



**ESCUELA SUPERIOR POLITÉCNICA DEL
LITORAL**

SISTEMA DE BASE DE DATOS AVANZADO

II Término, 2021

INTEGRANTES

- Gary Barzola
- Tommy Villagómez

PROYECTO DEL PRIMER PARCIAL

GOOGLE CLOUD SPANNER

DDL:

```
CREATE TABLE booking (  
    bookingid INT64 NOT NULL,  
    bookdate DATE,  
    flightid INT64 NOT NULL,  
    seatid INT64 NOT NULL  
) PRIMARY KEY(bookingid);  
  
CREATE TABLE bookingdetails (  
    bookingid INT64 NOT NULL,  
    passid INT64 NOT NULL,  
) PRIMARY KEY(bookingid, passid);  
  
ALTER TABLE bookingdetails ADD FOREIGN KEY(bookingid) REFERENCES  
booking(bookingid);  
  
CREATE TABLE flight (  
    flightid INT64 NOT NULL,  
    flightsource STRING(1024),  
    flightdest STRING(1024),  
    flightdate DATE  
) PRIMARY KEY(flightid);  
  
ALTER TABLE booking ADD FOREIGN KEY(flightid) REFERENCES  
flight(flightid);  
  
CREATE TABLE passenger (  
    passid INT64 NOT NULL,  
    passname STRING(1024),  
    passemail STRING(1024),  
    passdob DATE,  
) PRIMARY KEY(passid);  
  
ALTER TABLE bookingdetails ADD FOREIGN KEY(passid) REFERENCES  
passenger(passid);  
  
CREATE TABLE seat (  
    seatid INT64 NOT NULL,  
    seatnumber INT64 NOT NULL,  
    seatcost FLOAT64 NOT NULL,  
    flightid INT64 NOT NULL  
) PRIMARY KEY(seatid);
```

```
ALTER TABLE seat ADD FOREIGN KEY(flightid) REFERENCES flight(flightid);
ALTER TABLE booking ADD FOREIGN KEY(seatid) REFERENCES seat(seatid);
```

REDISEÑO DE LA BASE DE DATOS

Primero antes de hacer el rediseño de la base de datos se realizó la lectura con las recomendaciones para mejorar operaciones de lectura y escrito, con lo cual decidimos poner el id de todas las tablas como un timestamp aleatorio, y al momento de insertarlo en la base se verificó que no existiera para posteriormente insertarlo de manera desordenada, lo cual esto nos permitió que las operaciones de inserción no se acumulen en un solo nodo(esto al no ser un id autoincremental), sino que se distribuyan en los 3 nodos que teníamos.

Para no sobrecargar la tabla de Flight se procedió a crear una tabla extra llamada Seat, la cual contiene el precio, el número de asiento de un vuelo y una clave foránea que lo relaciona con el vuelo respectivo, esto se hizo con la finalidad de no saturar las operaciones de escritura y lectura dentro de la tabla Flight.

DATOS SINTÉTICOS USADOS PARA POBLAR NUESTRA BDD CON AL MENOS 1000 REGISTROS EN CADA TABLA:

```
function randomFlight(maximo)
{
    const {Spanner} = require('@google-cloud/spanner');
    var data = [];
    for (var i = 0; i < maximo; i++ )
    {
        data.push({
            flightid: String(i+1),
            flightsource: String(randomSource()),
            flightdest: String(randomDest()),
            flightdate: String(randomDate()),
        });
    }
    return data;
}

function randomPassenger(maximo)
{
    var data = [];
    for (var i = 0; i < maximo; i++ )
    {
        data.push({
            passid: String(i+1),
            passname: String(randomName()),
        });
    }
    return data;
}
```

```

        passemail: String(randomEmail()),
        passdob: String(randomDob()),
    });
}
return data;
}

function randomBooking(maximo)
{
    var data = [];
    for (var i = 0; i < maximo; i++ )
    {
        data.push({
            bookingid: String(i+1),
            bookdate: String(randomDate()),
            flightid: String(randomNumber(1,1000)),
            seatid: String(randomNumber(1,1000)),
        });
    }
    return data;
}

function randomBookingDetails(maximo){
    var data = [];
    for (var i = 0; i < maximo; i++ )
    {
        data.push({
            bookingid: String(i+1),
            passid: String(randomNumber(1,1000)),
        });
    }
    return data;
}

function randomSeat(maximo){
    const {Spanner} = require('@google-cloud/spanner');

    var data = [];
    for (var i = 0; i < maximo; i++ )
    {
        data.push({
            seatid: String(i+1),
            seatnumber: String(randomNumber(1,10)),
        });
    }
}

```

```

        seatcost:
Spanner.float(parseFloat(randomFloat(600,1000).toFixed(2))),
        flightid: String(randomNumber(1,1000)),
    });
}
return data;
}

```

Mismas funciones que sirven para insertar registros en nuestras tablas y que reposan dentro del archivo crud.js para luego ser llamadas dentro de una función principal:

```

async function insertData(instanceId, databaseId, projectId) {
    // [START spanner_insert_data]
    // Imports the Google Cloud client library
    const {Spanner} = require('@google-cloud/spanner');
    // Creates a client
    const spanner = new Spanner({
        projectId: projectId,
    });

    // Gets a reference to a Cloud Spanner instance and database
    const instance = spanner.instance(instanceId);
    const database = instance.database(databaseId);

    // Instantiate Spanner table objects
    const flightsTable = database.table('flight');
    const bookingsTable = database.table('booking');
    const passengersTable = database.table('passenger');
    const bookingdetailsTable = database.table('bookingdetails');
    const seatsTable = database.table('seat');

    // Inserts rows into the Singers table
    // Note: Cloud Spanner interprets Node.js numbers as FLOAT64s, so
    // they must be converted to strings before being inserted as INT64s
    let dataFlights = randomFlight(1000);
    let dataPassengers = randomPassenger(1000);
    let dataBookings = randomBooking(1000);
    let dataBookingDetails = randomBookingDetails(1000);
    let dataSeats = randomSeat(1000);
    try {
        await flightsTable.insert(dataFlights);
        await passengersTable.insert(dataPassengers);
        await seatsTable.insert(dataSeats);
    }
}

```

```

    await bookingsTable.insert(dataBookings);
    await bookingdetailsTable.insert(dataBookingDetails);

    console.log('Inserted data.');
```

```

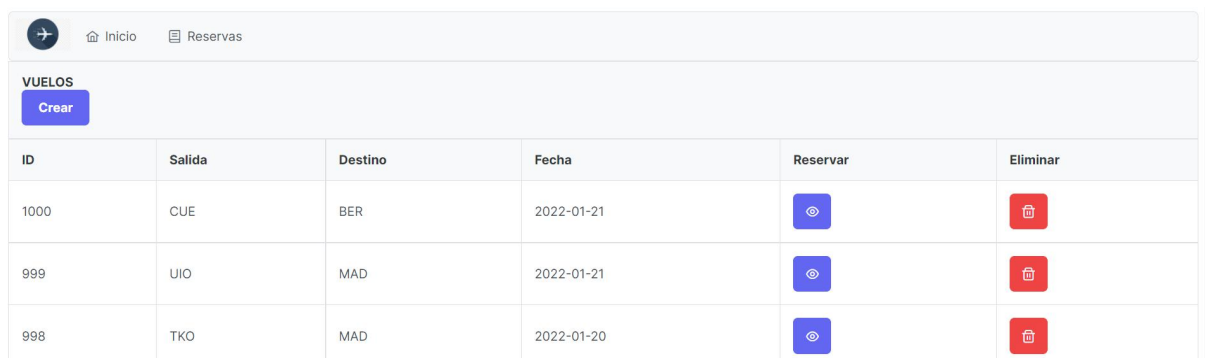
  } catch (err) {
    console.error('ERROR:', err);
  } finally {
    await database.close();
  }
}
// [END spanner_insert_data]
}

```

Y completar el proceso de poblar nuestra BDD.

VISTAS DE LA APLICACIÓN WEB.

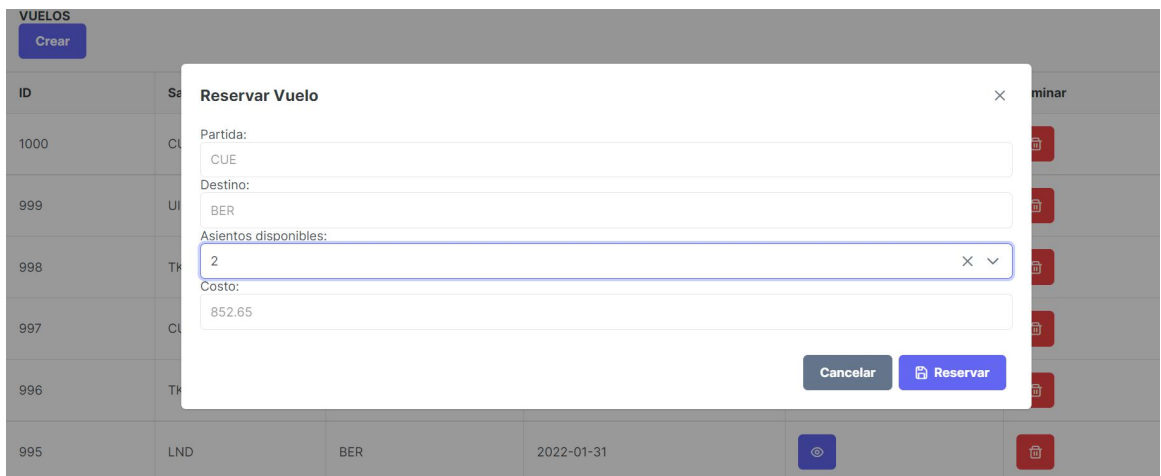
1. Pantalla donde se visualizan los vuelos:



ID	Salida	Destino	Fecha	Reservar	Eliminar
1000	CUE	BER	2022-01-21		
999	UIO	MAD	2022-01-21		
998	TKO	MAD	2022-01-20		

Misma donde se pueden eliminar los vuelos creados mientras no tengan una reserva disponible en curso.

2. Pantalla donde se realiza una reserva de vuelo:



Reservar Vuelo

Partida: CUE

Destino: BER

Asientos disponibles: 2

Costo: 852.65

Cancelar Reservar

Misma que permite escoger uno de los asientos disponibles y en paralelo se actualiza el costo del asiento escogido.

3. Pantalla donde se visualizan las reservas realizadas:

RESERVAS							
ID	Salida	Destino	Fecha	Asiento	Costo	Actualizar	Eliminar
999	ROM	MAD	2022-01-26	6	750.78		
998	BOG	PRV	2022-01-23	6	868.69		
997	CUE	TRV	2022-01-21	1	696.11		

Misma que permite eliminar las reservas que ya no deseemos.

4. Pantalla donde se crean los vuelos:

VUELOS

Crear

ID	Salida	Destino	Fecha	Asiento	Costo	Actualizar	Eliminar
1000	CUE	BOG	2022-01-23	6	868.69		
999	UIO	PRV	2022-01-23	6	868.69		
998	TKO	BAR	2022-01-25	6	750.78		
997	CUE	TRV	2022-01-21	1	696.11		
996	TKO	BAR	2022-01-25	6	750.78		
995	LND	BER	2022-01-31	6	750.78		

Crear Vuelo

Partida:

GUAYAQUIL

Destino:

QUITO

Fecha:

02/02/2022

Cancelar

Crear

Misma donde se puede escribir el lugar de Partida, Destino y la fecha de salida.

5. Pantalla donde se pueden actualizar las reservas realizadas:

RESERVAS

ID	Salida	Destino	Fecha	Asiento	Costo	Actualizar	Eliminar
999	ROM	MAD	2022-01-26	6	750.78		
998	BOG	PRV	2022-01-23	6	868.69		
997	CUE	TRV	2022-01-21	1	696.11		
996	CUE	TRV	2022-01-21	1	696.11		
995	ROM	MAD	2022-01-26	6	750.78		
994	UIO	BER	2022-01-23	5	977.96		
993	UIO	PRV	2022-01-23	9	714.88		

Actualizar Reserva

Partida:

ROM

Destino:

MAD

Asientos disponibles:

6

Costo:

750.78

Cancelar

Actualizar

Misma donde solo puede cambiar el asiento y a su vez se visualiza el costo de este.