**Reporte de Implementación**

**Database Server**

Javier Sagastuy 122627

Alejandro Escalante 125653

*Sistemas Distribuidos*

Profesor José Octavio Gutiérrez García

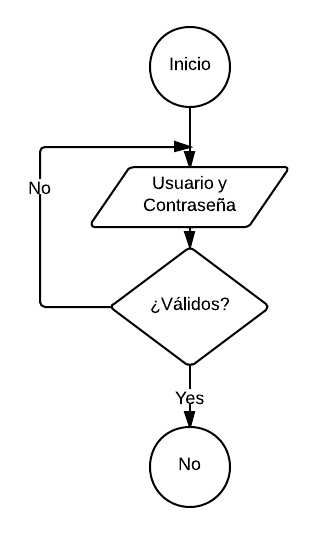
ITAM

Primavera 2014

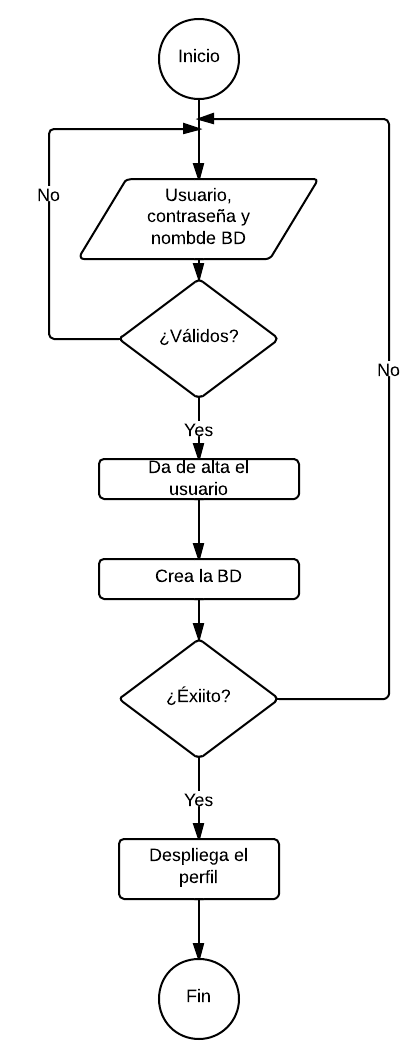
# Documentación UML

## Algoritmos de flujo de los procesos principales

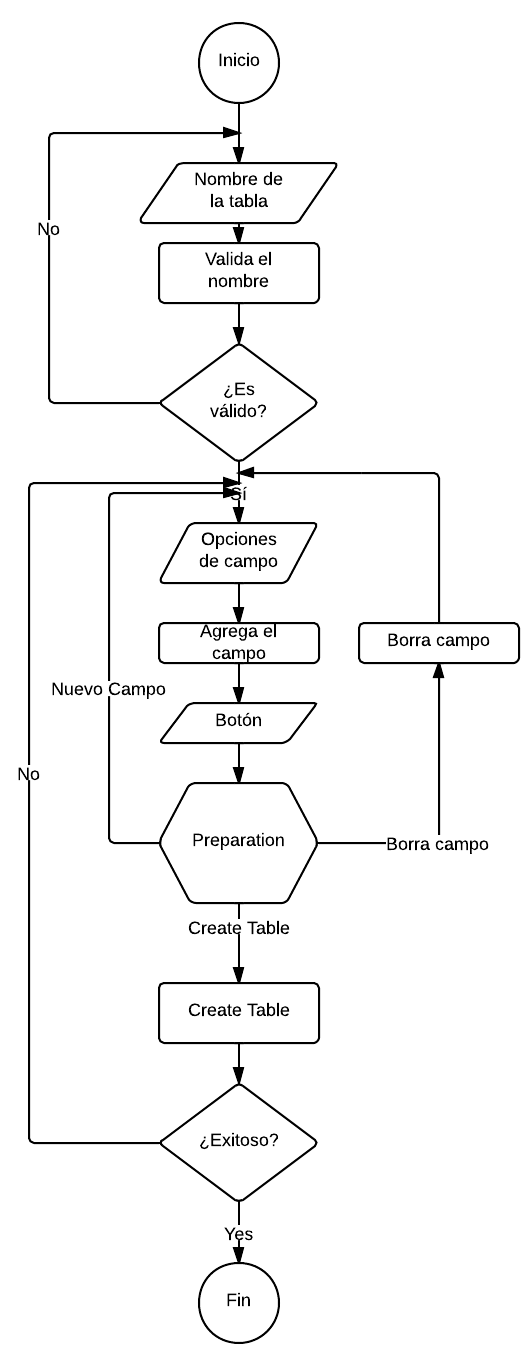
### Login



### Register

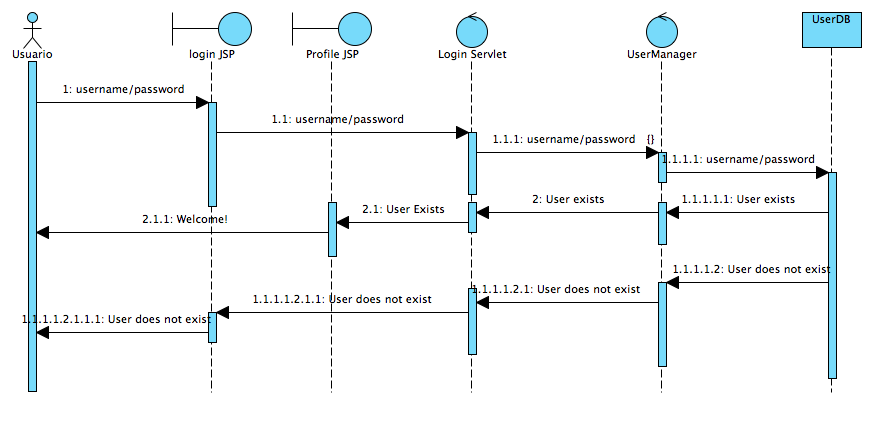


### Create Table

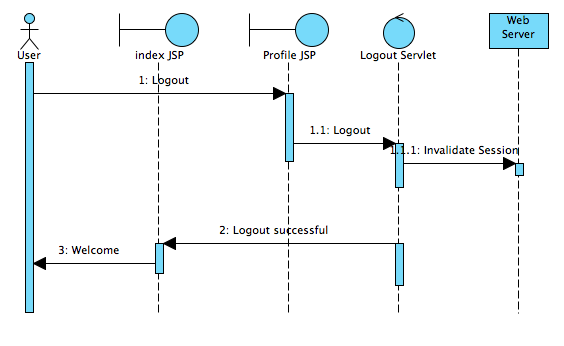


## Diagramas de secuencia

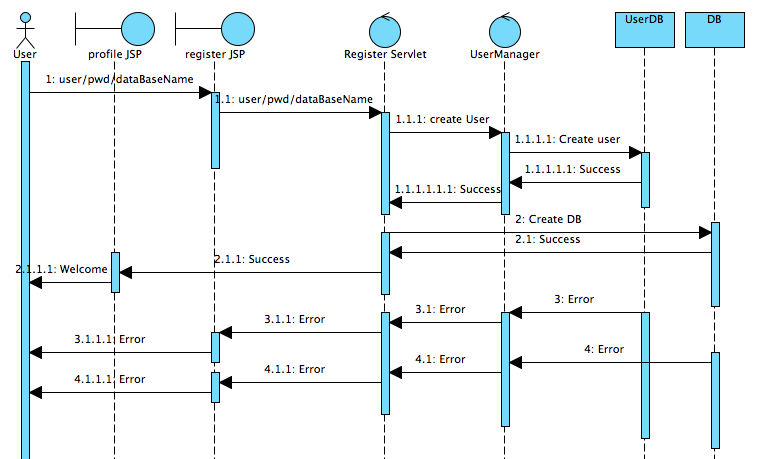
### Login



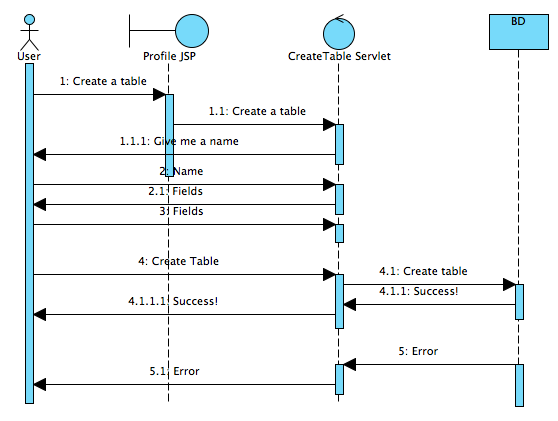
### Logout



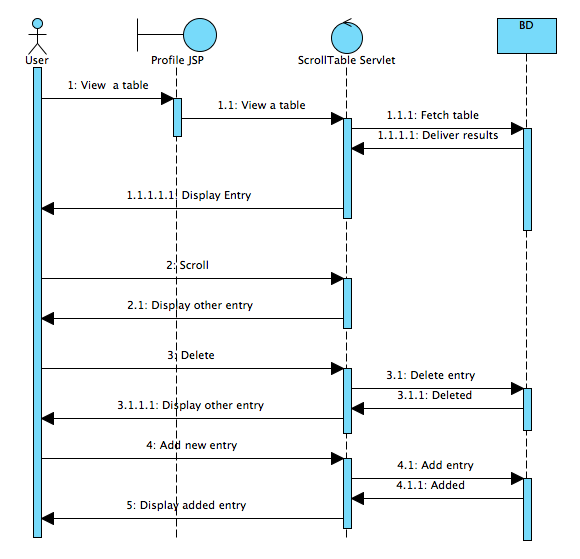
### Register



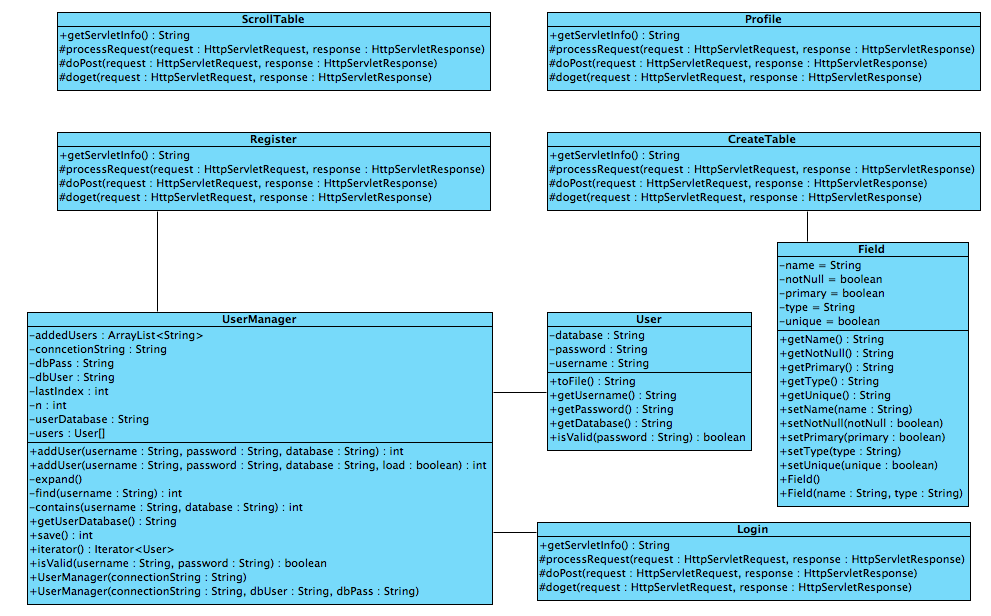
### Create Table



### View Data



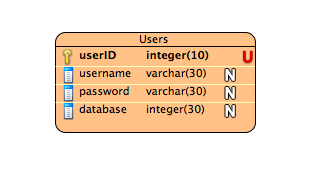
## Diagramas de clases



## Diagrama de base de datos

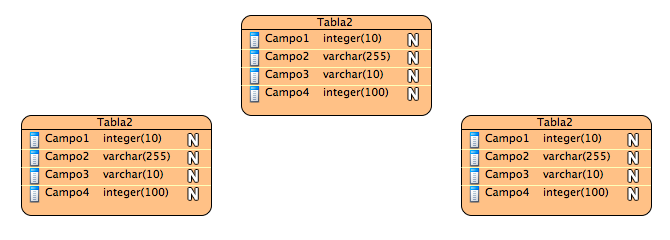
### UserDB

Esta base de datos se usa solo para almacenar los usuarios registrados y poder validar que los usuarios que ingresen ya existan en la base de datos. Cuenta de una sola tabla: Users. El detalle de la tabla se puede ver en la siguiente figura.



### Bases de datos de los usuarios

Estas bases de datos son las que pueden crear dinámicamente los usuarios. Por ello, es imposible dar una especificación exacta de la tabla. Dado que la funcionalidad de la aplicación está limitada, no podemos introducir relaciones entre las tablas. Entonces, la base de datos que se crea es sencilla, se crean tablas aisladas (sin relación entre ellas), cada una con sus campos del tipo y con las opciones especificadas en la aplicación web. A continuación se presenta un diagrama genérico de una base de datos con tablas aisladas.



# Estrategia utilizada

La estrategia utilizada fue demasiado ambiciosa para el tiempo con el que contábamos para hacer el proyecto. Por ello, la primera parte del proyecto la sigue a la perfección y la última parte se desvía un poco.

La idea básica es tener sólo tres JSPs: index.jsp, login.jsp y register.jsp. A través de hipervínculos se navega de una a la otra. La JSP register.jsp, trabaja junto con un Servlet: Register. Éste se encargará de dar de alta un usuario y crear la base de datos sobre la que va a trabajar. Las altas se realizan a través de la clase UserManager y la información se almacena en la tabla Users de la base de datos UserDB que se supone que ya existe con anterioridad. En register.jsp se valida que los campos capturados sean válidos y a través del servlet Register, se valida que los campos capturados sean nuevos, esto es: que no existan ya en la base de datos de los usuarios.

Una vez que se ha registrado el usuario, su información queda almacenada en la base de datos de usuarios y su propia base de datos ha sido creada. Ahora, el usuario debe hacer Login a través de login.jsp.

login.jsp sólo muestra los campos a capturar. Se apoya principalmente en un Servlet llamado Login. Dicho servlet valida que el usuario capturado exista y que su contraseña sea correcta. En ese caso, se crea una sesión http con los valores de nombre de usuario, contraseña y nombre de su base de datos.

Ahora, el procesamiento lo hará un servlet llamado Profile. Con los datos de la sesión el usuario, se despliega una página que da la bienvenida y permite hacer logout si así se desea. El logout simplemente invalida la sesión http. Por otro lado se encuentra la opción de crear una nueva tabla en la base de datos. Esto se hace a través de un servlet llamado CreateTable. Con ese servlet se construye dinámicamente una página html para que se pueda elegir qué campos y qué opciones tendrán en la nueva tabla. Al presionar el botón crear, se construye el comando que creará la tabla en la base de datos y se crea la tabla.

También en el la página que es generada por el servlet Profile, se despliega siempre un Drop Down List con los nombres de todas las tablas contenidas en la base de datos. La idea es que se pueda seleccionar una de ellas, y al presionar el botón “View table data”, se puedan ver los datos de la tabla y editarlos de forma dinámica. Dicho botón nos manda a un servlet llamado ScrollTable. En él se construye la página que no sólo permitirá hacer scroll registro por registro de la tabla, si no que permitirá agregar nuevas tuplas a la tabla y eliminar tuplas existentes.

# Obstáculos

* El primer obstáculo con el que nos encontramos fue cómo llevar el control de los usuarios dados de alta en el sistema. Comenzamos por crear un archivo con los nombres de los usuarios, sus contraseñas y el nombre de su base de datos. Posteriormente nos dimos cuenta de que esto era muy ineficiente y optamos por crear una base de datos que tuviera una tabla con los usuarios dados de alta. Para manejarla se crearon las clases User, UserManager que se encargan de abrir las conexiones la base de datos sólo cuando es necesario.
* El segundo obstáculo que nos encontramos fue cómo poder aceptar acentos en los campos de usuario y contraseña. Tras una búsqueda profunda, vimos que sólo se debe cambiar la codificación de la página entera en la que se quieren leer acentos. Esto se puede hacer ya sea en un JSP o en un servlet a través del método response.setContentType();
* El tercer obstáculo importante con el que nos encontramos fue la manera de desplegar el ResultSet resutlante de la consulta a determinada tabla para poder hacerle scroll y modificarlo. La parte complicada fue cerrar la conexión a tiempo. Para ello, se optó por una vez hecha la consulta, almacenar los resultados en una matriz de objetos. Dicha matriz se le enviaría al servlet encargado de desplegar el contenido de la tabla a través de la sesión http.
* Otro obstáculo relacionado con el anterior fue poder obtener los metadatos de los nombres de las columnas en una tabla dada. Dado que cerramos la conexión cada vez que se hacía una consulta, era necesario almacenar el ResultSet en un arreglo de objetos. Para poder hacer esto era necesario conocer los nombres de las columnas.
* Un obstáculo fundamental fue producir código aceptable después de las 4 am.

# Lista de bugs conocidos

* En el momento de la escritura de este documento existía un bug en el que al registrar un usuario, al acceder a su perfil no se mostraban sus datos. Creemos que el error es que el servlet Register no está pasando bien los datos del usuario al servlet Profile. Si el usuario hiciera logout y luego login, la aplicación funciona bien. Se espera que en la versión entregada del código este bug ya no exista.
* No es propiamente un bug si no una falta de funcionalidad, pero de momento la interfaz sólo permite que el primer campo agregado a una tabla sea la llave primaria. Los campos que se pueden agregar sólo pueden ser enteros, dobles o varchars.
* De momento, no se validan los nombres de los campos que se van a agregar en una nueva tabla para verificar que no se intenten agregar campos con el mismo nombre.