Clase 21

Java con persistencia en Base de Datos

Cómo crear una aplicación Java para gestión de base de datos (1era parte)

Cómo crear una aplicación Java para gestión de base de datos (Parte 1)

vamos Buenos Aires

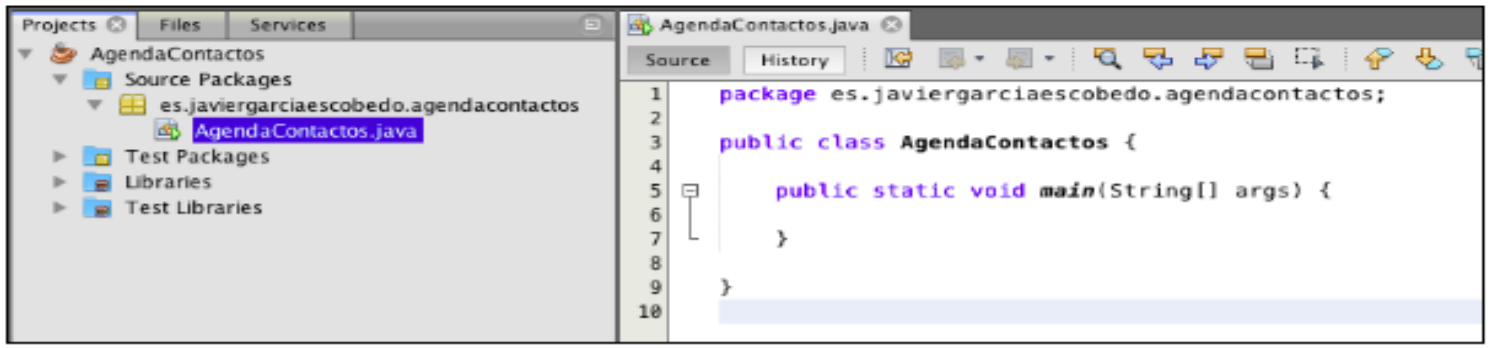
Siguiendo los pasos que se muestran a continuación podrás crear una **aplicación Java** que gestione una simple **lista con datos de personas**, almacenada en una **base datos** de manera que persistan los datos de los contactos aunque se cierre la aplicación. Esta aplicación podrá servirte de base para crear otras aplicaciones similares que realicen la gestión de otras bases de datos.

Para la creación completa de la aplicación se usarán distintas **tecnologías** que podrías cambiar por otras si lo estimaras oportuno, pero este tutorial se centrará en las siguientes:

- **Java**, como lenguaje de programación.
- **NetBeans**, como entorno de desarrollo.
- **Java DB** (Derby), como gestor de la base de datos.
- **JPA** (Java Persistence API), como API para el acceso a la base de datos.
- **JavaFX**, como API para el interfaz de usuario.
- **SceneBuilder**, como diseñador del interfaz de usuario.

Inicialmente deberás **crear una aplicación Java estándar**, asignándole el nombre ***AgendaContactos***, a la que posteriormente se le añadirá un interfaz gráfico más amigable para el usuario:

`File > New Project > Java > Java Application`

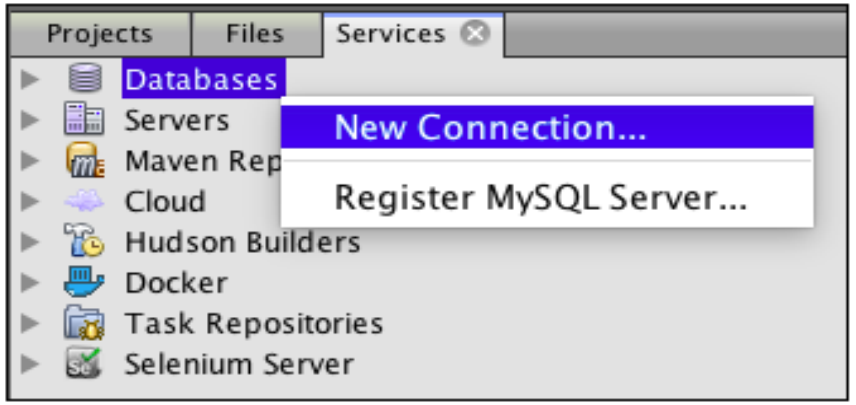


Base de datos Java DB

Registro de la base de datos en NetBeans

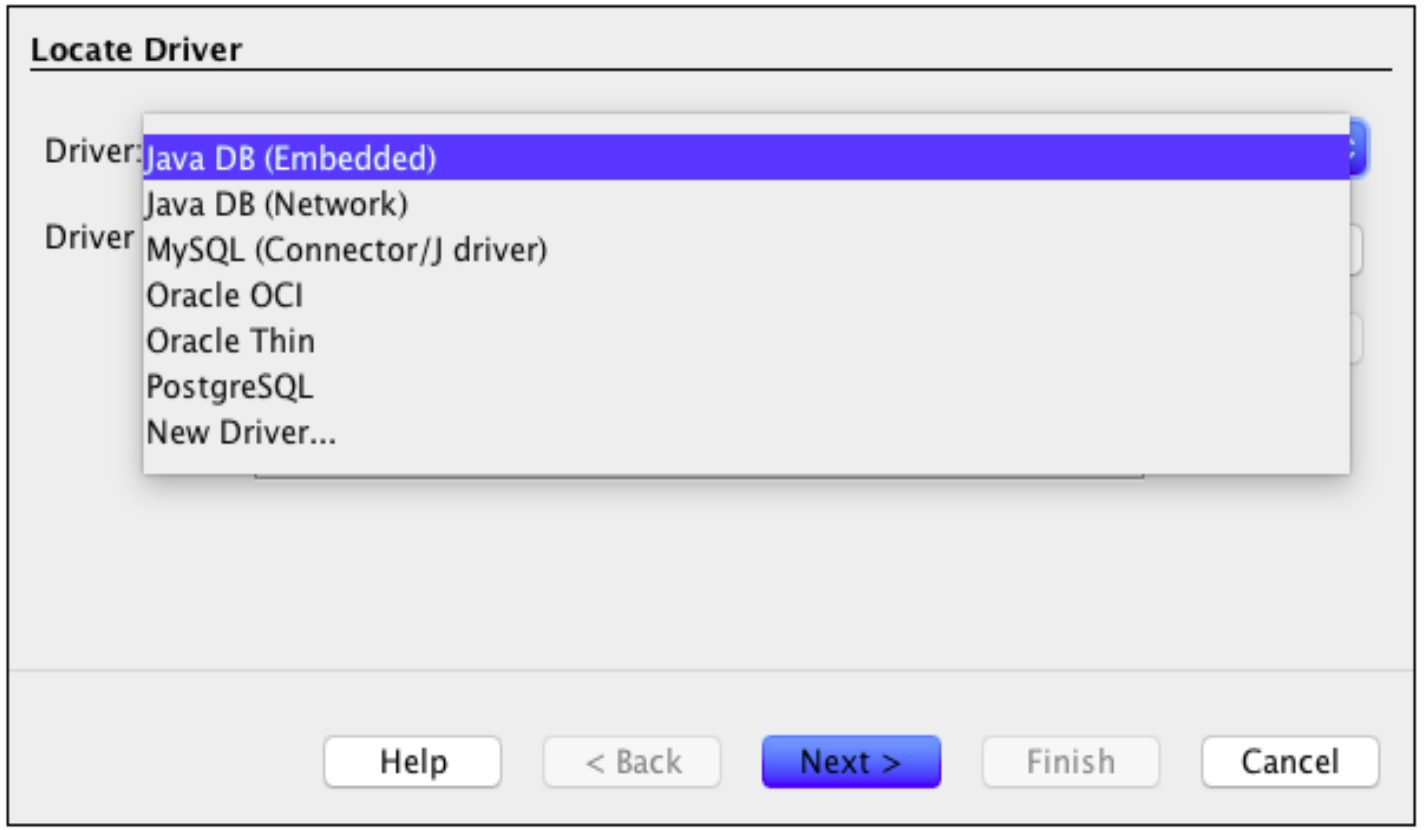
Con el fin de que la aplicación puede ser fácilmente utilizable por cualquier usuario, sin necesidad de que tenga que disponer de un servidor de base datos como MySQL (cuya instalación puede ser compleja para un usuario normal), se va a utilizar como gestor de base de datos el que incorpora Java, de manera que el usuario simplemente necesitará ejecutar la aplicación de la Agenda de Contactos para que pueda utilizarla, sin necesidad de tener otros requisitos para usar una base de datos. Java ofrece dentro de su kit de desarrollo el gestor de **base de datos Java DB que es una distribución por parte de Oracle de la base de datos Derby de Apache**. Debes tener en cuenta que el **lenguaje SQL de Java DB tiene ligeras diferencias** con el lenguaje SQL de MySQL, por lo que debes tener en cuenta su **página de referencia del lenguaje SQL de JavaDB**.

Para que la gestión de la base de datos de *Contactos* resulte más cómoda, conviene **registrar la base de datos en NetBeans**. Esto puedes hacerlo cómodamente desde la pestaña ***Services*** seleccionando, desde el menú contextual del elemento ***Databases,*** la opción ***New Connection***.



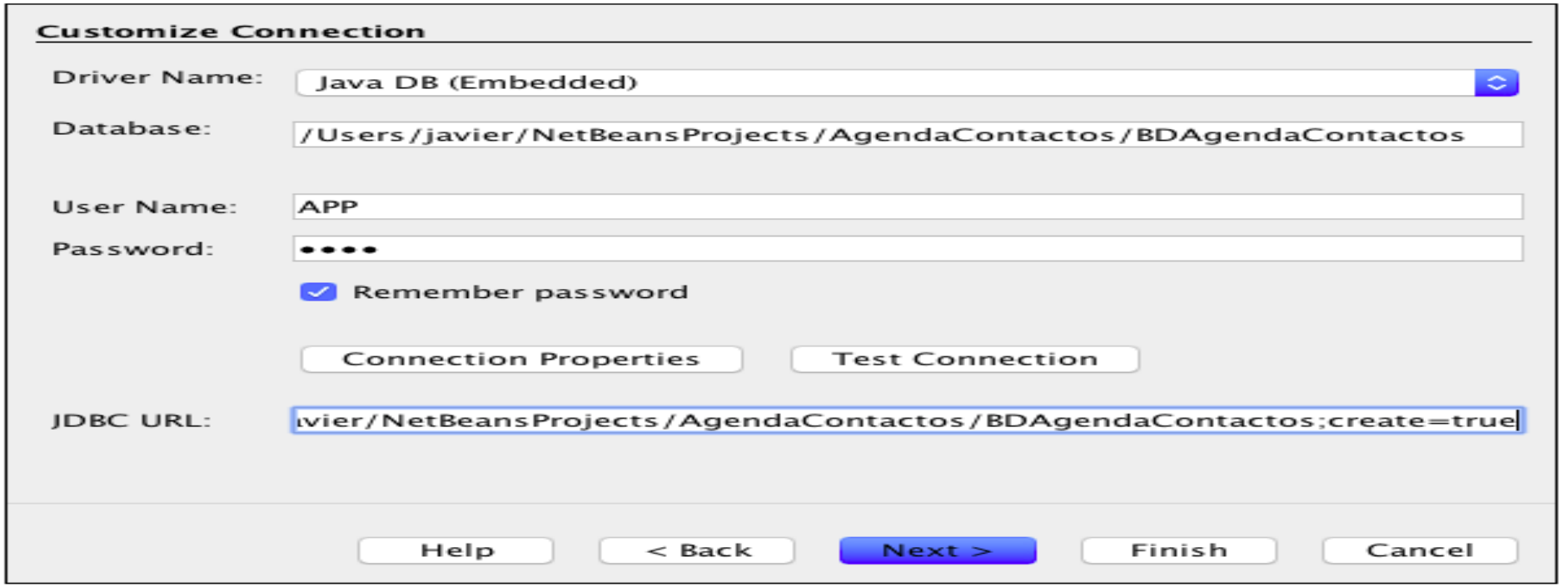
Base de datos Java DB

Selecciona la opción ***Java DB (Embedded)*** para utilizar el gestor de base de datos que se ha comentado anteriormente, de manera que no se requiera un servidor de base de datos externo a la aplicación que se está desarrollando. Si se prefiere utilizar un servidor de base de datos como MySQL, se deberá seleccionar la opción correspondiente.



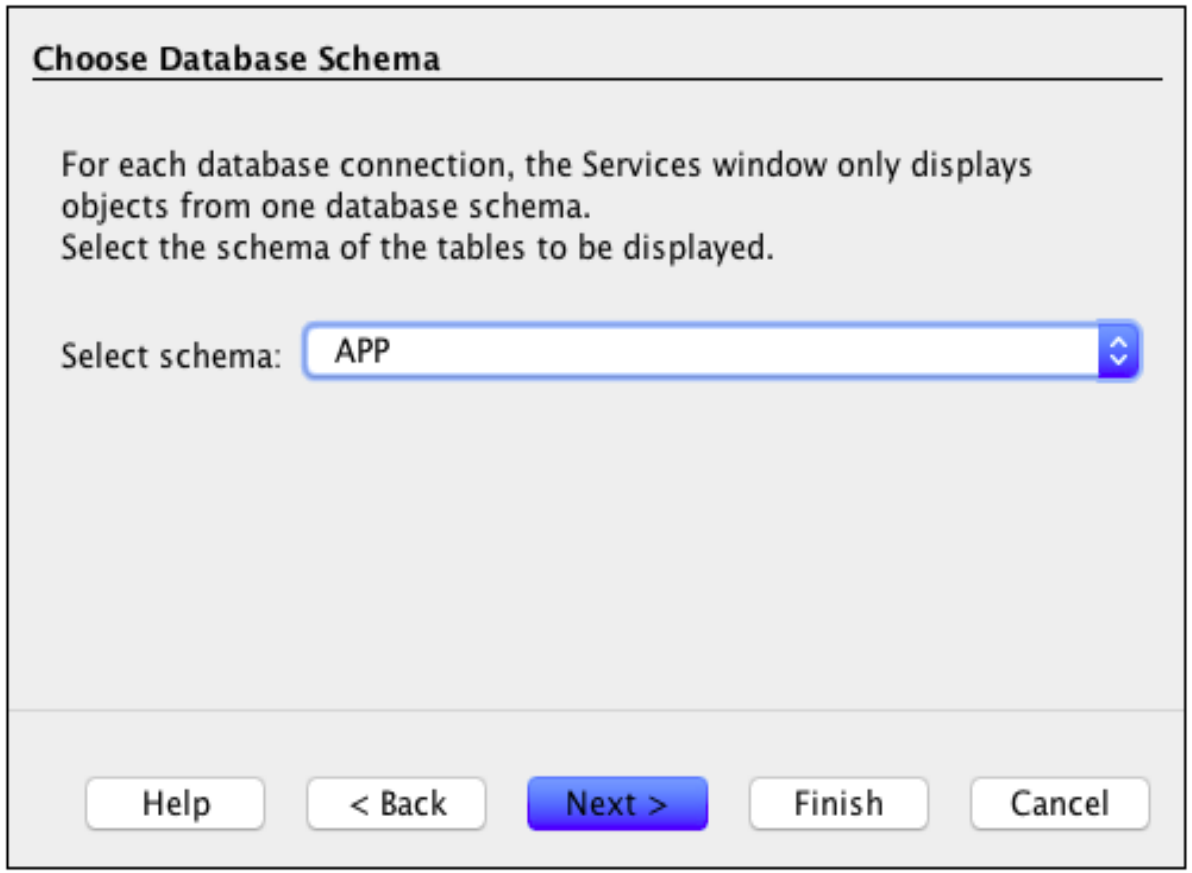
Base de datos Java DB

En la ventana de configuración de la conexión con la base de datos, indica la **ruta a la carpeta** donde has creado el proyecto junto con el **nombre deseado para la base de datos** (en este ejemplo *BDAgendaContactos*), el **nombre de usuario** (indica **APP** que es el usuario por defecto) **y contraseña** (la que quieras) que se desea utilizar para autorizar la conexión (indica los valores que desees), selecciona la opción recordar contraseña (Remember password) para que se almacene y no sea necesario introducirla cada vez que se desee hacer la conexión a la base de datos desde el entorno de NetBeans, y **añade el texto `;create=true` a la dirección (URL)** de conexión a la base de datos con el fin de que se **cree la base de datos en caso de que no exista** (que es el caso actualmente, ya que no se ha creado anteriormente dicha base de datos).



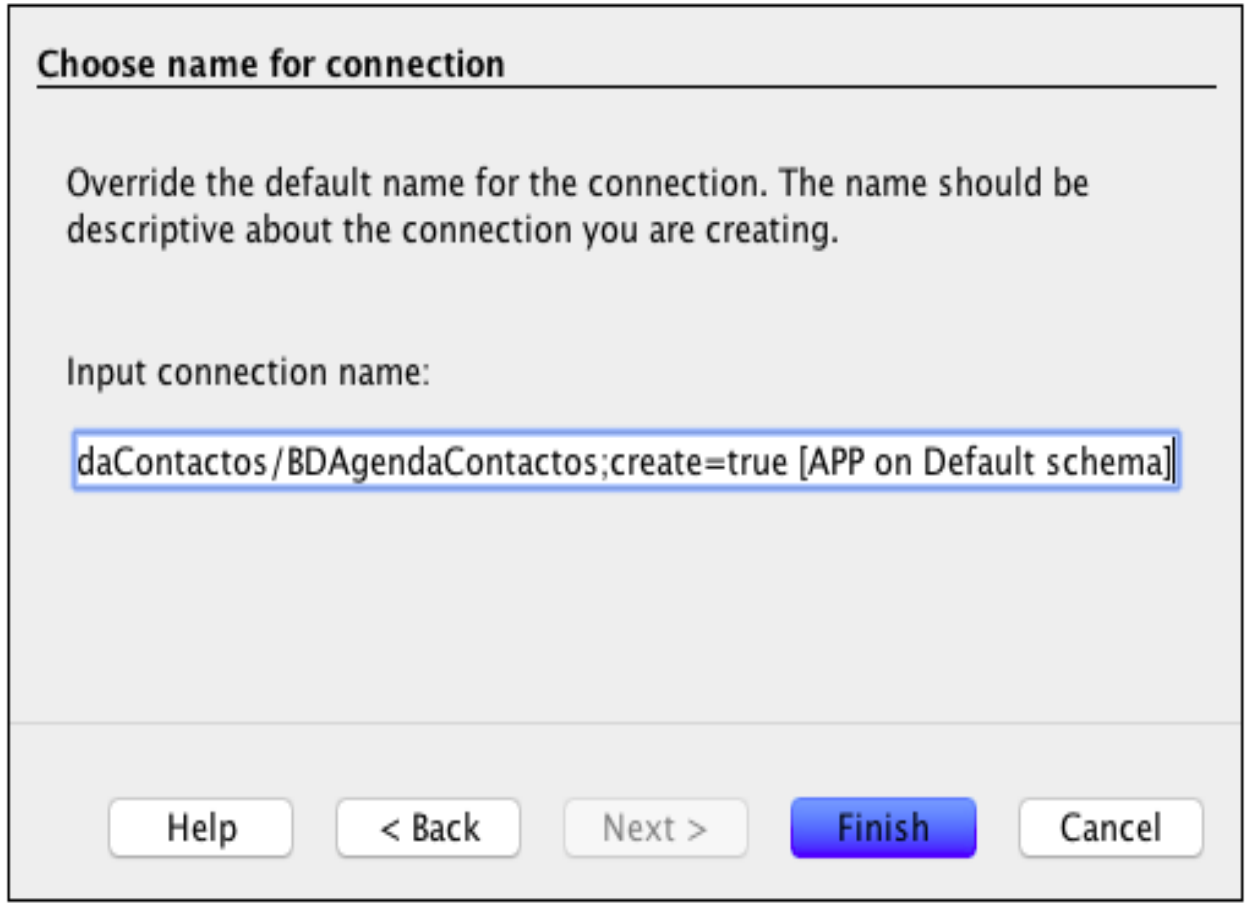
Base de datos Java DB

Para JavaDB debes indicar el esquema (***Schema***) de la base de datos que va a utilizar las herramientas de NetBeans, por lo que debes dejar el que aparece por defecto: **APP**, que es el utilizado para no indicar un usuario concreto de conexión a la base de datos.



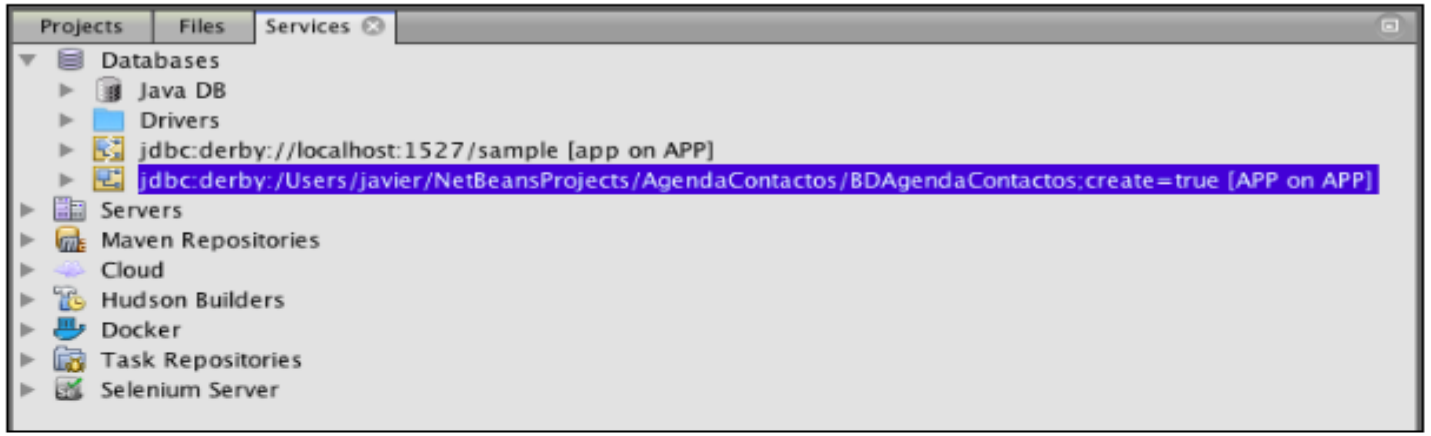
Base de datos Java DB

En la siguiente ventana podrás asignar un **nombre a la conexión** con la base de datos, aunque puedes dejar el que aparece por defecto.

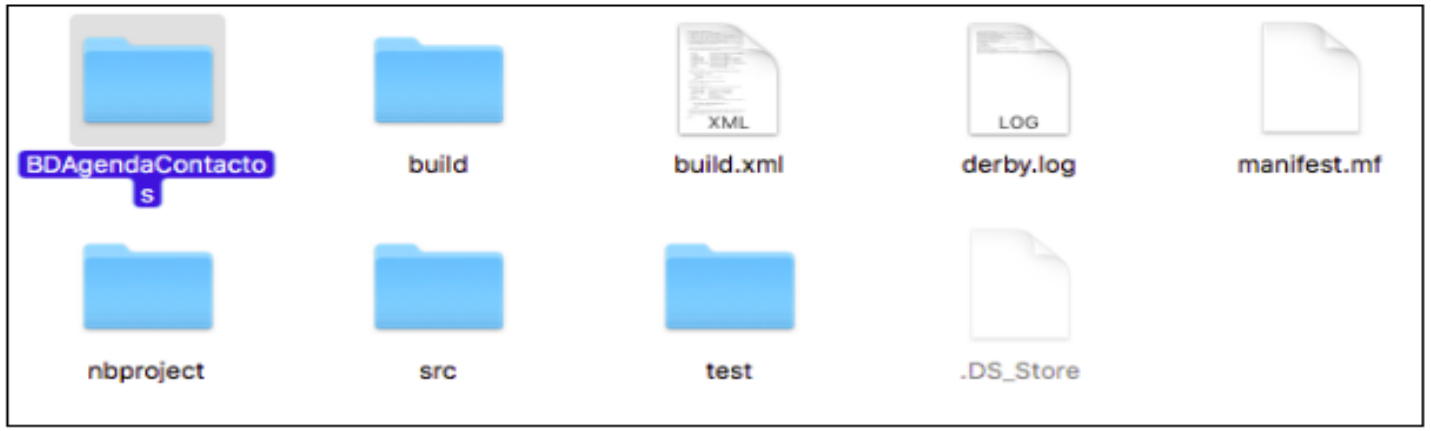


Base de datos Java DB

Una vez finalizado este proceso de registro de la base de datos en NetBeans, aparecerá su **conexión en el árbol *Databases***, y a patir de este momento será más sencillo realizar diversas operaciones con la base de datos desde el propio entorno de desarrollo NetBeans.



Comprueba también que en la carpeta donde has creado el proyecto se ha creado una **nueva carpeta correspondiente a la base de datos**.

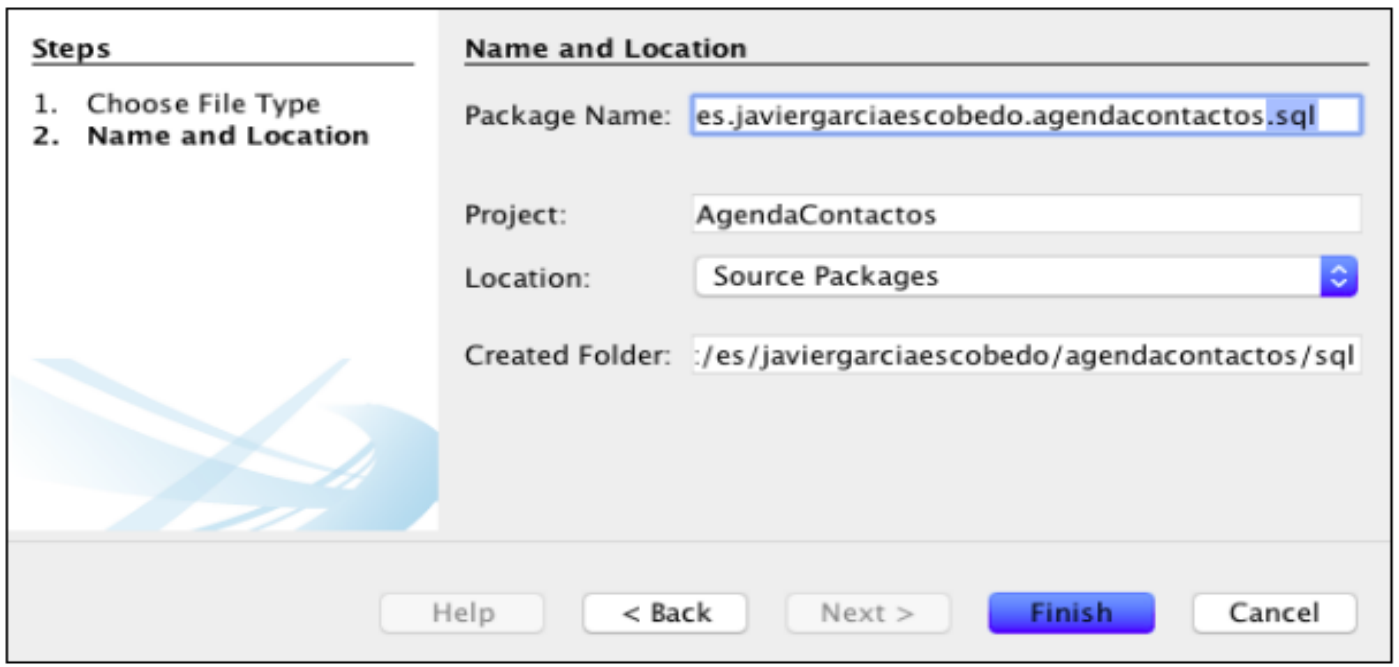


Creación de las tablas

Las tablas que van a formar parte de la base de datos se van a crear desde el mismo NetBeans utilizando un archivo SQL que contenga las sentencias SQL correspondientes a la creación de las tablas.

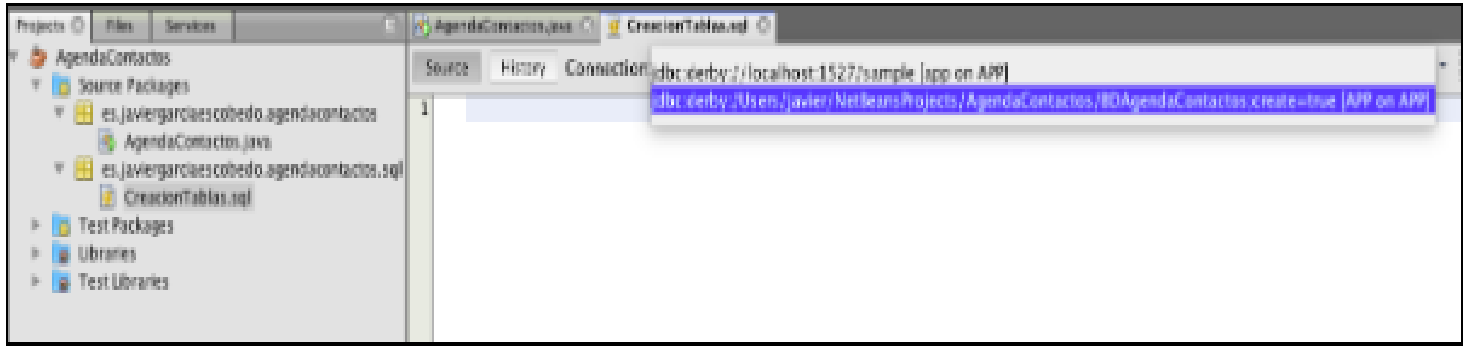
Con el fin de mantener organizado el código fuente de la aplicación, conviene crear un paquete que almacene el código SQL de manera separada del código Java propio de la aplicación Java que se desarrollará posteriormente.

Vuelve a la pestaña ***Projects***, y utiliza el menú contextual de la carpeta ***Source Packages*** para seleccionar la opción ***New*** > ***Java Package*** e indica el nombre para el paquete de fuentes que almacenará los archivos SQL. Un buen nombre puede ser el mismo que se ha utilizado para la aplicación, **seguido de *.sql***:



Creación de las tablas

Crea dentro de esta carpeta un archivo SQL desde su menú contextual, usando la opción ***New > Other > (Categories) Other > SQL file*** o *New > SQL File* si ya has usado anteriormente esta opción. Asigna un nombre al archivo, por ejemplo, *CreacionTablas*. Una vez abierto el archivo, asegúrate de **seleccionar en su parte superior la conexión con la base de datos** que se ha creado anteriormente, para que la ejecución de las sentencias SQL que se escriban se ejecuten sobre la base de datos deseada.



Como este tutorial no es más que un ejemplo de una posible base de datos, se va a crear una **tabla** para almacenar una serie de datos de **personas**. Los datos que se indicarán no tendrán mucho sentido en algunos casos, pero se intenta crear un ejemplo que utilice **diversos tipos de datos**, y de ahí que haya algunos datos añadidos de manera uno tanto forzada. También con el fin de crear un ejemplo con tablas relacionadas, se va a crear una **segunda tabla con las provincias** españolas que se relacionará con la tabla de personas, para indicar en qué provincia reside cada una.

Creación de las tablas

Para este ejemplo se ha creado el siguiente **script SQL** que se encargará de **crear las tablas**:

```
CREATE TABLE PROVINCIA (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
    CODIGO CHAR(2),
    NOMBRE VARCHAR(20) NOT NULL,
    CONSTRAINT ID_PROVINCIA_PK PRIMARY KEY (ID)
);

CREATE TABLE PERSONA (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY, -- Id autonumérico
    NOMBRE VARCHAR(20) NOT NULL,
    APELLIDOS VARCHAR(40) NOT NULL,
    TELEFONO VARCHAR(15),
    EMAIL VARCHAR(30),
    PROVINCIA INTEGER NOT NULL,
    FECHA_NACIMIENTO DATE,
    NUM_HIJOS SMALLINT,
    ESTADO_CIVIL CHAR(1),
    SALARIO DECIMAL(7,2),
    JUBILADO BOOLEAN,
    FOTO VARCHAR(30),
    CONSTRAINT ID_PERSONA_PK PRIMARY KEY (ID),
    CONSTRAINT PROV_PERSONA_FK FOREIGN KEY (PROVINCIA) REFERENCES PROVINCIA (ID)
);
```

Creación de las tablas

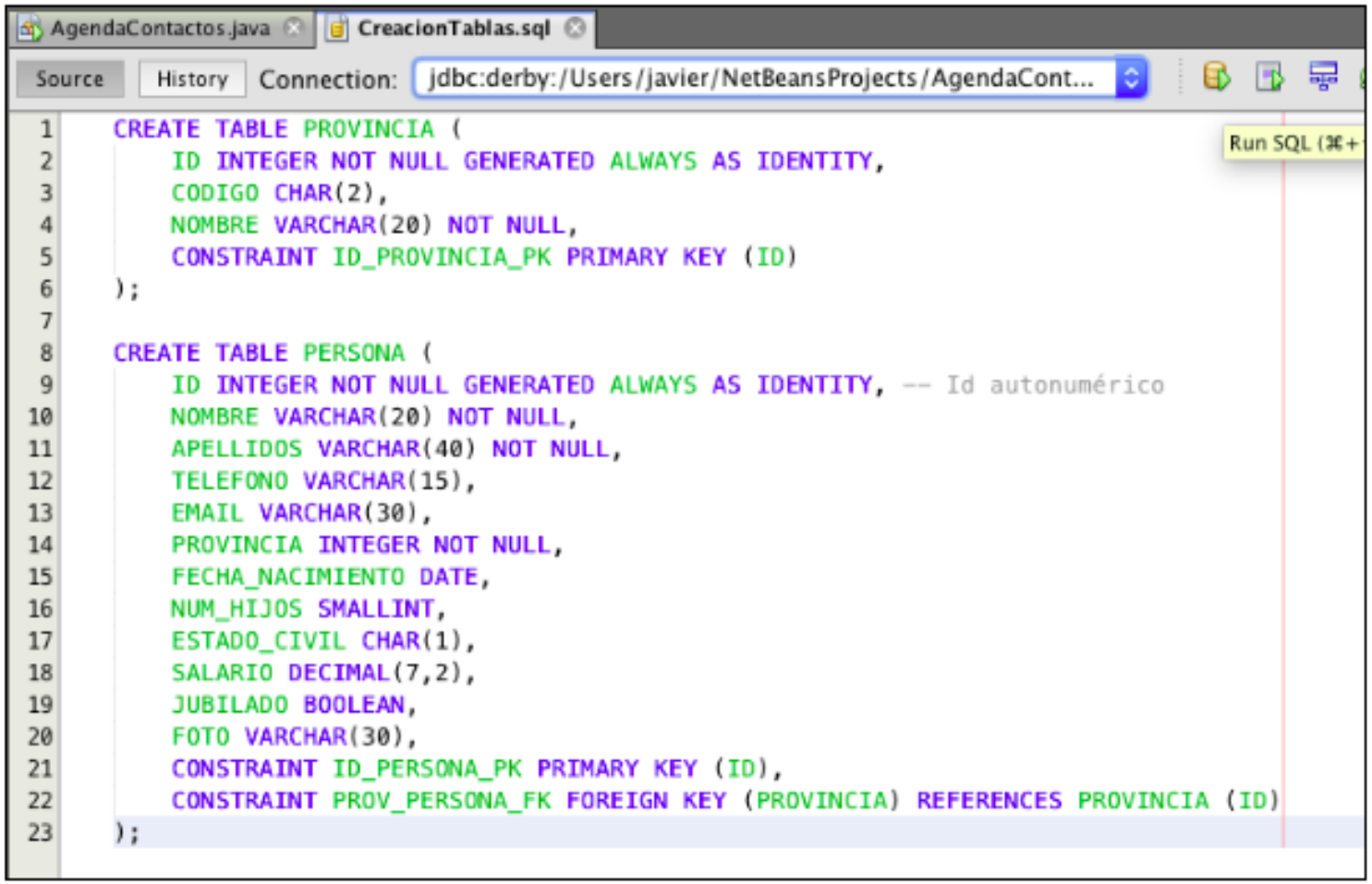
Puedes ver que cada tabla dispone de un campo **ID como identificador** para cada registro, que es **autonumérico** y que se utilizará como **clave primaria** de cada tabla. Ahí puedes ver que la declaración de este tipo autonumérico (GENERATED ALWAYS AS IDENTITY) es diferente a MySQL, por lo que si la base de datos se desea conectar a un servidor MySQL se deben modificar algunos aspectos del lenguaje SQL como ese.

Observa también que no sólo se han creado **columnas de tipo** VARCHAR, sino también algunas de tipo numérico con distintos tamaños (INTEGER, SMALLINT, DECIMAL, etc), así como la columna JUBILADO de tipo BOOLEAN o la fecha de nacimiento de tipo DATE, para poder conocer cómo se pueden gestionar esos otros tipos de datos desde Java. Consulta la documentación de los tipos de datos de Derby para conocerlos mejor.

La **relación entre las 2 tablas** se lleva a cabo mediante la clave foránea PROVINCIA de la tabla PERSONA que hará referencia a un valor numérico entero correspondiente a una ID de la tabla PROVINCIA. Consulta la documentación de la cláusula CONSTRAINT de Derby para conocer mejor su uso.

Creación de las tablas

Para **ejecutar el script SQL** anterior, haz clic sobre el **botón *Run SQL*** de la barra de herramientas asociada a este archivo.



Creación de las tablas

Como resultado de la ejecución, deberían aparecer, en la pestaña ***Services***, las tablas que se han creado, y si vas desplegando el contenido de cada una, verás los nombres de las columnas.