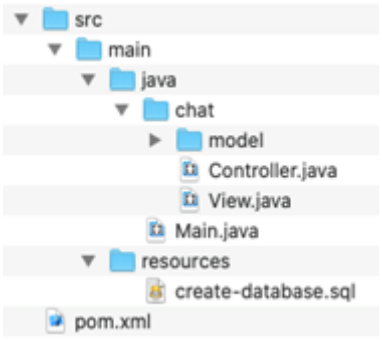
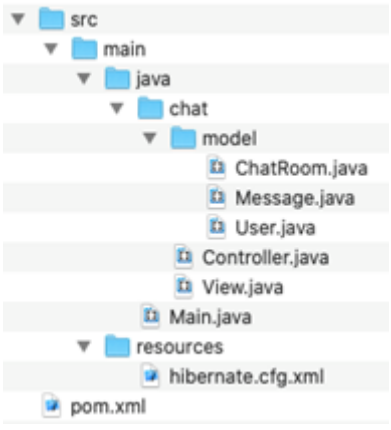
Normativa

1. La práctica se realizará en grupos de dos alumnos.
2. Se debe entregar, a través de la tarea creada en `Moodle`, un fichero `zip` que contenga:
   - Un fichero `txt` indicando los datos de los alumnos que han realizado la práctica.
   - Un directorio llamado `connector-j` que contenga el código fuente realizado para el Apartado 1. Dentro de este directorio deberá respetarse la estructura de ficheros y directorios del proyecto descargado. La siguiente imagen muestra la **ESTRUCTURA PERMITIDA** del directorio `connector-j`:



   - Un directorio llamado `hibernate` que contenga el código fuente realizado para el Apartado 2. Dentro de este directorio deberá respetarse la estructura de ficheros y directorios del proyecto descargado. La siguiente imagen muestra la **ESTRUCTURA PERMITIDA** del directorio `hibernate`:



   - La práctica deberá ser entregada a través de `Moodle` por un ÚNICO MIEMBRO de la pareja.
   - La fecha límite de entrega de la práctica deberá ser consultada en la plataforma `Moodle`.
   - Esta práctica podrá ser objeto de defensa individualizada ya sea mediante examen escrito u oral.

Aplicación de chat

El Gabinete de Tele-Educación (GATE) de la Universidad Politécnica de Madrid (UPM) desea poner a disposición de sus estudiantes un nuevo servicio de chat. El funcionamiento de este servicio será simple. Los usuarios accederán al sistema indicando su nombre de usuario (no requiere registro previo) y, a partir de ese momento podrán realizar las siguientes acciones:

1. Crear nuevas salas de chat.
2. Consultar los mensajes de las salas de chat existentes.
3. Enviar mensajes a las salas de chat.
4. Borrar sus mensajes de la sala de chat.

Con el fin de estudiar el uso que realizan los estudiantes de este nuevo servicio, la UPM encargó a la empresa CompuGlobalHiperSuperMegaNet S.L. el desarrollo de un prototipo funcional. Este prototipo debería funcionar de manera experimental durante los 6 primeros meses de 2021 y, posteriormente pasaría a desarrollarse la aplicación definitiva. Por desgracia, y por razones ajenas a la UPM, el prototipo no llegó a completarse.

El GATE ha contactado con las Escuela Técnica Superior de Ingeniería de Sistemas Informáticos (ETSISI) para que analicen el estado actual del prototipo y decidan si puede terminarse dentro de los plazos planificados para el proyecto. Como consecuencia de este análisis, se ha detectado que únicamente falta por completar la conexión de la aplicación con la base de datos, por lo que ha decidido proponer a los alumnos de la asignatura de Bases de Datos que finalicen el desarrollo.

1. Implementación mediante MySQL Connector/J 8.0

En este apartado deberá completarse el desarrollo del sistema de chat explicado anteriormente haciendo uso de `MySQL Connector/J 8.0`. Para ello, deberá completar las siguientes tareas:

1. Descargue el código fuente del proyecto del `Moodle` de la asignatura y cárguelo en un proyecto `Maven` de `IntelliJ`. Si lo prefiere, puede clonar el repositorio del proyecto alojado en `GitHub`:

   https://github.com/carloscamachogom/practica-conectorj-curso-20-21

2. Inicialice el servidor de `MariaDB` mediante `docker` tal y como hizo en la práctica 2. Anote la contraseña establecida para el administrador y el puerto en el que se 'levanta' el servicio.

3. Utilizando `phpMyAdmin`, cree la estructura de la base de datos mediante el siguiente código `SQL` (podrá encontrar este mismo código en el directorio src/main/resources):

```
CREATE SCHEMA chat
DEFAULT CHARACTER SET utf8
COLLATE utf8_spanish2_ci;

USE chat;

CREATE TABLE users (
  id       BIGINT UNIQUE NOT NULL AUTO_INCREMENT,
  username VARCHAR(50) NOT NULL,
  PRIMARY KEY (id)
```

```
);
CREATE TABLE chatrooms (
  id        BIGINT UNIQUE NOT NULL AUTO_INCREMENT,
  name      VARCHAR(50) NOT NULL,
  createdBy BIGINT NOT NULL,
  PRIMARY KEY (id),
  CONSTRAINT
      FOREIGN KEY (createdBy)
      REFERENCES users (id)
);

CREATE TABLE messages (
  id        BIGINT UNIQUE NOT NULL AUTO_INCREMENT,
  text      VARCHAR(255) NOT NULL,
  chatRoom  BIGINT NOT NULL,
  createdBy BIGINT NOT NULL,
  ts        TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id),
  CONSTRAINT
      FOREIGN KEY (chatRoom)
      REFERENCES chatrooms (id),
  CONSTRAINT
       FOREIGN KEY (createdBy)
       REFERENCES users (id)
);
```

4. Analice el código del proyecto y complételo siguiendo las instrucciones de los comentarios `@TODO`. Únicamente deberá modificar la codificación de la clase `chat.Controller`. No será necesario añadir nuevas clases, ni modificar el resto de las clases de las que se compone la aplicación.

5. Descargue las dependencias del proyecto mediante Maven y compruebe que la codificación añadida en el apartado 4 es correcta.

2. Implementación mediante Hibernate

En este apartado deberá codificarse la misma aplicación de chat que en el Apartado 1 pero haciendo uso del `ORM Hibernate` en lugar de `MySQL Connector/J 8.0`. Para ello, deberá completar las siguientes tareas:

1. Descargue el código fuente del proyecto del `Moodle` de la asignatura y cárguelo en un proyecto `Maven` de `IntelliJ`. Si lo prefiere, puede clonar el repositorio del proyecto alojado en `GitHub`:

   https://github.com/carloscamachogom/practica-hibernate-curso-20-21.git

2. Inicialice el servidor de `MariaDB` mediante `docker` tal y como realizó en la práctica 2. Anote la contraseña establecida para el administrador y el puerto en el que se "levanta" el servicio.

3. Utilizando `phpMyAdmin`, cree el *schema* de la base de datos sobre la que trabajará `Hibernate`. Puede hacerlo con la siguiente sentencia `SQL`:

```
CREATE SCHEMA chat2
DEFAULT CHARACTER SET utf8
COLLATE utf8_spanish2_ci;
```

4. Analice el código del proyecto y complételo siguiendo las instrucciones de los comentarios `@TODO`. Deberá anotar las clases `chat.model.User`, `chat.model.Message` y `chat.model.ChatRoom` para que puedan ser usadas por `Hibernate` y que éste cree la estructura de la base de datos requerida por la aplicación. Deberá también completar los métodos de la clase `chat.Controller`. No será necesario añadir nuevas clases, ni modificar el resto de las clases de las que se compone la aplicación.

5. Compruebe que la configuración de la conexión a la base de datos del fichero `hibernate.cfg.xml` es correcta.
6. Descargue las dependencias del proyecto mediante `Maven` y compruebe la aplicación ha sido codificada de manera adecuada.

Se pide:

1. Completar las tareas definidas en los Apartados 1 y 2 del presente documento.
2. Codificar la aplicación requerida mediante `MySQL Connector/J 8.0`.
3. Codificar la aplicación requerida mediante `Hibernate`.