**Agenda Online**

[Análisis funcional](#h.7e04e3qrvrg4)

[Añadir amigo](#h.jqf3h0am782x)

[Modificar amigo](#h.prx6oyayts2)

[Eliminar amigo](#h.i6a0jkooj64p)

[Ver todos](#h.18zidaty6j55)

[Navegación](#h.6csumvp5yitx)

[Análisis técnico](#h.2n2eyzrmcxme)

[Clases y JSPs](#h.xjvgeinysub6)

[JSPs](#h.3da35g4slb42)

[Clases](#h.llqvhjx1mpcu)

[Diagramas de flujo](#h.u2ihzk68pqt2)

[Añadir amigo](#h.nj2rpuq4ubp)

[Modificar amigo](#h.7lqoxuc1j7w)

[Eliminar amigo](#h.ktjhr6cxbzqj)

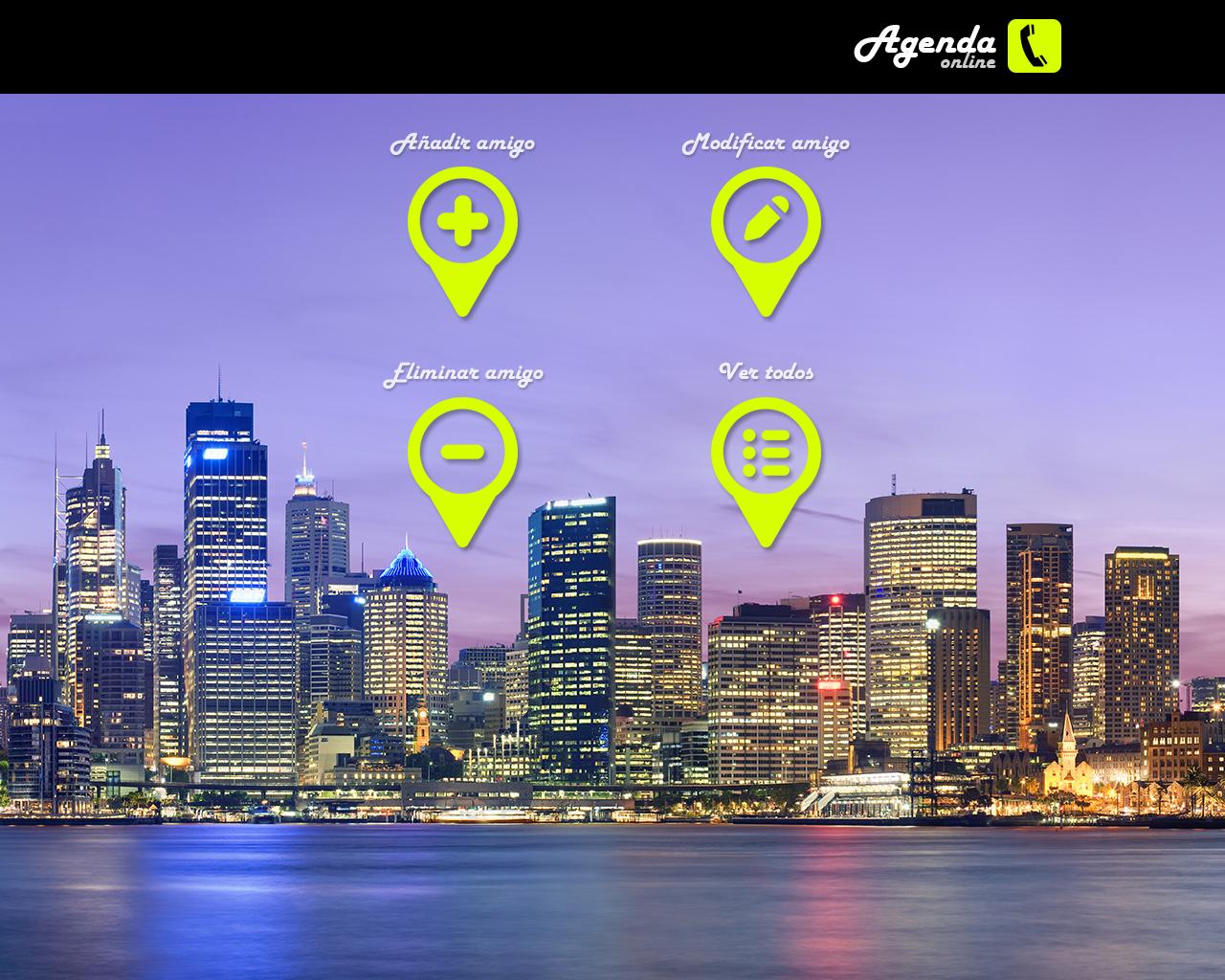
[Ver](#h.74cybtnnxozw)

# Análisis funcional

La app Agenda se utilizará como una agenda online para poder tratar y gestionar los diferentes contactos que tengamos en ella. Las diferentes opciones a nuestra disposición serán:

* Añadir
* Modificar
* Eliminar
* Ver

A continuación se muestra una pantalla de la home de la aplicación.

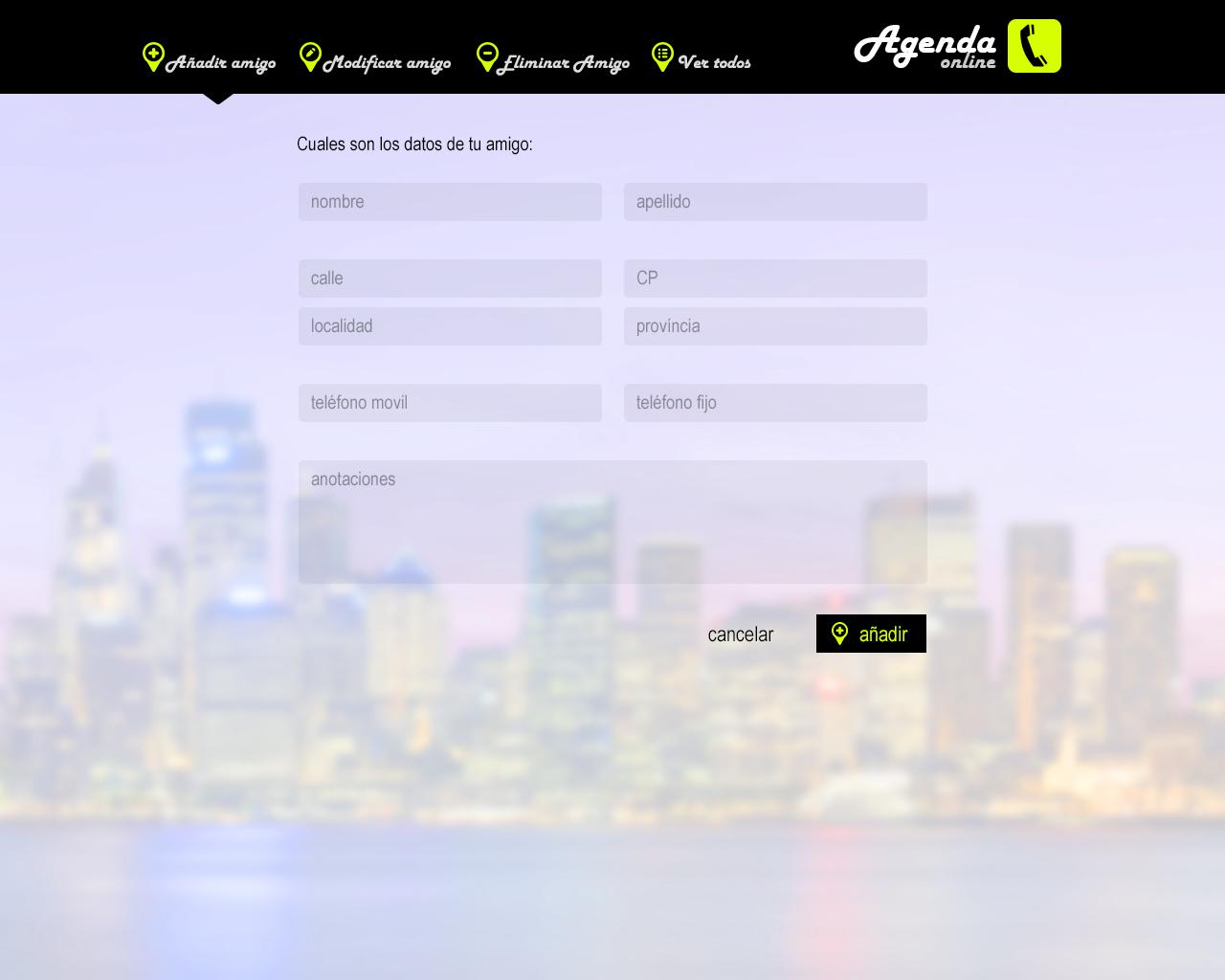


## Añadir amigo

Desde esta ventana se nos dará la posibilidad de añadir un nuevo contacto a nuestra agenda introduciendo todos sus datos personales. Los datos a introducir para el nuevo contacto serán:

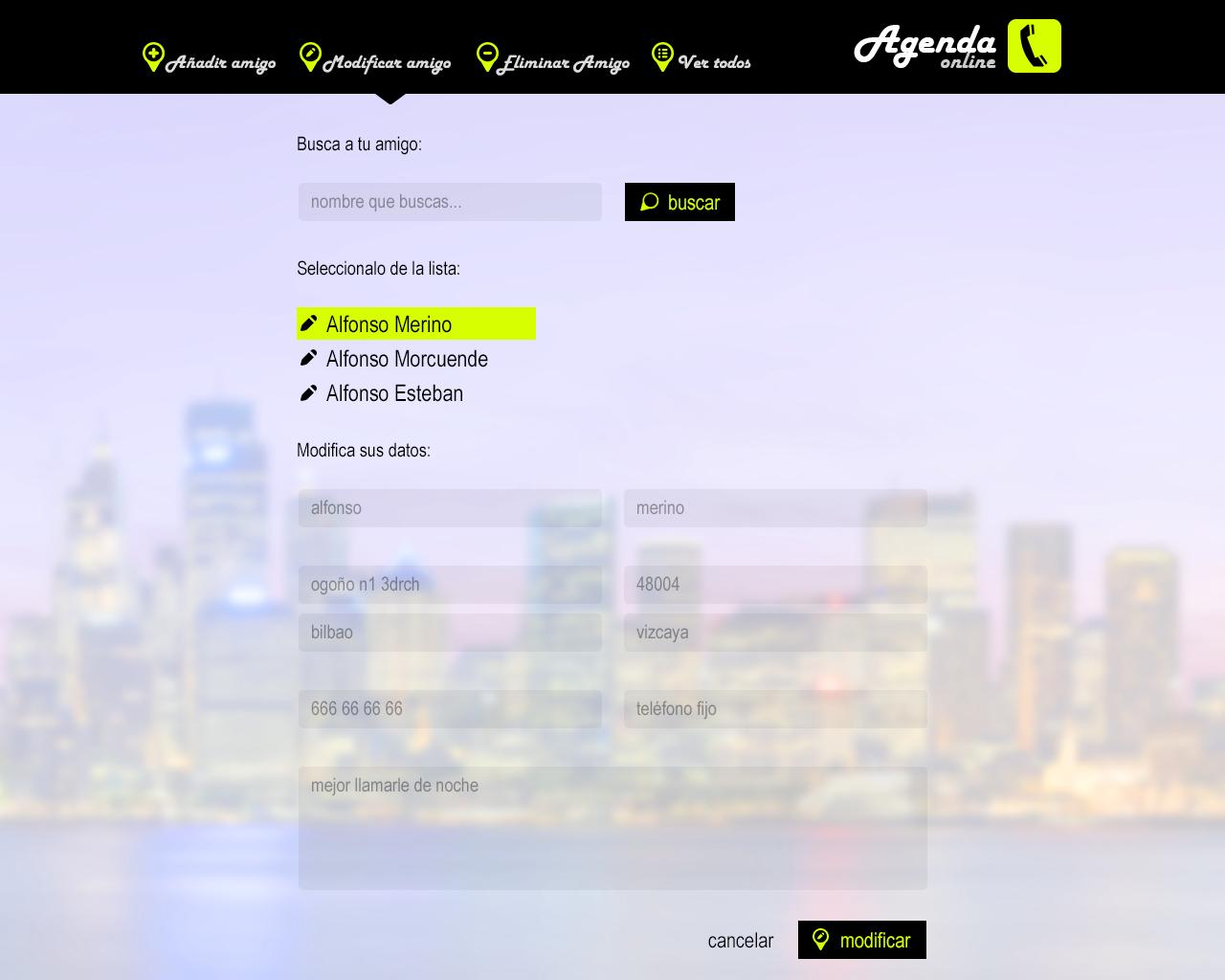
* Nombre
* Apellido
* Calle
* CP
* Localidad
* Provincia
* Teléfono móvil
* Teléfono fijo
* Anotaciones

Una vez introducidos los datos, el contacto será guardado mediante el botón *añadir*, mientras que con el botón *cancelar* volveremos a la home del programa.



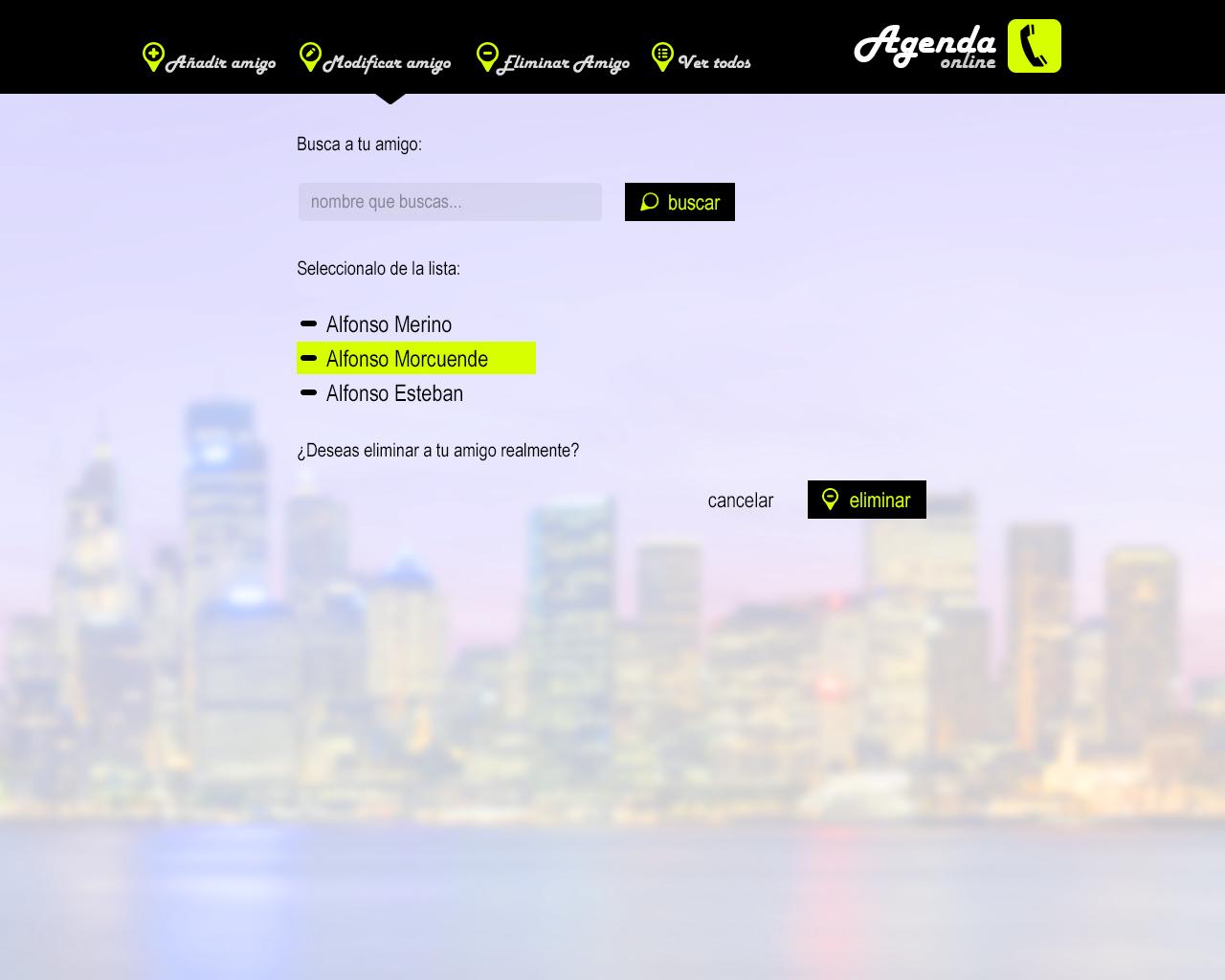
## Modificar amigo

Mediante esta opción seremos capaces de modificar los datos de un contacto ya almacenado en nuestra agenda online. Para ello deberemos buscar el contacto deseado mediante el buscador dinámico, el cual te va generando una lista con las coincidencias encontradas en la base de datos sobre lo que vayas escribiendo en el. Una vez localizado el contacto deseado, rellenaremos los campos que queramos editar. De igual manera que en el caso anterior, mediante el botón *modificar* guardaremos los cambios, mientras que con el botón *cancelar* volveremos a la home del programa.



## Eliminar amigo

Eliminar amigo nos permitirá borrar de la lista a uno de los amigos que tengamos añadidos previamente en la agenda. Al igual que en el caso anterior, podremos acceder al amigo mediante la lista que se genera al buscar. Como es habitual, mediante el botón *eliminar* guardaremos los cambios, mientras que con el botón *cancelar* volveremos a la home del programa.



## Ver todos

En esta pantalla se nos listará todos los contactos que tengamos añadidos en nuestra agenda junto a sus datos.



## Navegación

Durante todo momento, salvo en la home, contaremos con una barra de navegación en la parte superior de la página que nos permitirá acceder a las diferentes secciones de la aplicación, es decir, la home, añadir, modificar, eliminar y ver.

# Análisis técnico

## Clases y JSPs

A continuación se listan las clases y los JSPS necesarios para hacer funcionar la aplicación.

### JSPs

Las JSPs podemos dividirlas en dos secciones, las principales y las complementarias. Las principales serán las asociadas a las 5 páginas principales de la aplicación, es decir, la home, añadir, modificar, eliminar y ver. Todas estas incluyen las JSPs head.jsp, header.sp, footer.jsp, navBar.jsp y algunas incluirán formularios y el buscador.

* **index.jsp**: es la página principal que muestra las diferentes opciones que podremos realizar en la aplicación.
* **anadir.jsp**: es la página encargada de añadir los nuevos contactos a la agenda. Incluye el formulario forms/anadir.jsp
* **modificar.jsp**: mediante esta jsp modificaremos los datos de los contactos. Incluye el formulario forms/anadir.jsp y el forms/buscador.jsp
* **eliminar.jsp**: es la encargada de borrar los contactos de la agenda. Incluye forms/eliminar.jsp y forms/buscar.jsp
* **ver.jsp**: nos muestra el listado de los contactos de la agenda.

Dentro de la carpeta form se enuentran los siguientes JSPs adicionales:

* **anadir.jsp**: es el formulario necesario para la creación de nuevos contactos.
* **modificar.jsp**: es el formulario necesario para editar los datos de los contactos.
* **eliminar.jsp**: formulario que contiene el buscador y la lista de contactos para eliminar.
* **buscador.jsp**: es el buscador dinámico que irá generando una lista con los contactos según vaya encontrando coincidencias entre los escrito en el y la base de datos.

Además existen cuatro JSPs comunes para todas las páginas:

* **head.jsp**: contiene la cabecera de la página con su título y la carga de estilos y scripts.
* **header.jsp**: contiene el logo más la barra de navegación, esta última siempre y cuando no nos encontremos en la home de la aplicación.
* **footer.jsp**: contiene el pie de la página.
* **navBar.jsp**: esta será la barra de navegación de la aplicación. Si nos encontramos en la home se mostrará como cuatro iconos en grande en medio de la página, mientras que si estamos en las demás se mostrará como una barra con las diferentes opciones en la parte superior de la página.

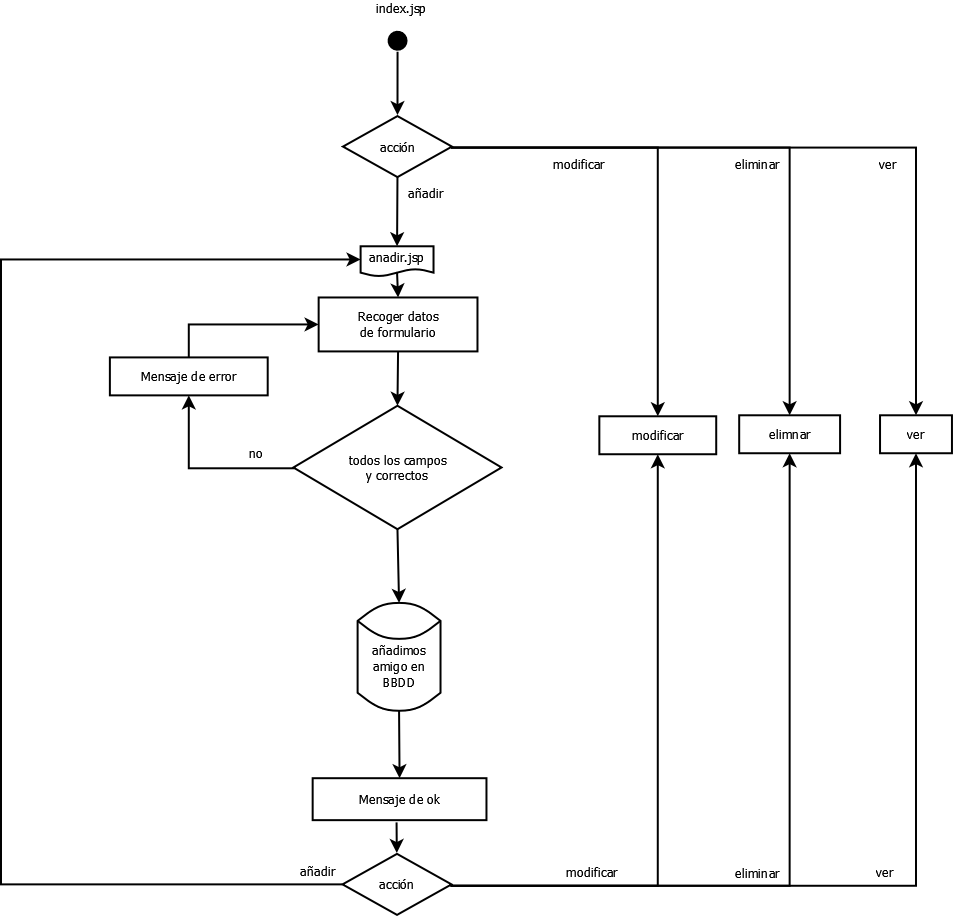
### Clases

Las clases mostradas en el siguiente listado son las necesarias para hacer funcionar la parte servidor de la aplicación Agenda Online.

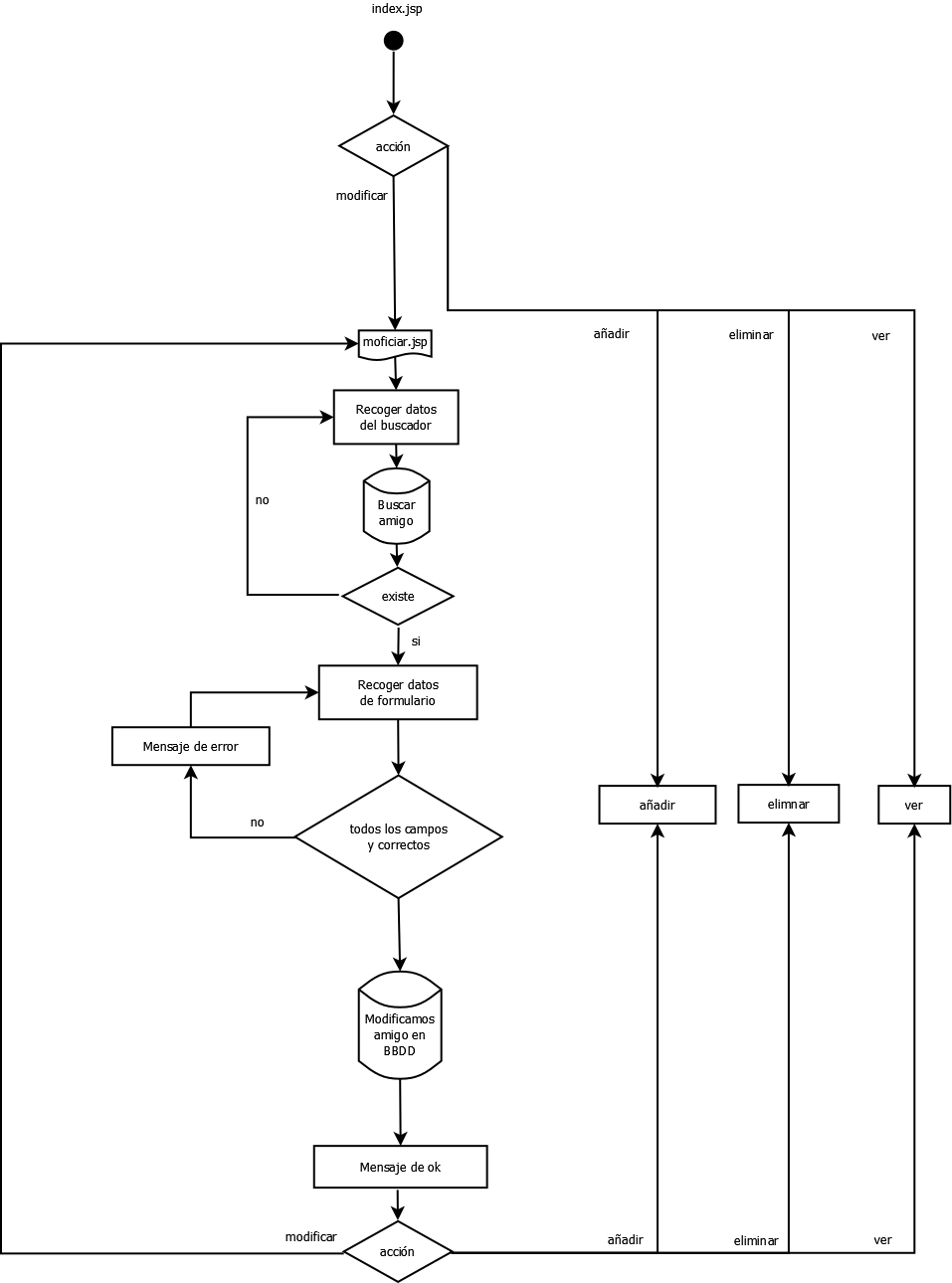
* **Amigo.java**: será la clase encargada de gestionar los objetos amigo o contacto que almacenarán los datos de los contactos que esté tratando la aplicación. Contará con los mismos atributos que los datos a introducir en la aplicación más un campo id para la base de datos:
  + id
  + nombre
  + apellido
  + calle
  + cp
  + localidad
  + provincia
  + móvil
  + fijo
  + anotaciones
* **MainServlet.java**: este será el servlet principal por el que pasarán todas las llamadas y que se encargará de realizar las diferentes operaciones. Dependiendo de los distintos parámetros que le lleguen mediante el get y el post, este realizará los siguientes métodos:
  + anadirAmigo: será el método encargado de guardar un nuevo contacto en la base de datos.
  + modificarAmigo: será el método encargado de modificar el contacto seleccionado.
  + eliminarAmigo: será el método encargado de eliminar un contacto.
  + verAmigos: será el encargado de listar todos los amigos.
  + buscar: realizará en la base de datos la búsqueda de los amigos que concuerden con el nombre y apellido introducidos.
* **InitListener.java**: este listener se encargará de inicializar tanto el log como el ConnectionFactory al inicio de la aplicación al desplegarse esta en el servidor.
* **ConnectionFactory.java**: clase con la que manejaremos la conexión a base de datos.
* **IDAOAmigo.java**: interfaz que define los métodos para el DAOAlumno. Estos métodos a implementar serán los siguientes:
  + boolean add(amigo)
  + ArrayList<Amigo> getAll()
  + ArrayList<Amigo> getByName(String value)
  + Amigo getById(int id)
  + boolean update(Amigo amigo)
  + delete(Amigo amigo)
* **DAOAmigo.java**: clase que implementa el interfaz IDAOAlumno y cuyo cometido es implementar sus métodos y realizar las conexiones pertinentes a base de datos.
* **ModeloAmigo.java**: esta clase es la encargada de crear el modelo del amigo para gestionar la base de datos de manera que el usuario no se percate de ello.

## Diagramas de flujo

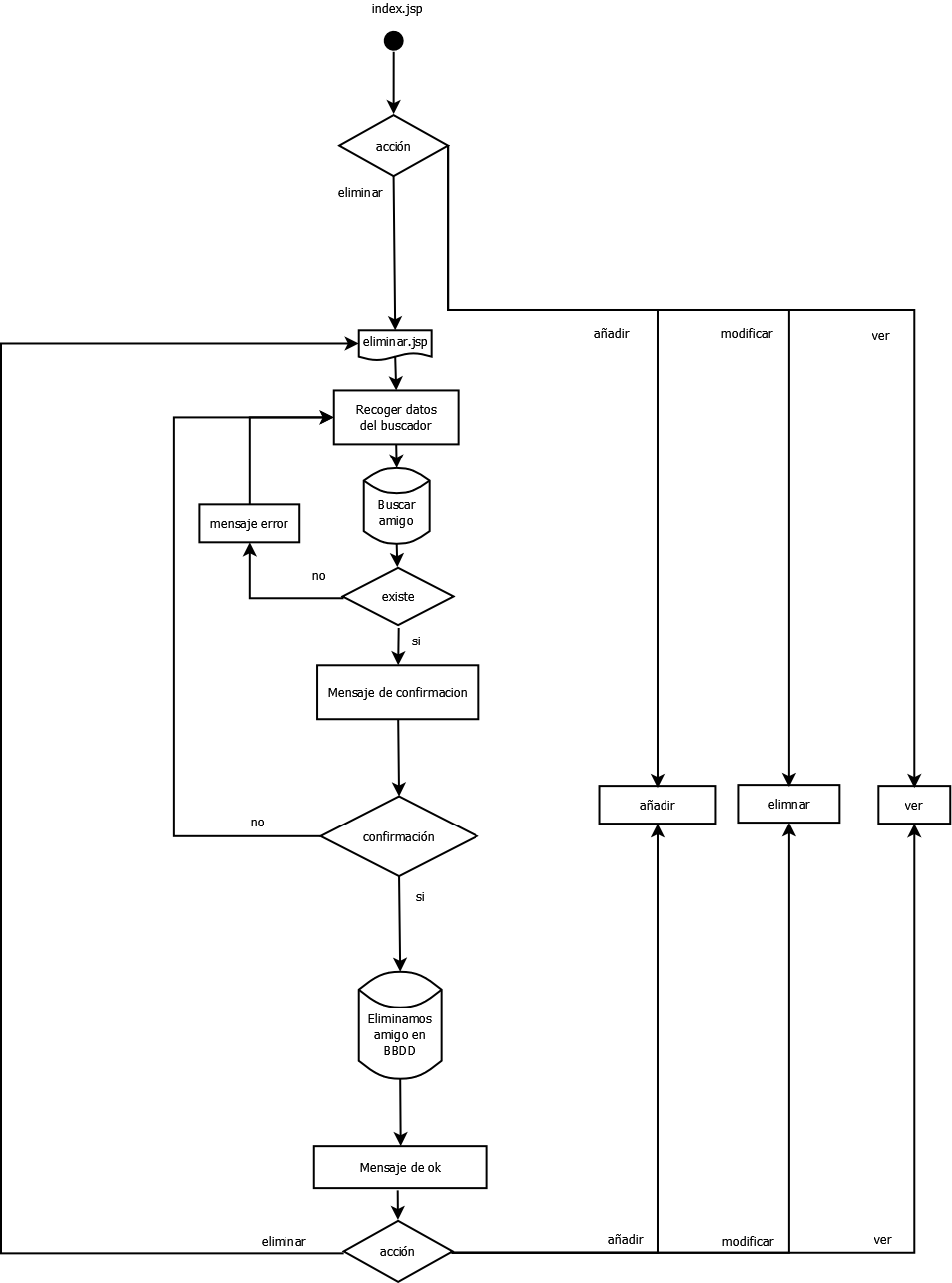
### Añadir amigo



### Modificar amigo



### Eliminar amigo



### Ver

