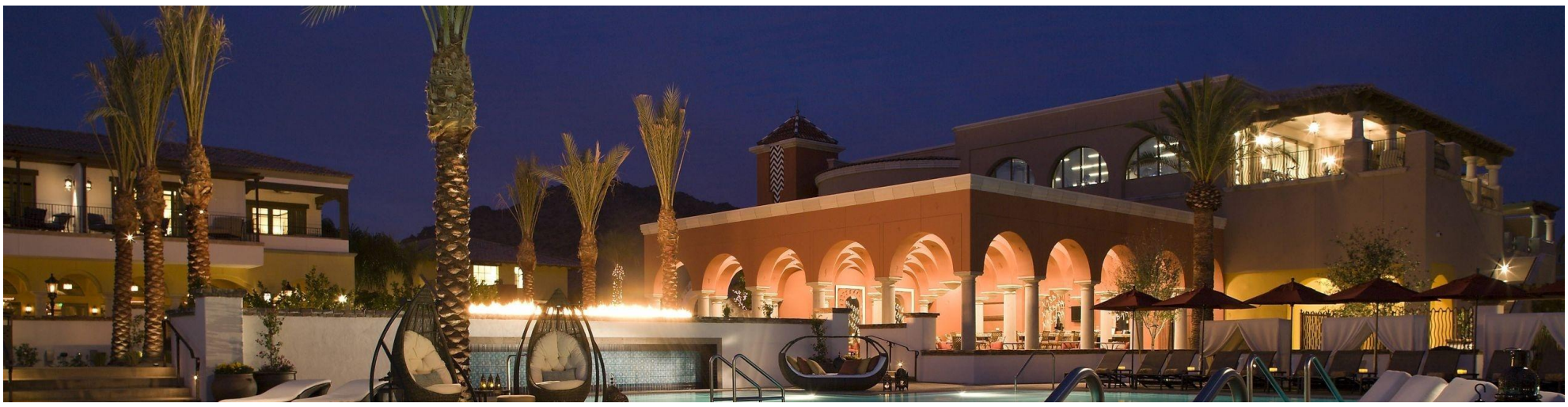


Enunciado.

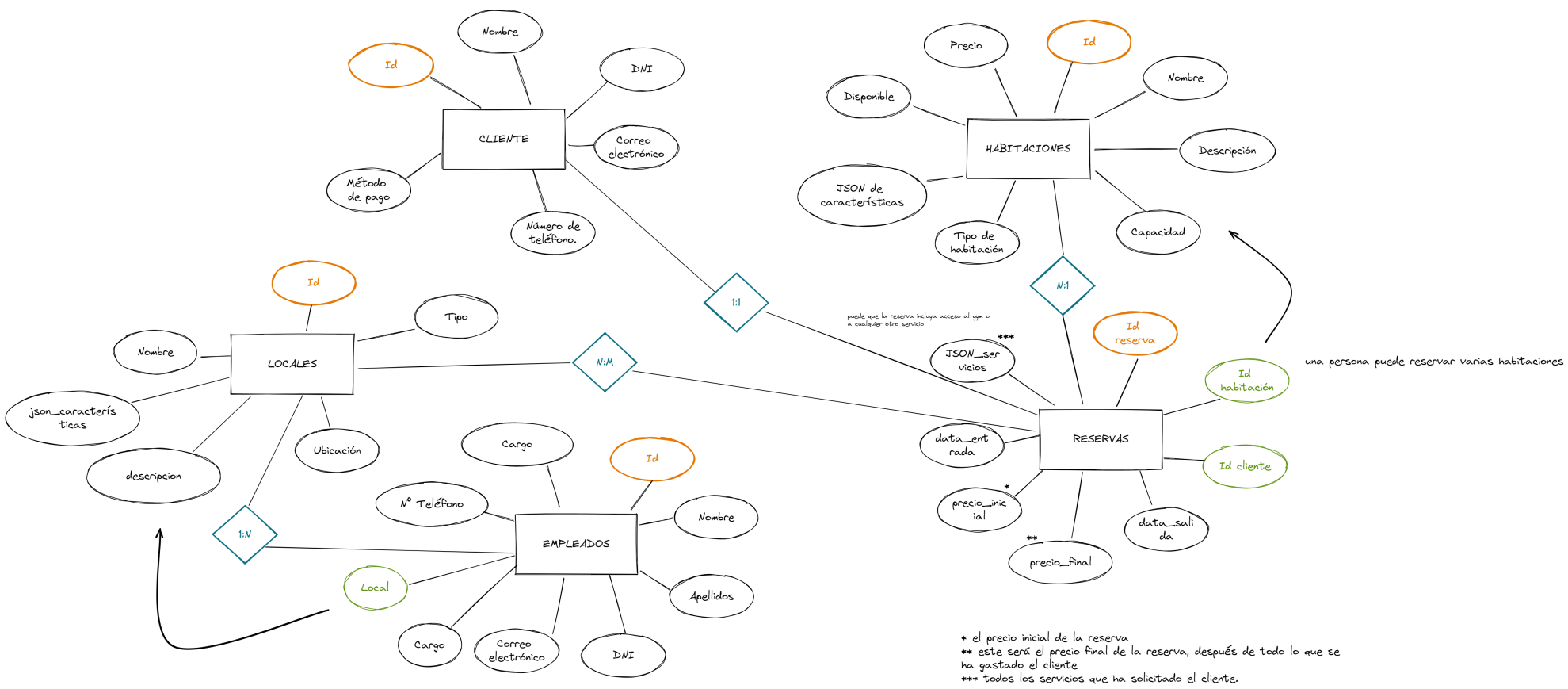
Upload your 3T Database Project about HOTEL MANAGEMENT as a zip file with the following documents:

1. Customer Requirements
2. E/R Diagram and Logical Model
3. Physical implementation: SQL file with the CREATE commands.
4. Online manual
5. Backup of your database in .sql format
6. Any other information related to your project

Name the file as ForenameLastname_HotelManagement.zip

Hotel Project.

Diagrama Entidad-Relación:



Functions and procedures ideas.

1. Función para obtener los `true` values (en este caso características) del archivo JSON de la tabla `habitaciones`.

```
BEGIN

DECLARE len INT DEFAULT 0;
DECLARE json_len INT DEFAULT 0;
DECLARE json JSON;
DECLARE result JSON;
DECLARE j INT DEFAULT 0;

SET result = JSON_ARRAY();
SET json = JSON_ARRAY();
SET len = (SELECT COUNT(*) FROM habitaciones);

SET json = (SELECT JSON_SEARCH(json_caracteristicas, 'all', '1') FROM habitaciones WHERE id = var_id_room);
SET json_len = JSON_LENGTH(json, '$');
        WHILE j < json_len DO
                SELECT JSON_ARRAY_INSERT( result, CONCAT('$[',j,']') ,
```

```
REPLACE(JSON_UNQUOTE(JSON_EXTRACT(json,CONCAT('$[',j,']'))), '$.', '') ) INTO result;
                SET j = j + 1;
        END WHILE;
SELECT result;

END
```

2. Procediimento para insertar reservas aleatoriamente.

```
BEGIN

DECLARE date1 DATE;
DECLARE date2 DATE;
DECLARE room INT DEFAULT 0;
DECLARE precio INT DEFAULT 0;

SELECT FLOOR(1 + (RAND() * 100)) INTO room;

SELECT precio FROM habitaciones WHERE id = room INTO precio;

SET date1 = randomDATE();
SET date2 = randomDATE();

WHILE (DATEDIFF(date2, date1) < 1) DO
        SET date1 = randomDATE();
        SET date2 = randomDATE();
END WHILE;

INSERT INTO reservas
VALUES
(DEFAULT, room, (FLOOR(1 + (RAND() * 500))), date1, date2, precio, (precio + FLOOR(10 + (RAND() * 300))),
trueValuesJSON(room));

END
```

3. Función para obtener una `DATE` aleatoria:

```
RETURN (SELECT CURDATE() - INTERVAL FLOOR(RAND() * 14) DAY);
```

Tables.

Tablas y sus columnas:

Clientes	Habitaciones	Empleados	Servicios	Reservas	Historial
ID	ID	ID	ID	ID	ID
Nombre	Nombre	Nombre	Nombre	ID Cliente	ID Cliente
~~Apellidos~~	Capacidad	Apellidos	Tipo de servicio	ID Habitación	ID Habitación
Métodos de pago.	JSON características	DNI	Ubicación	Data entrada	Data entrada
~~Nacionalidad.~~	Descripción	Número de teléfono		Data salida	Data salida
DNI.	Tipo	Correo electrónico		Estado de la reserva	Precio inicial
Correo electrónico.	Disponible	Cargo / ID local		Precio inicial	Precio final
Número de teléfono.	Precio			Precio final	JSON Servicios
				JSON Servicios	

Posibles servicios para una habitación de hotel.

Servicios que he elegido para mi hotel (estarán en el JSON de la tabla `habitaciones`:

- Wifi.
- Aire acondicionado.
- Cocina.
- Caja fuerte.
- Limpieza diaria.
- Cambio sabanas y toallas.
- Regalo.

Posibles puestos de trabajo de un hotel.

Los puestos de trabajo que pueden existir en un hotel son:

- Recepcionista.
- Botones (llevar maletas, proporcionar información , etc)
- Gobernante / gobernanta.
- Camarero.
- Cocinero.
- Seguridad.
- Socorrista.
- Masajista.

Pueden existir más, pero de momento solo tendremos estos.

Estados de una habitación.

- Available.
- Booked.
- Out of service.

Estados de una reserva.

- Check-in.
- Check-out.
- Out of service.
- In maintenance.

Creación de tablas en la base de datos.

Para resetear el AUTO_INCREMENT:

```
ALTER TABLE reservas AUTO_INCREMENT = 1; -- Próximo valor
```

Actualizar estas tablas.

- En la tabla `empleados`, donde `id_local`, los valores `NULL` significa que trabajan para el Hotel.

```
CREATE TABLE habitaciones (
        id INT PRIMARY KEY AUTO_INCREMENT,
        nombre varchar(255),
        descripcion longtext,
        capacidad int,
        tipo varchar(255),
        json_caracteristicas JSON,
        disponible boolean,
        precio decimal
);

CREATE TABLE cliente (
        id INT PRIMARY KEY AUTO_INCREMENT,
        nombre varchar(255),
        DNI varchar(255),
        email varchar(255),
        telefono varchar(255),
        metodo_pago varchar(255)
);

CREATE TABLE empleados (
        id INT PRIMARY KEY AUTO_INCREMENT,
        nombre varchar(255),
        apellidos varchar(255),
        DNI varchar(255),
        email varchar(255),
        telefono varchar(255),
        json_cargos JSON
);

CREATE TABLE locales (
        id INT PRIMARY KEY AUTO_INCREMENT,
        nombre varchar(255),
```

```
        tipo varchar(255),
        ubicacion varchar(255),
        descripcion longtext,
        json_caracteristicas JSON,
);

CREATE TABLE reservas (
        id_reserva INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
        id_habitacion INT NOT NULL,
        id_cliente INT NOT NULL,
        data_entrada DATE,
        data_salida DATE,
        precio_inicial decimal,
        precio_final decimal,
        json_servicios JSON,
        FOREIGN KEY (id_habitacion) REFERENCES habitaciones(id),
        FOREIGN KEY (id_cliente) REFERENCES clientes(id)
);

CREATE TABLE historic (
        id_reserva INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
        id_habitacion INT NOT NULL,
        id_cliente INT NOT NULL,
        data_entrada DATE,
        data_salida DATE,
        precio_inicial decimal,
        precio_final decimal,
        json_servicios JSON,
        FOREIGN KEY (id_habitacion) REFERENCES habitaciones(id),
        FOREIGN KEY (id_cliente) REFERENCES clientes(id)
);
```

Querys con ChatGPT.

Query para insertar **datos aleatorios** en la tabla `clientes`:

```
INSERT INTO clientes(
    id,
    nombre,
    DNI,
    email,
    telefono,
    metodo_pago
)
SELECT
        ROW_NUMBER() OVER(
ORDER BY
    (
SELECT NULL
)
) AS id,
    CONCAT(
    nombres.nombre,
    ' ',
    apellidos.apellido
) AS nombre,
    CONCAT(
    FLOOR(RAND() * 89999999) + 10000000,
    LEFT(DNI, 1)) AS DNI,
        CONCAT(
        LEFT(nombre, 1),
    REPLACE
        (apellido, ' ', ''),
        '@',
        dominios.dominio
    ) AS email,
        CONCAT(
        '9',
        FLOOR(RAND() * 89999999) + 10000000) AS telefono,
            metodos_pago.metodo AS metodo_pago
    FROM
        (
        SELECT
            'Juan' AS nombre
        UNION
    SELECT
        'María'
    UNION
SELECT
    'Pedro'
UNION
```

```
SELECT
    'Ana'
    ) AS nombres,
    (
    SELECT
        'Gómez' AS apellido
    UNION
SELECT
    'López'
UNION
SELECT
    'García'
UNION
SELECT
    'Martínez'
) AS apellidos,
(
    SELECT
        '12345678A' AS DNI
    UNION
SELECT
    '23456789B'
UNION
SELECT
    '34567890C'
UNION
SELECT
    '45678901D'
) AS DNI,
(
    SELECT
        'gmail.com' AS dominio
    UNION
SELECT
    'yahoo.com'
UNION
SELECT
    'hotmail.com'
UNION
SELECT
    'outlook.com'
) AS dominios,
(
    SELECT
        'Tarjeta' AS metodo
    UNION
SELECT
    'Transferencia'
UNION
SELECT
    'PayPal'
UNION
SELECT
    'Efectivo'
) AS metodos_pago
LIMIT 500;
```

Query para insertar locales aleatorios **que no se repiten** en `locales`:

```
INSERT INTO locales(
    id,
    nombre,
    tipo,
    ubicacion,
    descripcion,
    json_caracteristicas
)
SELECT
    ROW_NUMBER() OVER(
ORDER BY
    (
SELECT NULL
)
) AS id,
CONCAT('Local ', numero) AS nombre,
tipos.tipo AS tipo,
ubicaciones.ubicacion AS ubicacion,
CONCAT('Descripción del local ', numero) AS descripcion,
CONCAT(
    '{"wifi": ',
    caracteristicas.wifi,
    ', "tv": ',
```

```
    caracteristicas.tv,
    ', "aire_acondicionado": ',
    caracteristicas.aire_acondicionado,
    ', "terraza": ',
    caracteristicas.terraza,
    '}'
) AS json_caracteristicas
FROM
    (
    SELECT
        1 +(seq.seq - 1) AS numero
    FROM
        seq_1_to_20 seq
) AS numeros,
(
    SELECT
        'Restaurante' AS tipo
    UNION
SELECT
    'Cafetería'
UNION
SELECT
    'Tienda'
UNION
SELECT
    'Salón de eventos'
) AS tipos,
(
    SELECT
        'Lobby' AS ubicacion
    UNION
SELECT
    'Planta baja'
UNION
SELECT
    'Piso 1'
UNION
SELECT
    'Piso 2'
UNION
SELECT
    'Piso 3'
) AS ubicaciones,
(
    SELECT
        FLOOR(RAND() * 2) AS wifi,
        FLOOR(RAND() * 2) AS tv,
        FLOOR(RAND() * 2) AS aire_acondicionado,
        FLOOR(RAND() * 2) AS terraza) AS caracteristicas
    WHERE NOT
        EXISTS(
        SELECT
            1
        FROM
            locales
        WHERE
            nombre = CONCAT('Local ', numero)
    )
LIMIT 20;
```

Query para actualizar los datos de la tabla `habitaciones`, específicamente en la tabla `json_características`:

```
UPDATE
    habitaciones
SET
    json_caracteristicas = JSON_OBJECT(
        'wifi',
        FLOOR(RAND() * 2),
        'aire_acondicionado',
        FLOOR(RAND() * 2),
        'cocina',
        FLOOR(RAND() * 2),
        'caja_fuerte',
        FLOOR(RAND() * 2),
        'limpieza_diaria',
        FLOOR(RAND() * 2),
        'cambio_sabanas_toallas',
        FLOOR(RAND() * 2),
        'regalo',
        FLOOR(RAND() * 2))
```

```
    WHERE
        id BETWEEN 1 AND 100;
```

JSON

JSON_caracteristicas de la tabla `habitaciones`:

```
{
    "wifi" : false,
    "aire_acondicionado" : false,
    "cocina" : true,
    "caja_fuerte" : false,
    "limpieza_diaria" : true,
    "cambio_sabanas_toallas" : false,
    "regalo" : false
}
```

JSON_caracteristicas de la tabla `locales`:

```
{
    "wifi" : false,
    "aire_acondicionado" : false,
    "interior" : false,
    "terraza" : true,
    "piscina" : false
}
```

Funciones, vistas, eventos, etc.

Procedimiento para seleccionar los valores `true` de una key del `json_caracteristicas` y pasarlos a la tabla `reservations`.

Para esto usaremos un procedimiento:

Primero, buscaremos todas las `keys` que estén a `true` o `1`:

```
SELECT JSON_SEARCH(json_caracteristicas, 'all', '1') AS searchJSON
FROM habitaciones;
```

Para si que si el `SELECT = 0`, pues sacame esto; sino, sacame esto.

```
SELECT
        CASE JSON_VALUE(json_caracteristicas, "$.wifi")
        WHEN 0 THEN "no"
        ELSE "si"
    END
AS "Wi-Fi"
FROM habitaciones;
```

- VIEW para una vista humana de las habitaciones y sus servicios:

```
CREATE VIEW habitaciones_view AS
SELECT
    `hotel`.`habitaciones`.`id` AS `id`,
    `hotel`.`habitaciones`.`nombre` AS `nombre`,
    `hotel`.`habitaciones`.`descripcion` AS `descripcion`,
    `hotel`.`habitaciones`.`capacidad` AS `capacidad`,
    `hotel`.`habitaciones`.`tipo` AS `tipo`,
    CASE `hotel`.`habitaciones`.`disponible` WHEN 0 THEN 'No' ELSE 'Si'
END AS `disponible`,
`hotel`.`habitaciones`.`precio` AS `precio`,
CASE json_value(
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.wifi'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `wifi`,
CASE json_value(
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.aire_acondicionado'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `aire_acondicionado`,
CASE json_value(
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.cocina'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `cocina`,
CASE json_value(
```

```
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.caja_fuerte'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `caja_fuerte`,
CASE json_value(
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.limpieza_diaria'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `limpieza_diaria`,
CASE json_value(
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.cambio_sabanas_toallas'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `cambio_sabanas_y_toallas`,
CASE json_value(
    `hotel`.`habitaciones`.`json_caracteristicas`,
    '$.regalo'
) WHEN 0 THEN 'No' ELSE 'Si'
END AS `regalo`
FROM
    `hotel`.`habitaciones`
```

- Evento para insertar una reserva cada 1 minuto:

```
CREATE EVENT reservationInsertEvery1min
ON SCHEDULE EVERY 1 MINUTE
DO
CALL randomReservations();
```