==========
=== CONFIGURACIÓN DEL ENTORNO DE TRABAJO ===
==========

1 - Crear la base de datos:
    * Insertar el script del archivo "jsp_libreria.sql"
    * nombre bbdd: jsp_libreria
    * tablas: 3
        - libros
        - usuarios
        - alquiler

2 - Crear el proyecto y la estructura de carpetas y paquetes:
    * Dynamic Web Project
        - nombre: Proyecto_01_Servlets-Libreria

    * En la carpeta "src/main/webapp" crear las carpetas:
        - js
        - css
        - img

    * Meter el archivo de imagen para la pantalla de LOGIN "login.jpg"

    * En la carpeta "src/main/java" crear los paquetes:
        - connection
        - controller
        - beans
        - servlets
        - test

==========
=== Sección | JAVA EE ===
==========

3 - En el paquete "beans" crear las clases:
    * Libro
    * Usuario
    * Alquiler

4 - En la carpeta "src/main/webapp/WEB-INF/lib":
    * copiar el conector "mysql-connector-java-8.0.28.jar"

5 - Agregar el conector al "Java Build Path" del proyecto:
    * Botón derecho en el nombre del proyecto
    * Properties
    * Java Build Path
    * Libraries/Classpath
        - Add JARs...
        - Seleccionar el proyecto
        - src/main/webapp/WEB-INF/lib
        - mysql-connector-java-8.0.28.jar
        - OK
        - Apply and Close

6 - En el paquete "connection":
    * Crear clase "DBConnection.java"

7 - En el paquete "test"
    * Crear clase "OperacionesBD.java"

****
****
NOTA: PROCEDIMIENTO/ARQUITECTURA DE LA APLICACIÓN

1º Crear JS (método/funcionalidad).
2º Crear Servlet (recibe petición http) y llama
al método del Controlador.
3º Desde JS llama al Servlet.
4º El Servlet llama a un método del Controlador.
5º El controlador realiza la lógica/método y
se comunica con la bbdd.
****
****

==========
=== Sección | LOGIN DE USUARIOS ===
==========
El proceso de LOGIN consiste en lo siguiente:
- En la pantalla de LOGIN
- El usuario introduce: "usuario" y "contraseña"
- Con esos datos se hace una comunicación AJAX entre
cliente y servidor pasandole dichos parámetros
- Se comunica con un "servlet" que lee esos parámetros
y va a invocar a un método en el "controlador"
- El controlador va a consultar en la bbdd si existe
un usuario con ese nombre y contraseña.
- Si es correcto, va a devolver todos los datos
de ese usaurio
- Si NO es correcto, devuelve false

8 - Crear la pantalla de LOGIN de la aplicación,
en la carpeta "src/main/webapp":
* Crear el archivo "index.html"

9 - Crear estilos para la pantalla de LOGIN,
en la carpeta "src/main/webapp/css":
* Crear el archivo "styles.css"

10 - En el paquete "controller":
* Crear interface "IUsuarioController.java"
* Se definen todos los métodos relacionados
con la funcionalidad del usuario

12 - En la carpeta "src/main/webapp/WEB-INF/lib"
* Copiar archivo "gson-2.8.9.jar"
* Que está en D:/www
* Una vez copiado ya automáticamente se puede instanciar
en el controlador sin problema

13 - En el paquete "controller":
* Crear clase "UsuarioController.java"
* Va a implementar la interface "IUsuarioController.java"

14 - En el paquete "servlets":
* Crear el servlet "ServletUsuarioLogin.java"

****
Cómo leer los parámetros del formulario en JQuery,
como hacer la petición AJAX pasándole esos dos valores
del formulario. Y cómo procesar la respuesta que está
enviando el Servlet "ServletUsuarioLogin" ante la comprobación
de si existe usuario con el "username" y "contrasena" recibidos.
****

15 - En la carpeta "src/main/webapp/js":
* Crear archivo javascript "index.js"
* Referenciar este archivo "index.js" en el "index.html"
* Para hacer la llamada AJAX

16 - En la carpeta "src/main/webapp":
* Crear archivo "home.html"

17 - En la carpeta "src/main/webapp":
    * Crear archivo "register.html"

18 - En la tabla "usuarios" de la bbdd:
    * Crear un usuario para pruebas
        username: salva
        contraseña: 1234
        nombre: salvador
        apellidos: belloso
        email: salva@gmail.com
        saldo: 10
        premium: 1

19 - En la carpeta "src/main/webapp":
    * Ir al archivo "index.html"
    * Incluir validación HTML en cada
        campo del formulario de login
        con el atributo "required" en
        cada "input".

****
****
NOTA IMPORTANTE !!!

Al trabajar con chrome como navegador deshabilitar
cuando se está trabajando con este proyecto todas
las extensiones de chrome, para evitar el mensaje
por consola de:
"Unchecked runtime.lastError: The message port closed before a
response was received."
****
****

==========
=== Sección | REGISTRO DE USUARIOS ===
==========

20 - En la carpeta "src/main/webapp":
    * Ir al archivo "register.html"
    * Crear el diseño html de la pantalla
        de registro.
    * Formulario de registro.

21 - En la carpeta "src/main/webapp":
    * Ir al archivo "js/index.js"
    * Crear validación del registro mediante
        la función "registrarUsuario()".

22 - Insertar usuarios en la bbdd:
    * Ir al archivo "js/index.js"
    * Crear petición AJAX al "ServletUsuarioRegister"
        para insertar los usuarios creados,
        registrarUsuario().

23 - En el paquete "servlets":
    * Crear "ServletUsuarioRegister", que va a invocar
        al controlador "UsuarioController"

24 - En la interface "IUsuarioController":
    * Añadir el método "register()"

25 - En el controlador "UsuarioController":
    * Implementar método de la interface "register()"

****
Crear la comunicación AJAX que es la que va a realizar la
petición http al servlet y que va a procesar lo que devuelve
la inserción del método del controlador que inserta el usuario.

NOTA IMPORTANTE:
Cuando estemos ejecutando el proyecto y nos vayamos a loguear,
pulsar CTRL+F5 para limpiar la caché del navegador por si hay
algún error.
****

26 - En la carpeta "src/main/webapp":
    * Ir al archivo "js/index.js"
    * Implementar método "registrarUsuario()".

==========
=== Sección | GESTIÓN DE LIBROS ===
==========

27 - En la carpeta "src/main/webapp":
    * Ir al archivo "home.html"
    * Diseñar la pantalla principal
        barra de navegación.
    * Diseñar la pantalla principal
        tabla para libros.

****
Cargar los datos del usuario cuando accedemos a la
pantalla de inicio. Obtener todos los datos del usuario
para realizar las operaciones, por ejemplo cargar el
saldo disponible o asociar reservas...
****

28 - En la carpeta "src/main/webapp/js":
    * Crear archivo "main.js"
    * Crear funciones con las operaciones
        AJAX a realizar.

29 - En el paquete "servlets":
    * Crear servlet "ServletUsuarioPedir"

30 - En la interface "IUsuarioController":
    * Crear el método "pedir()"

31 - En el controlador "UsuarioController:
    * Implementar el método "pedir()" que
        se creó en el interface "IUsuarioController".

32 - En el archivo "main.js":
    * Crear el método para traer
        todos los libros disponibles
    * Crear el método para mostrar
        todos los libros disponibles
        en la tabla html

33 - En el paquete "servlets":
    * Crear servlet "ServletLibroListar"

34 - En el paquete "controller":
    * Crear interface "ILibroController"
    * Crear métodos

35 - En el paquete "controller":
    * Crear clase "LibroController"

36 - En el archivo "styles.css":
    * Crear estilos CSS
        para ordenar por genero
        en la tabla html

37 - En el archivo "main.js":
* Crear métodos/funcionalidad
para ordenar los libros
por género.

38 - En el archivo "main.js":
* Crear la funcionalidad del
botón "alquilarLibro()"

39 - En el paquete "servlets":
* Crear el servlet "ServletLibroAlquilar.java"

40 - En el paquete "controller":
* Ir a la interface "ILibroController.java"
* Crear método abstracto "alquilar()"

41 - En el controlador "LibroController.java":
* Implementar el método "alquilar()"
de la interface.

42 - En el paquete "controller":
* Ir a la interface "ILibroController.java"
* Crear método abstracto "modificar()"

43 - En el controlador "LibroController.java":
* Implementar el método "modificar()"
de la interface.

44 - En el archivo "main.js":
* Crear la funcionalidad del
método "restarDinero()"

45 - En el paquete "servlets":
* Crear el servlet "ServletUsuarioRestarDinero.java"

46 - En el paquete "controller":
* Ir a la interface "IUsuarioController.java"
* Crear método abstracto "restarDinero()"

43 - En el controlador "UsuarioController.java":
* Implementar el método "restarDinero()"
de la interface.

==========
=== Sección | PERFIL DEL USUARIO ===
==========

44 - En la carpeta "src/main/webapp":
* Crear archivo "profile.html"
Para la pantalla del
historial de reservas
del usuario

45 - En la carpeta "src/main/webapp/js":
* Crear archivo "profile.js"
* Hacer petición ajax al servlet
"ServletUsuarioPedir"
* Crear la functión getAlquiladas()
con la llamada AJAX
para traer los alquileres
del usuario.

46 - En el paquete "servlets":
* Crear servlet "ServletAlquilarListar"

47 - En el paquete "controller":
* Crear la interface "IAlquilerController"
para crear los métodos abstractos.
* Crear el controlador "AlquilerController"
que va a implementar esta interface.

48 - En el archivo "profile.js":
    * Implementar función "mostrarHistorial()".

49 - En el archivo "profile.js":
    * Implementar función "devolverLibro()".

50 - En le paquete "servlets":
    * Crear servlet "ServletLibroDevolver".

51 - En el paquete "controller":
    * Crear la interface "ILibroController"
        crear el método abstracto "devolver()".
    * Crear el método abstracto "sumarCantidad()".
    * Implementar los métodos en el controlador
        "LibroController()".

52 - En el archivo "profile.js":
    * Implementar función "modificarUsuario()".

53 - En el paquete "servlets":
    * Crear servlet "ServletUsuarioModificar".

54 - En el archivo "IUsuarioController.java":
    * Crear el método abstracto "modificar()".

55 - En el archivo "UsuarioController.java":
    * Implementar método "modificar()".
    * Implementar método "eliminar()".

56 - En el archivo "profile.js":
    * Implementar función "eliminarCuenta()".

57 - En el paquete "servlets":
    * Crear servlet "ServletUsuarioEliminar".

58 - En el archivo "IUsuarioController.java":
    * Crear el método abstracto "verCopias()".
    * Crear el método abstracto "devolverLibros()".
    * Crear el método abstracto "elininar()".

59 - En el archivo "UsuarioController.java":
    * Implementar método "verCopias()".
    * Implementar método "devolverLibros()".
    * Implementar método "eliminar()".