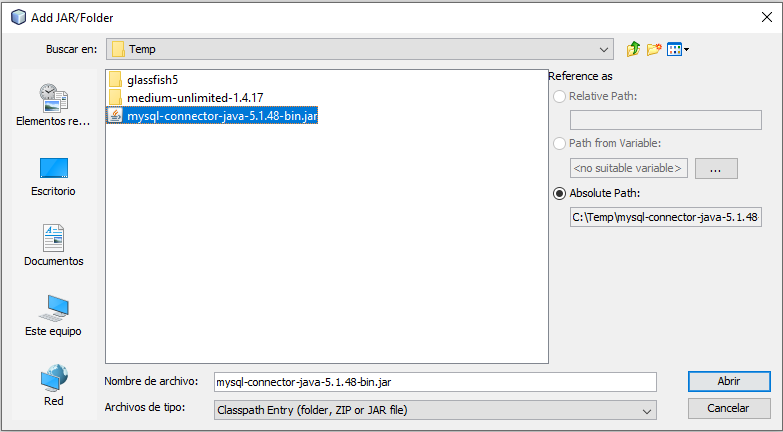


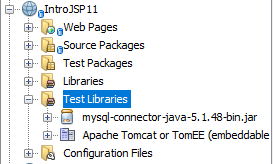
### Creación de proyecto

Crearemos un nuevo proyecto “Web Application” llamado “IntroJSP11”.



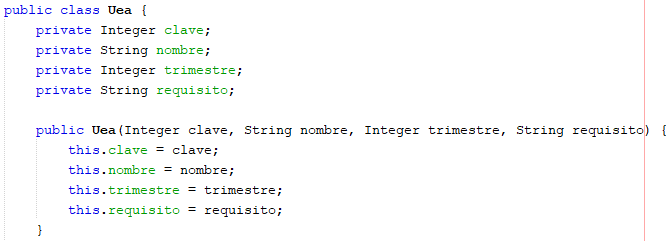
Agregaremos el driver de MySQL a la carpeta de librerías.





### Bean Uea

Dentro de la carpeta de “Source Packages” creamos una carpeta “model” y creamos una clase Java “Uea” que contendrá los atributos: clave, nombre, trimestre, requisito. Agregamos los respectivos setters y getters, además, agregamos un constructor que reciba todos los atributos.



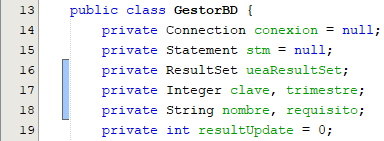
### Conexión a base de datos

Ahora empezaremos a desarrollar la conexión con la base de datos. Tendremos diferentes métodos que nos ayudarán a establecer la conexión con la base de datos que creamos al inicio del ejercicio.

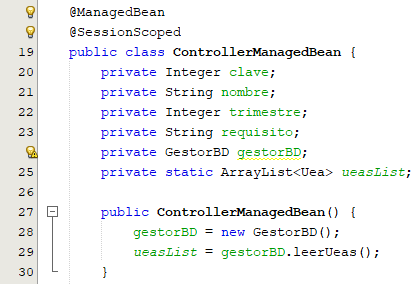
Dentro de nuestro constructor tendremos la lógica para establecer la conexión pasando como parámetros al driver nuestro usuario, contraseña y el url en donde se encuentra nuestra base de datos. Por otra parte, tendremos dos métodos que nos permitirán obtener la conexión o cerrar el flujo de la comunicación.



Dentro del mismo paquete “model” tendremos otra clase Java “GestorBD”, que contendrá la lógica para realizar la comunicación con la base de datos y realizar las operaciones de lectura, escritura, etc.

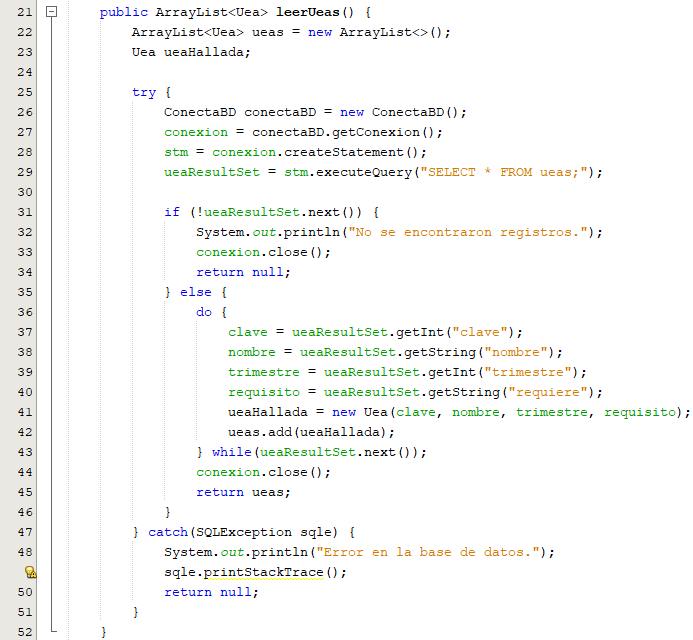


Por último, crearemos un paquete “controller” en el que crearemos una clase Java “ControllerManagedBean”. Esta clase contendrá cierta lógica para utilizar los métodos de “GestorBD”. Iniciaremos por añadir los atributos a la clase: clave, nombre, trimestre, requisito, gestorDB de tipo GestorBD, y una lista ueasList de tipo ArrayList<Uea>, se agregarán los respectivos setters y getters con un constructor de la clase que no reciba ningún parámetro.

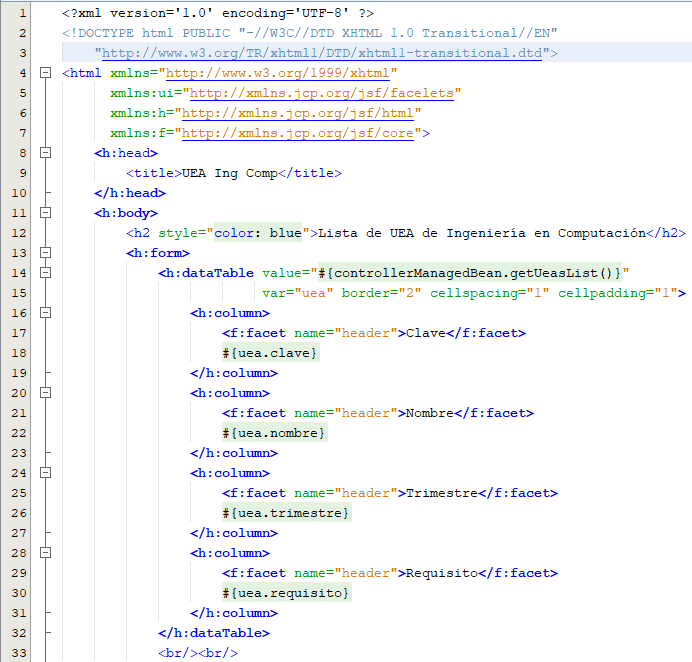


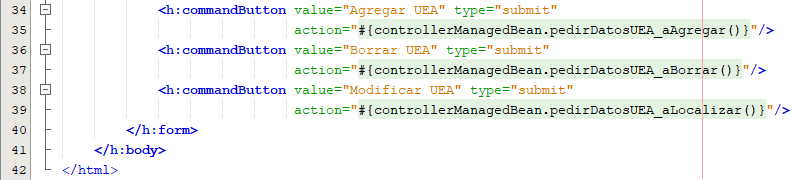
### Leer

Comenzando con la parte de la lectura deberemos leer todas las “ueas” y devolverlas en una lista.



Para mostrar las “ueas” deberemos crear un JSP en la carpeta de “Web Pages” que será nuestro “index.xhtml”. Recordemos que la extensión de nuestro archivo cambia al estar utilizando el framework JSF.





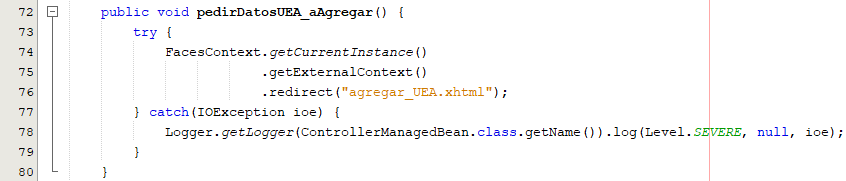
### Crear

En nuestra página index tendremos el botón para agregar un nuevo elemento. Procederemos a desarrollar el método para agregarlo a la base de datos.

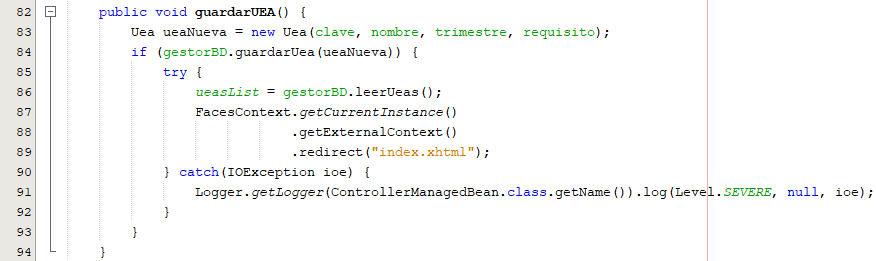
La página que corresponde para dar de alta un nuevo registro se llama “agregar\_UEA.xhtml”.



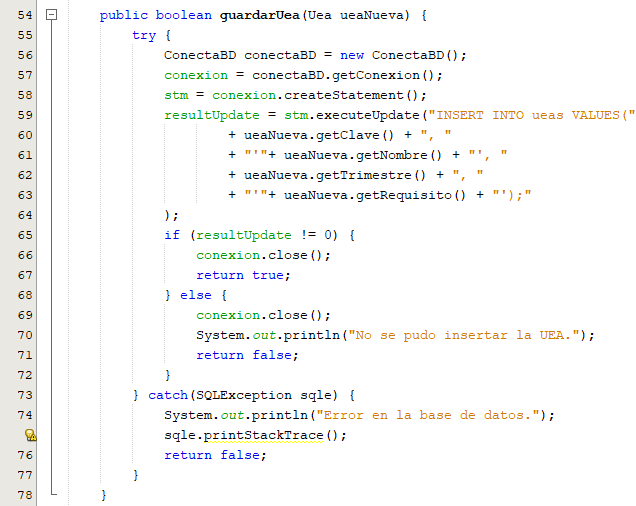
En la clase “ControllerManagedBean” tendremos que pedir los datos que se están enviando desde el formulario.



Dentro de la misma clase controladora, añadiremos el método que nos permitirá guardar los datos.



Por último, en nuestra clase “GestorBD”, correspondiente a nuestro modelo, deberemos añadir el método que nos permita realizar la instrucción “insert”.

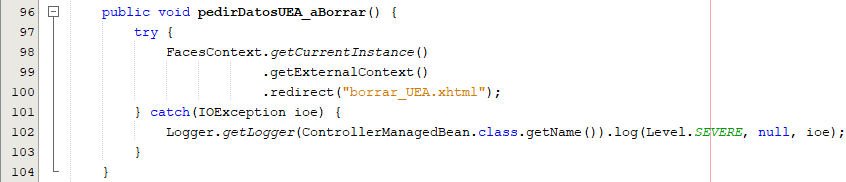


### Borrar

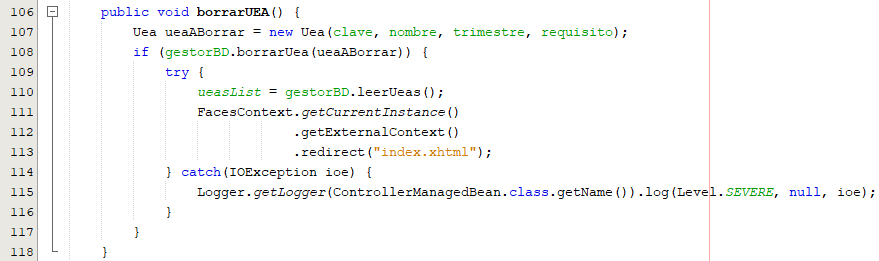
Para eliminar una “uea” crearemos una página “borrar\_UEA.xhtml” en la que pediremos al usuario que introduzca la clave y el nombre de la UEA a eliminar.



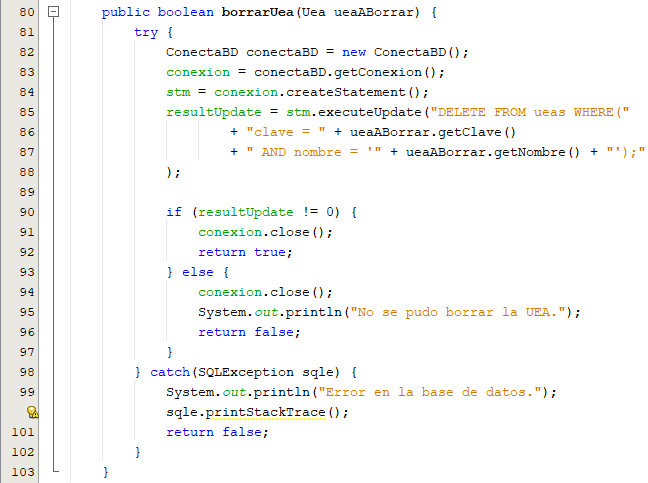
En nuestro controlador tendremos el método que solicita los datos a borrar. Dicho método es llamado cuando el botón de borrar en la página “index” es presionado.



En el mismo controlador, tendremos el método que elimina la “uea”.



En el modelo desarrollaremos la instrucción que elimine la “uea” solicitada.

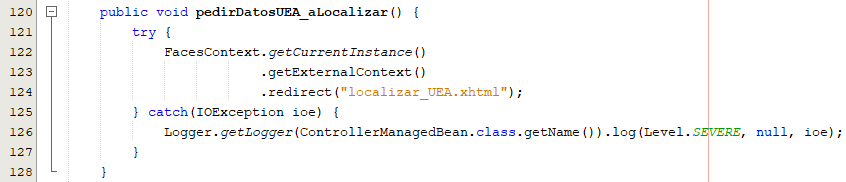


### Modificar

Cuando estemos en la página “index” también tendremos nuestro respectivo botón que nos permitirá modificar cualquier “uea” redirigiéndonos a la página “localizar\_UEA.xhtml”.



En el controlador tendremos los respectivos métodos para realizar la localización de dicha “uea”.

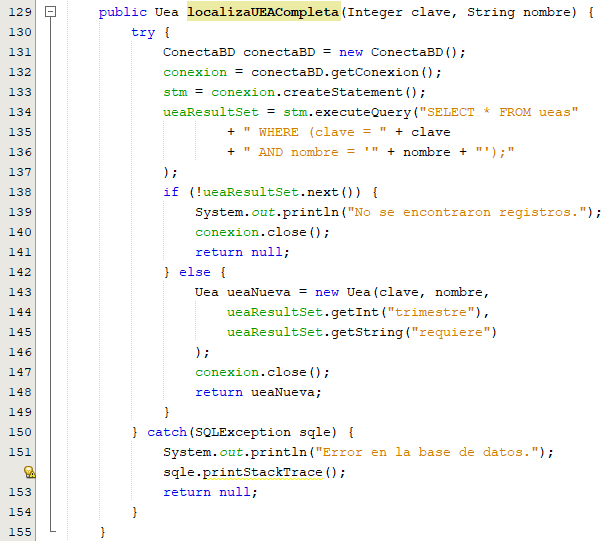




Para evitar errores al no encontrar la “uea” solicitada, nos apoyaremos con una página adicional “error.xhtml” que nos permitirá mostrarle al usuario que los datos introducidos no se encuentran como parte de los registros que hay en la base de datos.



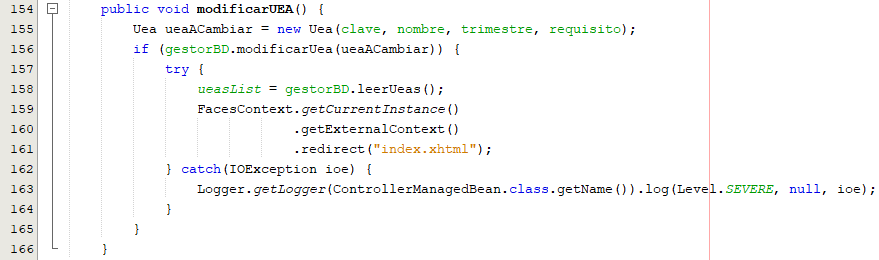
En nuestro modelo tendremos la siguiente instrucción “select” para localizar la “uea” solicitada.



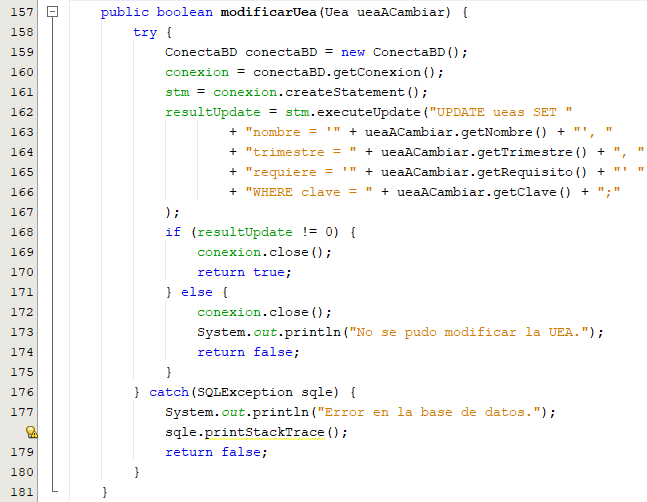
En cuanto los datos son extraídos de la base de datos se redireccionará a la página “modificar.xhtml” mostrando todos los datos de la “uea”.



Cuando seleccionemos el botón de “Enviar” en la vista de modificar los datos, el controlador inmediatamente actualizará los datos de la “uea”.



Y por último, el modelo se encargará de persistir los datos en la base de datos.



### Resultados

Para comprobar que todo funciona correctamente, realizaremos una prueba creando una nueva “uea”.

Clave: 480946

Nombre: “Arquitectura de Aplicaciones Empresariales”

Trimestre: 9

Requiere: “Programación Web”

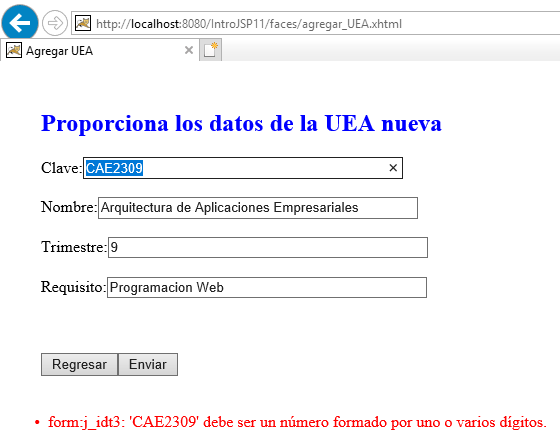
Podemos observar como al iniciar la aplicación se muestran todas las “ueas” que se encuentran registradas en la base de datos.



Procediendo a la creación presionamos el botón “Agregar UEA” y capturamos los datos especificados con anterioridad.



En caso de capturar erróneamente alguno de los campos se especifica el error cometido.



En cuanto se dé clic en el botón “Enviar” los datos se guardarán en la base de datos y la página se redirigirá a la página principal mostrando todos los registros.



Vamos a modificar la “uea” y cambiaremos el requisito por “Desarrollo de Aplicaciones Empresariales”.

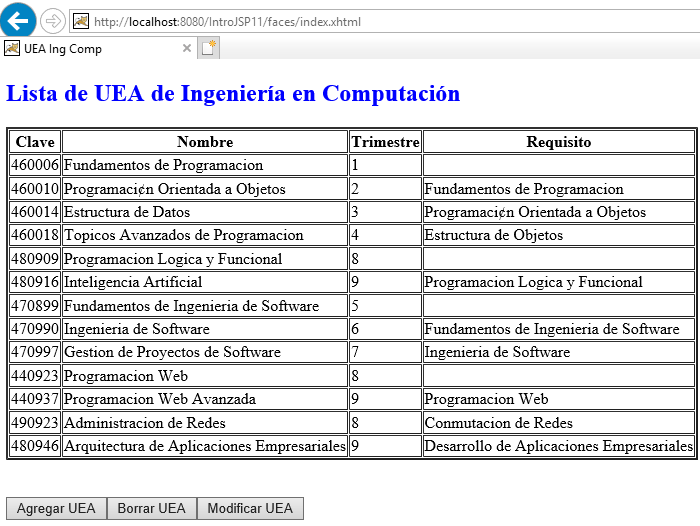
Primero la localizamos.



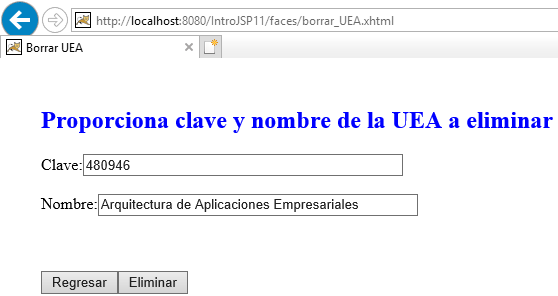
Y ahora realizamos el cambio.



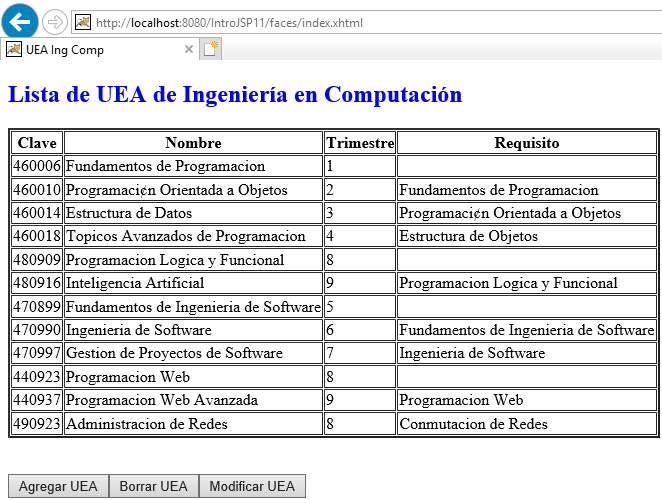
Comprobamos que efectivamente el cambio se realizó con éxito.



Para borrar el registro necesitamos localizarlo.



Compramos que se haya eliminado correctamente y no aparezca en los registros de la base de datos.



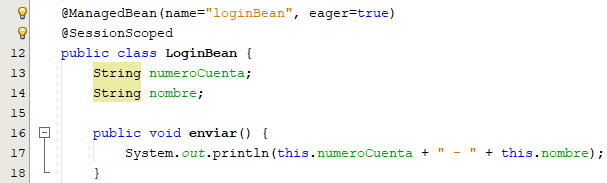
## Ejercicio 12. Validadores y existencia de librerías adicionales

En este ejercicio se trabajará con los validadores que nos ofrece el framework JSF, utilizaremos las librerías adicionales para su ejemplificación.

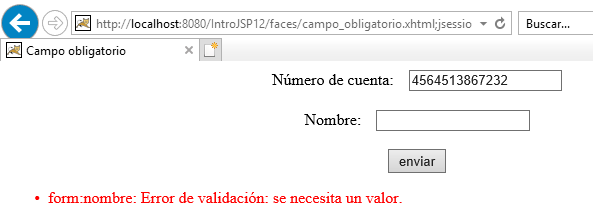
### Campo obligatorio

Cuando un campo es necesario para el procesamiento de los datos del formulario, podemos optar por agregar atributos al input del campo que necesitamos que se introduzca obligatoriamente.

Tendremos un Managed Bean “LoginBean” que tendrá los atributos de “numeroCuenta” y “nombre”, además, el método “enviar” que de momento solo imprimirá ambos atributos.



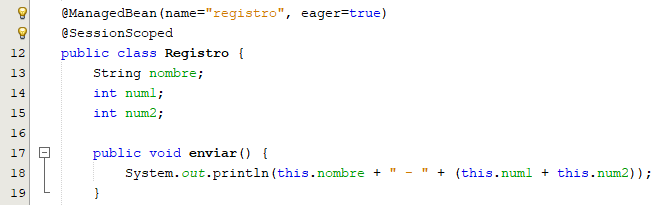
La página del formulario tendrá los campos para capturar “numeroCuenta” y “nombre” ambos obligatorios. Si alguno de los dos no es capturado, mostrará el mensaje de error.



### Validador de longitud

Si necesitamos que un campo no exceda el límite de cierta cantidad de caracteres se debe agregar el atributo a la etiqueta del input para que se bloquee la captura en dicho campo.

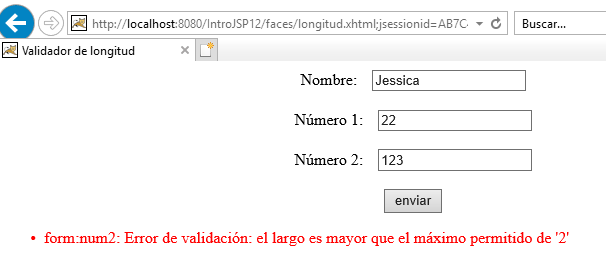
Tendremos un Java Bean “registro”, contendrá los atributos nombre, num1, num2 y el método de enviar.



La página del formulario tendrá los 3 campos para capturar los datos, la diferencia con los campos para los números es que tendrá la etiqueta “f:validateLength” con su atributo “máximum” que se encargarán de limitar la longitud del número.



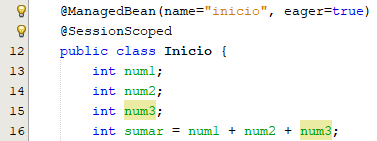
Comprobamos que efectivamente al exceder la cantidad de caracteres establecida nos manda el mensaje de error.



### Validador de rango

En cuestión de campos numéricos podemos limitar las opciones de captura para dicho número estableciendo ciertos rangos de captura.

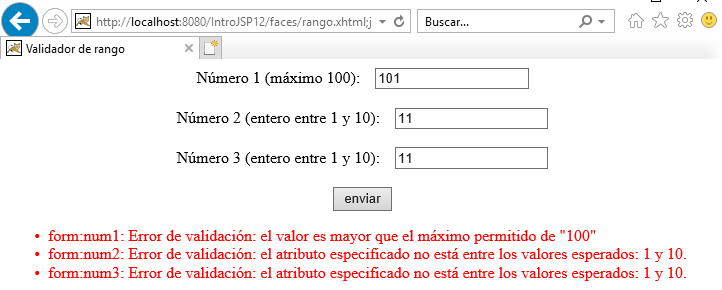
Para este ejemplo tendremos un Java Bean “inicio”, en el cual tendremos los atributos num1, num2, num3 y sumar con sus respectivos setters y getters.



Nuestro formulario lucirá de la siguiente manera, tendremos los inputs para los 3 números. Lo importante en cada etiqueta será la etiqueta de validación “f:validateLongRange” con sus atributos “mínimum” y “máximum”, permitiendo delimitar el rango en el que la captura nos permitirá introducir los datos.



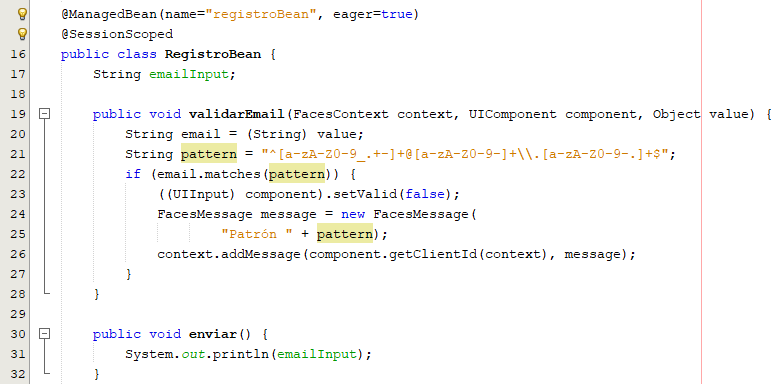
Si los datos introducidos son erróneos se mostrarán los mensajes de error.



### Métodos validadores

A un campo de texto se le puede agregar un atributo de validación para propósitos particulares en los que necesites realizar una validación especial. Para que se ejecute la validación del campo, el atributo “required” se debe agregar en la etiqueta de nuestro formulario.

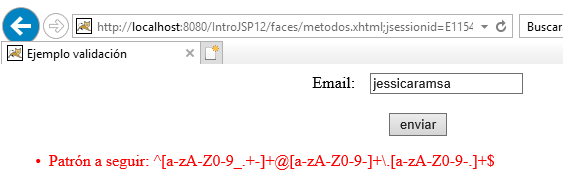
Tendremos el Java Bean “registroBean” con el atributo “emailInput”, los métodos set y get de dicho atributo junto con el método “validarEmail” que se encargará de realizar la validación de que el email introducido tiene la estructura correcta.



La página del formulario lucirá de la siguiente manera.



Comprobamos que al introducir una estructura errónea de un email nos manda el mensaje de error.



# CONCLUSIÓN

Construir una aplicación web con acceso a una base de datos para el procesamiento y la recuperación de los datos se ha vuelto algo tan importante y básico de desarrollar en la actualidad. Permite a los usuarios ver y filtrar el estado de todos los valores de un objeto, ver el historial de búsquedas de ese objeto, buscar documentos relacionados con ese objeto y cargar datos de otra aplicación.

Las validaciones en los formularios son muy importantes para recordar y notificar al usuario acerca de la información correcta. Se deberían incluir validaciones en todos los formularios, incluyendo la salida, así como en las operaciones del CRUD donde los campos del formulario dependen del propio formulario: a menudo no es un requisito de la aplicación, puede ser una característica si los campos requeridos son obligatorios.