Grand Hotel Manager

MANUAL DE PROGRAMADOR

Guía del software para programadores

Junio de 2024
Version No. 1.0

Tabla de contenidos

1. Introducción....3

1.1. Introducción, objetivos....3

Descripción del Proyecto....3

Tecnologías Utilizadas....3

Objetivos del Manual....4

2. Estructura del proyecto....5

Organización de Directorios....5

Componentes Principales....7

3. Base de datos....9

Modelo Entidad-Relación....9

Estructura de Tablas....9

Índices y Claves Primarias....12

Relaciones entre Tablas....13

4. Servidor....14

Instalación de Apache, PHP, phpMyAdmin y MySQL....14

Paso 1: Actualizar los paquetes del sistema....14

Paso 2: Instalar Apache....14

Paso 3: Instalar PHP y extensiones necesarias....14

Paso 4: Instalar MySQL (MariaDB)....14

Paso 5: Configurar MySQL....15

Paso 6: Instalar phpMyAdmin....15

Configuración de Apache para Aceptar Conexiones Externas....15

Paso 1: Editar el archivo de configuración de Apache....15

Paso 2: Cambiar la dirección de escucha de Listen 80 a Listen 0.0.0.0:80....15

Paso 3: Reiniciar Apache para aplicar los cambios....15

Estructura del Servidor....16

1. Introducción

1.1. Introducción, objetivos

Bienvenido al manual del programador para el sistema de gestión hotelera del Grand Hotel. Este manual está diseñado para proporcionar una guía detallada sobre la arquitectura, configuración, desarrollo y mantenimiento del sistema, facilitando a los desarrolladores la comprensión y gestión del proyecto.

Descripción del Proyecto

El sistema de gestión hotelera del Grand Hotel es una solución integral desarrollada para optimizar la administración operativa del hotel. Esta solución incluye tanto una aplicación de escritorio desarrollada en Java como un servidor backend basado en Linux, que utiliza tecnologías como Apache, PHP, phpMyAdmin y MySQL (MariaDB).

Tecnologías Utilizadas

- **Backend**:
  - **Sistema Operativo**: Linux
  - **Servidor Web**: Apache
  - **Lenguaje de Programación**: PHP
  - **Administración de Base de Datos**: phpMyAdmin
  - **Base de Datos**: MySQL (MariaDB)
- **Frontend**:
  - **Lenguaje de Programación**: Java con OpenJDK 22.0.1
  - **Bibliotecas Utilizadas**:
    - `apache.httpcomponents.httpclient`
    - `apache.httpcomponents.httpmime`
    - `apache.maven.plugins.shade.plugin`
    - `fontawesomefx-commons-9.1.2`

    - glassfish.tyrus.bundles.standalone.client
    - glassfish.tyrus.bundles.standalone.client1
    - google.code.gson
    - net.sf.jasperreports

Objetivos del Manual

Este manual tiene como objetivo proporcionar a los desarrolladores la información necesaria para:

- Comprender la estructura y arquitectura del sistema.
- Configurar el entorno de desarrollo.

2. Estructura del proyecto

El proyecto de gestión hotelera está organizado en una estructura de directorios clara y lógica para facilitar su desarrollo, mantenimiento y escalabilidad. A continuación, se detalla la estructura de directorios y la función de cada clase y recurso en el proyecto.

Organización de Directorios

1. **`src/main/java/es/iesgrancapitan/proyectohotelfx/controllers`**: Este directorio contiene todas las clases controladoras que manejan la lógica de la interfaz de usuario y la interacción con los usuarios.
    - **Clases de Controladores**:
        - `AplicacionController`: Controlador principal que maneja el menú principal y las navegaciones entre las diferentes vistas.
        - `CrearClienteController`: Controlador para la vista de creación de nuevos clientes.
        - `CrearEmpleadoController`: Controlador para la vista de creación de nuevos empleados.
        - `CrearHabitacionController`: Controlador para la vista de creación de nuevas habitaciones.
        - `CrearUsuarioController`: Controlador para la vista de creación de nuevos usuarios.
        - `HabitacionViewController`: Controlador para la vista de gestión de habitaciones.
        - `HospedarViewController`: Controlador para la vista de hospedaje de clientes.
        - `InfoClienteController`: Controlador para la vista de información detallada de clientes.
        - `InfoDeReservaController`: Controlador para la vista de información detallada de reservas.
        - `InfoEmpleadoController`: Controlador para la vista de información detallada de empleados.
        - `InfoPedidoController`: Controlador para la vista de información detallada de pedidos.
        - `InfoReservasViewController`: Controlador para la vista de gestión de reservas.

        - `LoginController`: Controlador para la vista de inicio de sesión.
2. **`src/main/java/es/iesgrancapitan/proyectohotelfx`**: Este directorio contiene las clases principales de la aplicación que no están directamente relacionadas con el control de las vistas.
    - **Clases Principales**:
        - `AlertBox`: Clase utilitaria para mostrar cuadros de alerta en la aplicación.
        - `Cliente`: Clase que representa a un cliente.
        - `Config`: Clase que maneja la configuración de la aplicación.
        - `Empleado`: Clase que representa a un empleado.
        - `Habitacion`: Clase que representa a una habitación.
        - `Launcher`: Clase con un método `main` alternativo para ejecutar la aplicación cuando se crea el artifact con el jar.
        - `LoginApplication`: Clase principal con el método `main` que inicia la aplicación.
        - `PasswordEncryptor`: Clase utilitaria para la encriptación de contraseñas.
        - `Pedido`: Clase que representa un pedido realizado por un cliente.
        - `Producto`: Clase que representa un producto disponible para pedido.
        - `ReportGenerator`: Clase utilitaria para la generación de reportes.
        - `Reserva`: Clase que representa una reserva.
        - `UserSession`: Clase que maneja la sesión del usuario.
3. **`src/main/resources/es/iesgrancapitan/proyectohotelfx`**: Este directorio contiene los archivos FXML que definen las vistas de la aplicación y los archivos de estilo CSS.
    - **Archivos FXML**:
        - `crear-cliente.fxml`: Vista para la creación de nuevos clientes.
        - `crear-empleado.fxml`: Vista para la creación de nuevos empleados.
        - `crear-habitacion.fxml`: Vista para la creación de nuevas habitaciones.
        - `crear-usuario.fxml`: Vista para la creación de nuevos usuarios.
        - `HabitacionView.fxml`: Vista de gestión de habitaciones.

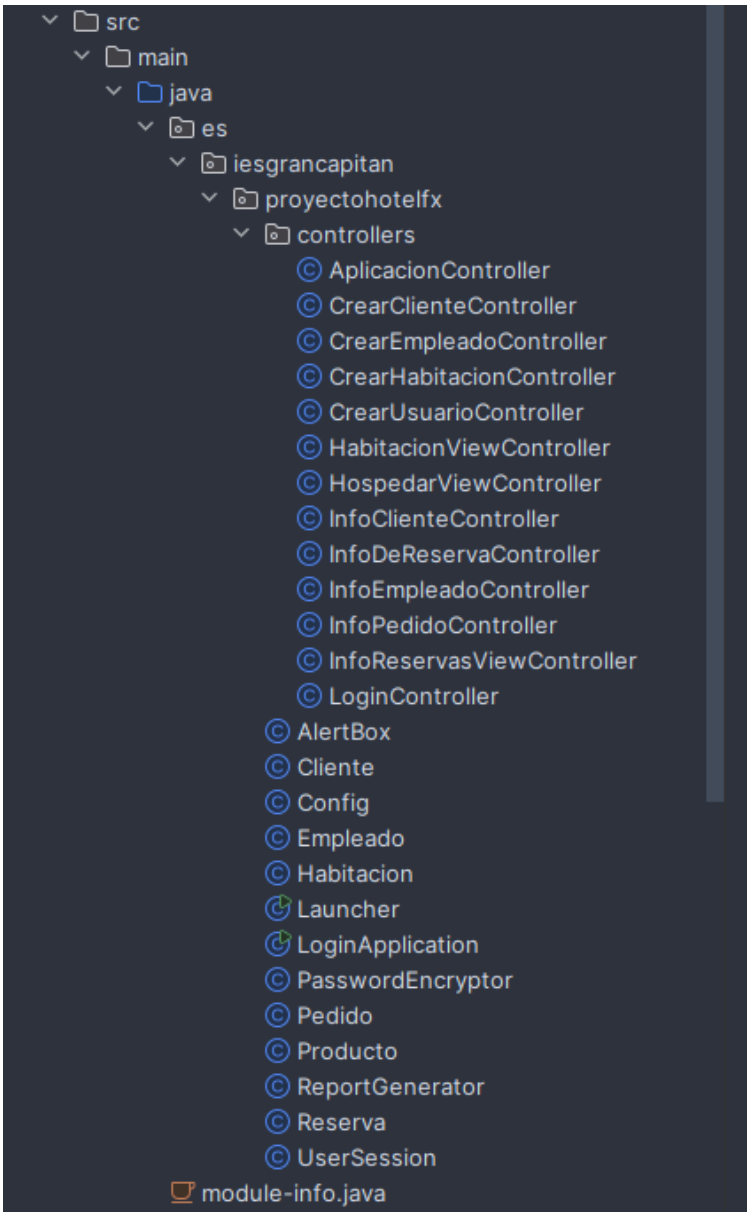
        - `hospedar-view.fxml`: Vista de hospedaje de clientes.
        - `info-cliente.fxml`: Vista de información detallada de clientes.
        - `info-de-reserva.fxml`: Vista de información detallada de reservas.
        - `info-empleado.fxml`: Vista de información detallada de empleados.
        - `info-pedido.fxml`: Vista de información detallada de pedidos.
        - `info-reservas-view.fxml`: Vista de gestión de reservas.
        - `login-view.fxml`: Vista de inicio de sesión.
        - `user-app.fxml`: Vista principal de la aplicación después del inicio de sesión.
    - **Archivos CSS**:
        - `login-style.css`: Estilos para la vista de inicio de sesión.
        - `style.css`: Estilos generales para la aplicación.
4. **`src/main/resources/images`**: Este directorio contiene las imágenes utilizadas en la aplicación.
    - **Imágenes**:
        - `hotel.jpg`: Imagen representativa del hotel.
5. **`src/main/resources/jasper`**: Este directorio contiene los archivos de reportes Jasper.
    - **Reportes Jasper**:
        - `Factura2GrandHotel.jrxml`: Plantilla de reporte para las facturas.
6. **`src/main/resources`**: Este directorio contiene archivos de configuración.
    - **Archivos de Configuración**:
        - `config.properties`: Archivo de propiedades de configuración que incluye parámetros como `server.ip=34.175.164.212` (que se puede cambiar según sea necesario).

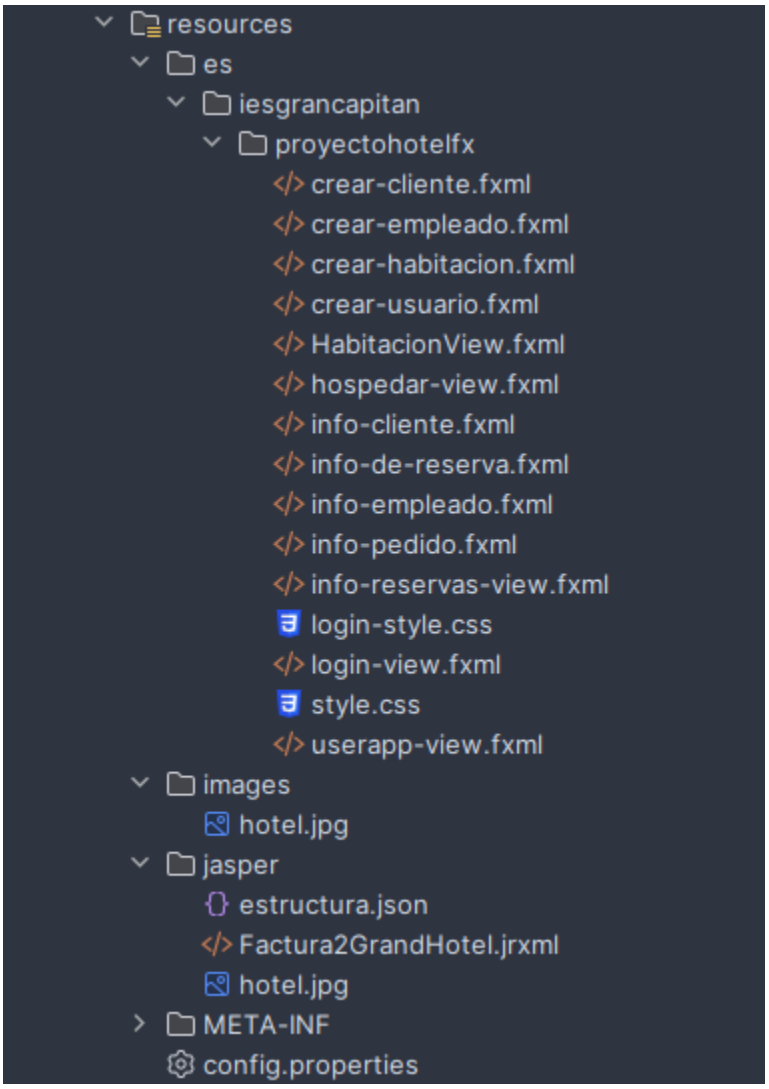
Componentes Principales

- **Clase Principal (`LoginApplication`)**: Esta clase contiene el método `main` que inicia la aplicación de escritorio.
- **Controlador Principal (`AplicacionController`)**: Esta clase maneja el menú principal de la aplicación y la navegación a las diferentes vistas.

- **Configuración (`Config`)**: Clase que carga y gestiona las configuraciones del archivo `config.properties`.
- **Generador de Reportes (`ReportGenerator`)**: Clase encargada de generar reportes utilizando JasperReports.
- **Encriptador de Contraseñas (`PasswordEncryptor`)**: Clase utilitaria para la encriptación de contraseñas.

Esta organización de directorios y clases asegura una estructura clara y mantenible para el proyecto, facilitando la colaboración entre desarrolladores y permitiendo una fácil navegación y comprensión del código fuente.

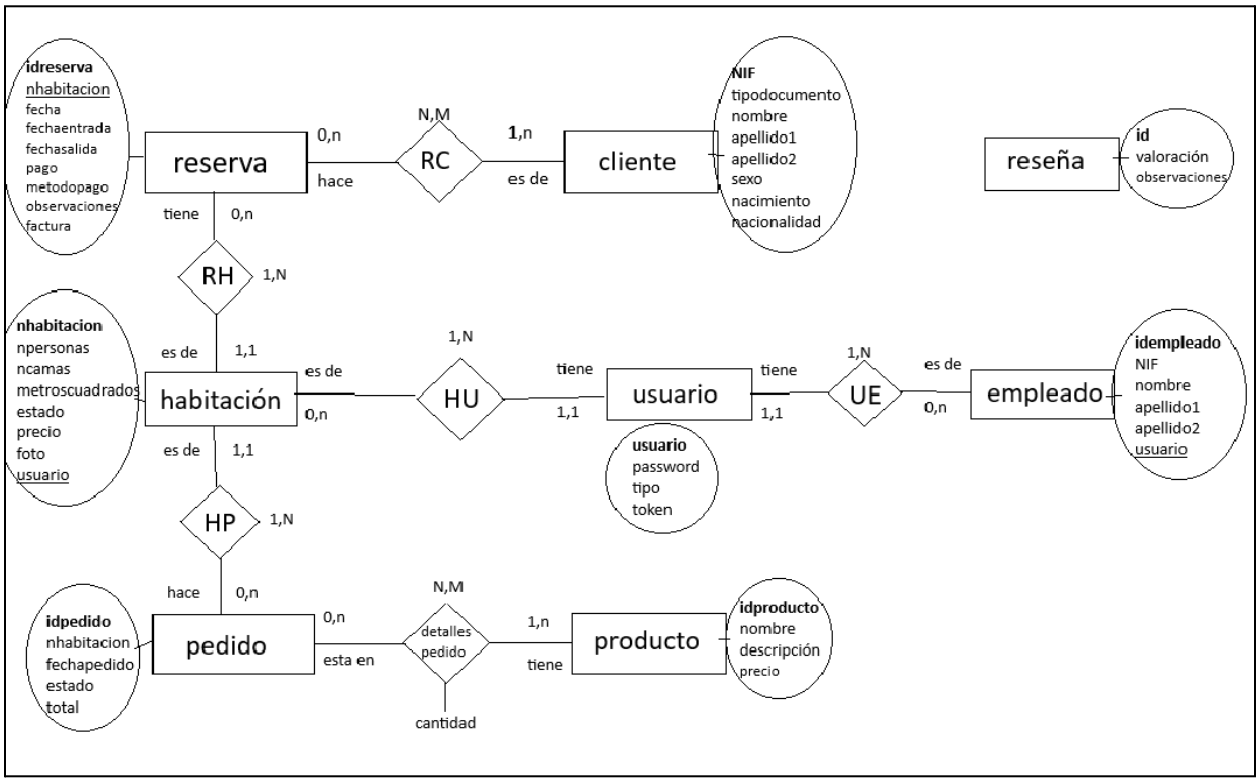




3. Base de datos

La base de datos del proyecto de gestión hotelera está implementada en MariaDB y consta de varias tablas relacionadas que almacenan la información necesaria para gestionar clientes, empleados, habitaciones, pedidos, productos, reseñas, reservas y usuarios. A continuación, se detalla la estructura de la base de datos y las relaciones entre las tablas.

Modelo Entidad-Relación



Estructura de Tablas

1. **Tabla `cliente`**:
    - Almacena la información de los clientes.
    - **Campos**:
        - `NIF` (varchar(20), NOT NULL): Identificador único del cliente.
        - `tipodocumento` (varchar(45), DEFAULT NULL): Tipo de documento del cliente.
        - `nombre` (varchar(45), DEFAULT NULL): Nombre del cliente.

        - `apellido1` (varchar(45), DEFAULT NULL): Primer apellido del cliente.
        - `apellido2` (varchar(45), DEFAULT NULL): Segundo apellido del cliente.
        - `sexo` (varchar(45), DEFAULT NULL): Sexo del cliente.
        - `nacimiento` (varchar(45), DEFAULT NULL): Fecha de nacimiento del cliente.
        - `nacionalidad` (varchar(45), DEFAULT NULL): Nacionalidad del cliente.
2. **Tabla `detallespedido`**:
    - Almacena los detalles de los pedidos realizados.
    - **Campos**:
        - `iddetalle` (int(11), NOT NULL, AUTO_INCREMENT): Identificador único del detalle.
        - `idpedido` (int(11), DEFAULT NULL): Identificador del pedido.
        - `idproducto` (int(11), DEFAULT NULL): Identificador del producto.
        - `cantidad` (int(11), NOT NULL): Cantidad del producto pedido.
3. **Tabla `empleado`**:
    - Almacena la información de los empleados.
    - **Campos**:
        - `idempleado` (int(11), NOT NULL, AUTO_INCREMENT): Identificador único del empleado.
        - `NIF` (varchar(9), NOT NULL): Identificador fiscal del empleado.
        - `nombre` (varchar(45), DEFAULT NULL): Nombre del empleado.
        - `apellido1` (varchar(45), DEFAULT NULL): Primer apellido del empleado.
        - `apellido2` (varchar(45), DEFAULT NULL): Segundo apellido del empleado.
        - `usuario` (varchar(45), DEFAULT NULL): Usuario asociado al empleado.
4. **Tabla `habitacion`**:
    - Almacena la información de las habitaciones.
    - **Campos**:
        - `nhabitacion` (int(11), NOT NULL): Número de la habitación.
        - `npersonas` (varchar(45), DEFAULT NULL): Capacidad de personas.

        - `ncamas` (varchar(45), DEFAULT NULL): Número de camas.
        - `metroscuadrados` (varchar(45), DEFAULT NULL): Metros cuadrados de la habitación.
        - `estado` (varchar(20), NOT NULL): Estado de la habitación (disponible, ocupada, etc.).
        - `precio` (float, NOT NULL): Precio por noche.
        - `foto` (longblob, DEFAULT NULL): Foto de la habitación.
        - `usuario` (varchar(45), DEFAULT NULL): Usuario responsable de la habitación.
5. **Tabla `pedido`**:
    - Almacena la información de los pedidos realizados por los clientes.
    - **Campos**:
        - `idpedido` (int(11), NOT NULL, AUTO_INCREMENT): Identificador único del pedido.
        - `nhabitacion` (int(11), DEFAULT NULL): Número de la habitación que realizó el pedido.
        - `fechapedido` (datetime, NOT NULL): Fecha y hora del pedido.
        - `estado` (varchar(20), NOT NULL): Estado del pedido (procesado, completado, etc.).
        - `total` (float, NOT NULL): Total del pedido en euros.
6. **Tabla `producto`**:
    - Almacena la información de los productos disponibles para pedido.
    - **Campos**:
        - `idproducto` (int(11), NOT NULL, AUTO_INCREMENT): Identificador único del producto.
        - `nombre` (varchar(100), NOT NULL): Nombre del producto.
        - `descripcion` (varchar(255), DEFAULT NULL): Descripción del producto.
        - `precio` (float, NOT NULL): Precio del producto.
7. **Tabla `resena`**:
    - Almacena las reseñas realizadas por los clientes.
    - **Campos**:
        - `id` (int(11), NOT NULL, AUTO_INCREMENT): Identificador único de la reseña.
        - `valoracion` (float, NOT NULL): Valoración otorgada.

        - `observaciones` (varchar(200), NOT NULL): Observaciones de la reseña.
8. **Tabla `reserva`**:
    - Almacena la información de las reservas realizadas.
    - **Campos**:
        - `idreserva` (int(11), NOT NULL, AUTO_INCREMENT): Identificador único de la reserva.
        - `nhabitacion` (int(11), DEFAULT NULL): Número de la habitación reservada.
        - `fecha` (date, NOT NULL): Fecha de la reserva.
        - `fechaentrada` (date, DEFAULT NULL): Fecha de entrada.
        - `fechasalida` (date, DEFAULT NULL): Fecha de salida.
        - `pago` (float, NOT NULL): Pago realizado.
        - `metodopago` (varchar(32), NOT NULL): Método de pago utilizado.
        - `observaciones` (varchar(255), NOT NULL): Observaciones de la reserva.
        - `factura` (mediumblob, DEFAULT NULL): Factura generada para la reserva.
9. **Tabla `reservacliente`**:
    - Relaciona las reservas con los clientes.
    - **Campos**:
        - `idreserva` (int(11), NOT NULL): Identificador de la reserva.
        - `nifcliente` (varchar(20), NOT NULL): NIF del cliente.
10. **Tabla `usuario`**:
    - Almacena la información de los usuarios del sistema.
    - **Campos**:
        - `usuario` (varchar(45), NOT NULL): Nombre de usuario.
        - `password` (varchar(100), NOT NULL): Contraseña encriptada del usuario.
        - `tipo` (int(11), NOT NULL): Tipo de usuario (administrador, empleado, etc.).
        - `token` (varchar(64), DEFAULT NULL): Token de autenticación.

Índices y Claves Primarias

- Cada tabla tiene su clave primaria (`PRIMARY KEY`) definida para garantizar la unicidad de las filas.

- Se han añadido índices (`KEY`) para optimizar las consultas y mejorar el rendimiento.

Relaciones entre Tablas

- **`detallespedido`**:
  - `idpedido` referencia a `pedido(idpedido)`.
  - `idproducto` referencia a `producto(idproducto)`.
- **`empleado`**:
  - `usuario` referencia a `usuario(usuario)`.
- **`habitacion`**:
  - `usuario` referencia a `usuario(usuario)` con la opción de eliminación en cascada (`ON DELETE SET NULL`).
- **`pedido`**:
  - `nhabitacion` referencia a `habitacion(nhabitacion)`.
- **`reserva`**:
  - `nhabitacion` referencia a `habitacion(nhabitacion)`.
- **`reservacliente`**:
  - `idreserva` referencia a `reserva(idreserva)`.
  - `nifcliente` referencia a `cliente(NIF)`.

4. Servidor

Este apartado detalla el proceso de instalación y configuración del servidor en Linux, incluyendo la instalación de Apache, PHP, phpMyAdmin y MySQL (MariaDB). También incluye la configuración de Apache para aceptar conexiones desde cualquier IP y la estructura del servidor para manejar las solicitudes de la aplicación.

Instalación de Apache, PHP, phpMyAdmin y MySQL

Paso 1: Actualizar los paquetes del sistema

bash

Copiar código

```
sudo apt update
sudo apt upgrade -y
```

Paso 2: Instalar Apache

bash

Copiar código

```
sudo apt install apache2 -y
```

Paso 3: Instalar PHP y extensiones necesarias

bash

Copiar código

```
sudo apt install php libapache2-mod-php php-mysql -y
```

Paso 4: Instalar MySQL (MariaDB)

bash

Copiar código

```
sudo apt install mariadb-server mariadb-client -y
```

Paso 5: Configurar MySQL

bash

Copiar código

```
sudo mysql_secure_installation
```

Sigue las instrucciones en pantalla para configurar la seguridad de MySQL (establecer contraseña root, eliminar usuarios anónimos, deshabilitar el acceso root remoto, etc.).

Paso 6: Instalar phpMyAdmin

bash

Copiar código

```
sudo apt install phpmyadmin -y
```

Durante la instalación, selecciona Apache y configura phpMyAdmin con el servidor de bases de datos. Crea una base de datos y un usuario para phpMyAdmin cuando se te solicite.

Configuración de Apache para Aceptar Conexiones Externas

Paso 1: Editar el archivo de configuración de Apache

bash

Copiar código

```
sudo nano /etc/apache2/ports.conf
```

Paso 2: Cambiar la dirección de escucha de `Listen 80` a `Listen 0.0.0.0:80`

text

Copiar código

```
Listen 0.0.0.0:80
```

Paso 3: Reiniciar Apache para aplicar los cambios

bash

Copiar código

```
sudo systemctl restart apache2
```

Estructura del Servidor

La estructura del servidor se organiza en el directorio `/var/www/html/hotel/`, con los siguientes archivos PHP para manejar diferentes operaciones de la aplicación:

- **Actualización**
  - `actualizar-cliente.php`
  - `actualizar-empleado.php`
- **Eliminación**
  - `borrar-cliente.php`
  - `borrar-empleado.php`
  - `borrarpedido.php`
  - `borrar-reserva.php`
  - `borrar-reservas-antiguas.php`
- **Cambiar Estado**
  - `cambiar-estado-hab.php`
  - `completar-pedido.php`
- **Descargas**
  - `descargar-factura.php`
  - `descargar-foto-hab.php`
- **Información**
  - `info-reserva.php`
  - `obtener-info.php`
  - `obtener-pedido.php`
  - `obtener-pedidos.php`
- **Inserciones**
  - `insertar-cliente.php`
  - `insertar-empleado.php`
  - `insertar-habitacion.php`
  - `insertar-pedido.php`
  - `insertar-reserva.php`
  - `insertar-usuario.php`

- **Autenticación y Sesión**
  - `login.php`
  - `usersession.php`
- **Valoración**
  - `valoracion.php`
- **Archivos Auxiliares**
  - `connection.php` (para conectar con la base de datos)
  - Carpeta `img` (para almacenar imágenes subidas)

Cada archivo PHP se conecta a la base de datos utilizando `connection.php` y valida un token de autenticación (`token_auth`) para mayor seguridad. Además, los archivos PHP devuelven una respuesta en formato JSON, que es utilizada por la aplicación Java para procesar los datos.

Grand Hotel
+34 123 123 123
info@grandhotel.es
www.grandhotel.es