PROYECTO FINAL DE TRIMESTRE

Para este proyecto he diseñado una base de datos de un hotel compuesta por cuatro colecciones: reservas, habitaciones, clientes y reviews.

Teniendo en cuenta el motivo académico del proyecto, algunos datos aparecen más simplificados que en la realidad. Del mismo modo, algunas consultas resultan menos realista a fin de utilizar la mayor cantidad de operadores nuevos posibles y explorar todas las posibilidades que ofrece MongoDB.

La primera colección "bookings" guarda información de las reservas del hotel. En ella aparecen los siguientes campos:

_id (PK)		Number	Identificador de la reserva
customerID (FK)		String	Identificador del cliente
checkinDate		Date	Fecha de entrada al hotel
checkoutDate		Date	Fecha de salida del hotel
rooms	**class (FK)**	String	Identificador del tipo de habitación
	boardType	String	Tipo de servicio (Pensión completa, media pensión…)
	dailyBoardTax	Number	Precio diario del servicio
coupon		Number	Descuento al precio total
atCharge		String	Empleado al cargo de reserva y de cumplir las necesidades de los clientes

bookings.js > ...

```
db.bookings.drop()
db.bookings.insertMany([
    {
        _id: 1,
        customerID: "00001",
        checkinDate: ISODate("2022-01-14T12:00:00Z"),
        checkoutDate: ISODate("2022-01-19T12:00:00Z"),
        bookingDate: ISODate("2022-01-09T10:50:00Z"),
        rooms: [
            {
                class: "LX2",
                boardType: "Full Board",
                dailyBoardTax: 200
            },
            {
                class: "LX2",
                boardType: "Half Board",
                dailyBoardTax: 100
            }
        ],
        coupon: 0.05,
        atCharge: "Michaela Cross"
    },
```

La segunda colección “rooms” guarda información de los tipos de habitaciones del hotel. En ella aparecen los siguientes campos:

_id (PK)	String	Identificador de la habitación
description	String	Clasificación de habitación.
roomsQty	Number	Número de habitaciones de este tipo en el hotel.
maxCapacity	Number	Número máximo de personas por habitación
dailyPrice	Number	Precio diario de la habitación
tv	Boolean	Indica si la habitación cuenta o no con televisión.

```
JS rooms.js > ...
1   db.rooms.drop()
2   db.rooms.insertMany([
3       {
4           _id: "ST1A",
5           description: "Standart",
6           roomsQty: 20,
7           maxCapacity: 1,
8           dailyPrice: 30,
9           tv: false
10      },
```

La tercera colección “customers” guarda la información de contacto del cliente, así como datos de aquellos otros acompañantes que participen en la reserva. En ella aparecen los siguientes campos:

_id (PK)		String	Identificador del cliente
contactName		String	Nombre del cliente que realiza la reserva
contactPhone		String	Teléfono del cliente que realiza la reserva
nationality		String	Nacionalidad de los clientes
data	**name**	String	Nombre del cliente
	age	Number	Edad del cliente

```
JS customers.js > data
db.customers.drop()
db.customers.insertMany([
    {
        _id: "00001",
        contactName: "Peter Grunwald",
        contactPhone: "+49 0211106040",
        nationality: "German",
        data: [
            {
                name: "Peter Grunwald",
                age: 51
            },
            {
                name: "Christina Kortig",
                age: 49
            },
            {
                name: "Leah Grunwald",
                age: 12
            },
            {
                name: "Paul Grunwald",
                age: 24
            }
        ]
    },
```

La cuarta colección “reviews” guarda la información de las valoraciones realizadas por los clientes al salir del hotel. En ella aparecen los siguientes campos:

_id (PK)	String	Identificador de la review
bookingID (FK)	Number	Identificador asociado a la reserva.
customerID (FK)	String	Identificador asociado al cliente.
satisfaction	Number	Satisfacción con la estancia.
tags	Array	Lista de etiquetas usadas por los clientes para describir su estancia.

```
JS reviews.js > ...
db.reviews.drop()
db.reviews.insertMany([
    {
        _id: "R2022N00001",
        bookingID: 1,
        customerID: "00001",
        satisfaction: 9,
        tags: ["average price", "entertaining", "clean", "excellent restaurant"]
    },
```

Para las consultas se ha hecho uso de los operadores vistos en clase…

✔ $unwind	✔ $multiply	✔ $arrayElemAt
✔ $substract	✔ $addFields	✔ $out
✔ $sort	✔ $avg	✔ $match
✔ $round	✔ $lookup	✔ $expr

Además, también se han usado nuevos operadores que dejaré brevemente explicados:

- $addToSet: devuelve un array con todos los valores únicos para los campos seleccionados entre cada documento del grupo (sin repeticiones).

```
{ $addToSet: <expression> }
```

- $cond: evalúa una expresión booleana para devolver una de las dos expresiones de devolución especificadas.

```
{ $cond: { if: <boolean-expression>, then: <true-case>, else:
<false-case> } }
```

- $dateToString: convierte un dato fecha en un string con un formato especificado por el usuario.

```
{ $dateToString: {
    date: <dateExpression>,
    format: <formatString>,
    timezone: <tzExpression>,
    onNull: <expression>
} }
```

- $size: cuenta y devuelve el número total de elementos en un array.

```
{ $size: <expression> }
```

- $switch: evalúa una serie de expresiones de casos. Cuando encuentra una expresión que se evalúa como verdadera, $switch ejecuta una expresión específica y sale del flujo de control.

```
$switch: {
     branches: [
          { case: <expression>, then: <expression> },
          { case: <expression>, then: <expression> },
          ...
     ],
     default: <expression>
}
```

- $limit: limita el número de documentos pasados a la siguiente etapa en la canalización.

```
{ $limit: <positive 64-bit integer> }
```

- $range: devuelve una matriz cuyos elementos son una secuencia de números generada a partir del número inicial especificado incrementando sucesivamente el número inicial por el valor de paso especificado hasta el punto final sin incluirlo.

```
{ $range: [ <start>, <end>, <non-zero step> ] }
```

- $add: suma números a un número o a una fecha.

```
{ $add: [ <expression1>, <expression2>, ... ] }
```

- $map: aplica una expresión a cada elemento de un array y devuelve un array con los resultados aplicados.

```
{ $map: { input: <expression>, as: <string>, in: <expression> } }
```