**PROYECTO 1: GOOGLE CLOUD SPANNER**

**SISTEMAS DE BASES DE DATOS AVANZADOS**

**GRUPO 8**

**INTEGRANTES:**

MANUEL LOORMACÍAS

LUIS SÁNCHEZ PLAZA

**PROFESORA:**

MSc. VANESSA ECHEVERRÍA BARZOLA

**REDISEÑO DE LA BASE DE DATOS.**

Para el rediseño de nuestra base de datos aplicamos las recomendaciones sobre esquema y modelado de datos de la documentación de Google Cloud Spanner. Primeramente, se estableció que las claves primarias de cada tabla deben ser compuestas, y para no afectar considerablemente el rendimiento de lectura/escritura fue necesario evitar ubicar una columna incremental como el inicio de la clave primaria. Por ejemplo, en la tabla flight (vuelo) la clave primaria está compuesta por la columna de flightid y flighdate, donde la columna flightid contiene datos que siguen la secuencia, sin repetidos, de 1 a 1000, por tal motivo, se ubicó la columna no incremental flightdate como inicio de la clave compuesta. Además, para este caso, se aprovechó el tipo de dato DATE de la columna flightdate para ordenar las filas en orden descendente (DESC), de acuerdo a las recomendaciones de la documentación oficial.

Adicionalmente, se consideró la creación de la tabla seat (asiento) dado que la base de datos original contenía el costo del vuelo dentro y la tabla flight, pero aplicando el proceso real, el costo de la reserva para viajar en avión por lo general varía de acuerdo al asiento reservado, por tanto, se decidió separar esa columna de la tabla flight, y se ubicó en la tabla seat para que esté relacionado con la tabla booking. La nueva tabla seat contiene las columnas de seatid, seatnumber, seatcost, la clave foránea flightid y la clave primaria está conformada por flightid y seatid, ya que flightid como es una clave foránea no era un valor incremental dentro de la tabla seat.

Resumiendo lo explicado anteriormente, las claves primarias para cada tabla de la base de datos, se asignaron de la siguiente manera:

* Tabla flight: PRIMARY KEY (flightdate DESC, flightid)
* Tabla seat: PRIMARY KEY (flightid, seatid)
* Tabla booking: PRIMARY KEY (bookdate DESC, seatid, flightid, bookingid)
* Tabla passenger: PRIMARY KEY (passdob DESC, flightid)
* Tabla bookingdetails: PRIMARY KEY (bookingid, flightid)

**CÓDIGO DDL DE LA BASE DE DATOS.**

`CREATE TABLE flight (

        flightid     INT64 NOT NULL,

        flightsource  STRING(20) NOT NULL,

        flightdest   STRING(20) NOT NULL,

        flightdate   DATE NOT NULL

      ) PRIMARY KEY (flightdate DESC, flightid)`,

      `CREATE TABLE seat (

        seatid     INT64 NOT NULL,

        flightid   INT64,

        seatnumber   INT64 NOT NULL,

        seatcost   FLOAT64 NOT NULL,

        CONSTRAINT FK\_Seat\_Flight FOREIGN KEY (flightid) REFERENCES flight (flightid)

      ) PRIMARY KEY (seatnumber, seatid)`,

      `CREATE TABLE booking (

        bookingid    INT64 NOT NULL,

        seatid       INT64 NOT NULL,

        flightid     INT64 NOT NULL,

        bookdate   DATE NOT NULL,

        CONSTRAINT FK\_Booking\_Flight FOREIGN KEY (flightid) REFERENCES flight (flightid),

        CONSTRAINT FK\_Booking\_Seat FOREIGN KEY (seatid) REFERENCES seat (seatid),

      ) PRIMARY KEY (bookdate DESC, seatid, flightid, bookingid)`,

      `CREATE TABLE passenger (

        passid     INT64 NOT NULL,

        passname   STRING(50) NOT NULL,

        passemail   STRING(50),

        passdob DATE NOT NULL

      ) PRIMARY KEY (passdob DESC, passid)`,

      `CREATE TABLE bookingdetails (

        bookingid     INT64 NOT NULL,

        passid     INT64 NOT NULL,

        CONSTRAINT FK\_Bookingdetails\_Booking FOREIGN KEY (bookingid) REFERENCES booking (bookingid),

        CONSTRAINT FK\_Bookingdetails\_Passenger FOREIGN KEY (passid) REFERENCES passenger (passid)

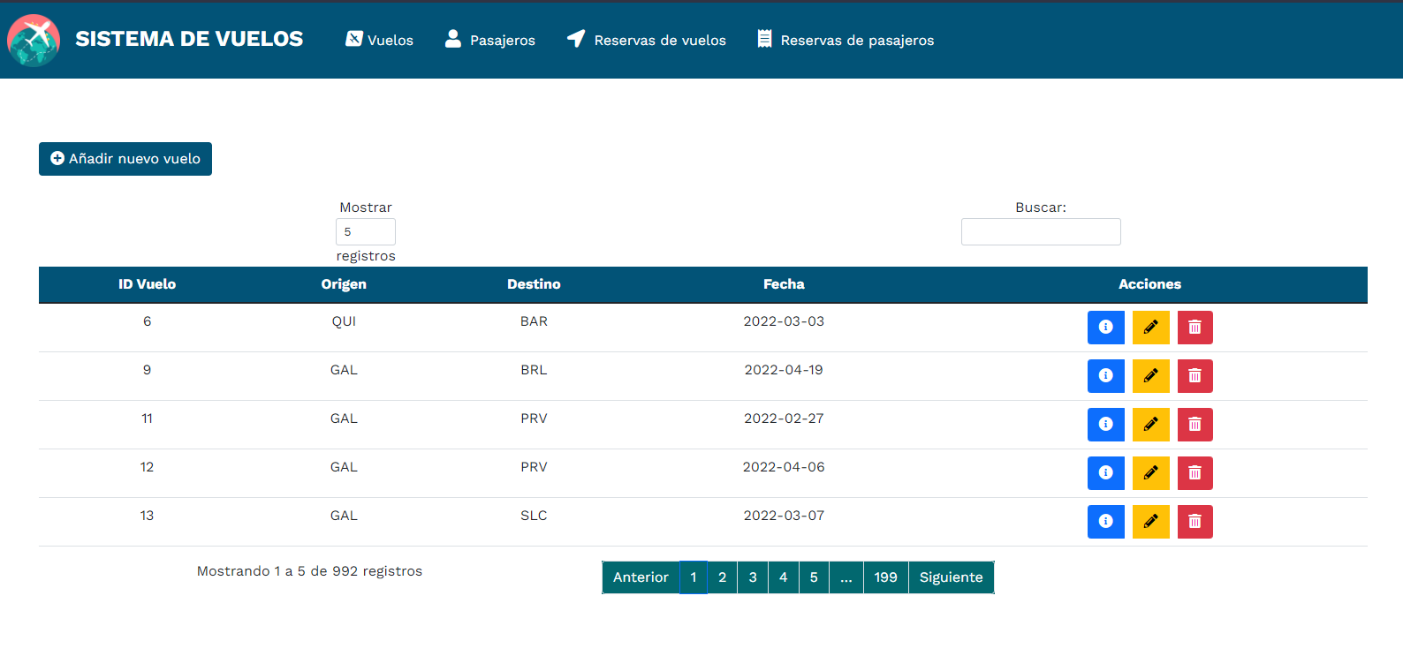
      ) PRIMARY KEY (passid, bookingid)`

Donde se puede observar que entre la tabla flight y seat existe una relación 1: M, entre la tabla flight y booking existe una relación 1: M, así mismo entre la tabla seat y booking existe una relación de 1: M. Por otra parte, entre la tabla booking y Passenger existe una relación M: M, que es soportada por la tabla bookingdetails.

**FUNCIONAMIENTO DEL SISTEMA**

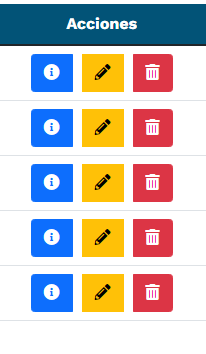
**Primeramente, para la creación del Front-end de la aplicación se utilizó el framework Vuejs. Por otra parte, para la creación del Back-end de la aplicación se utilizó Expressjs.**

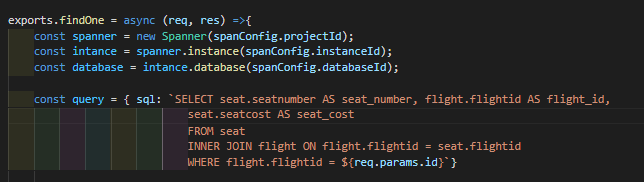
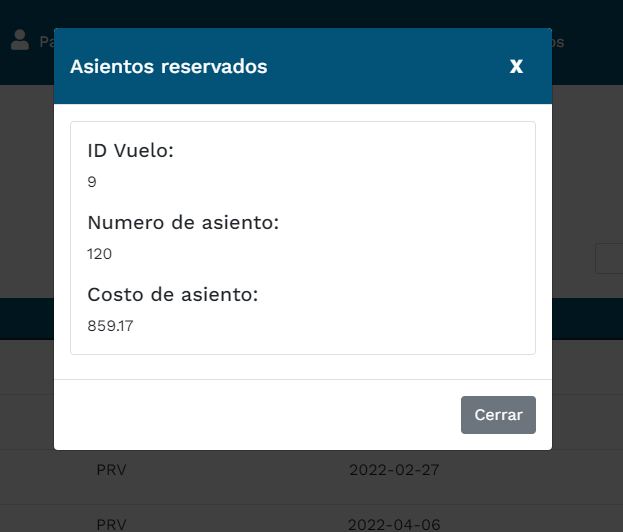
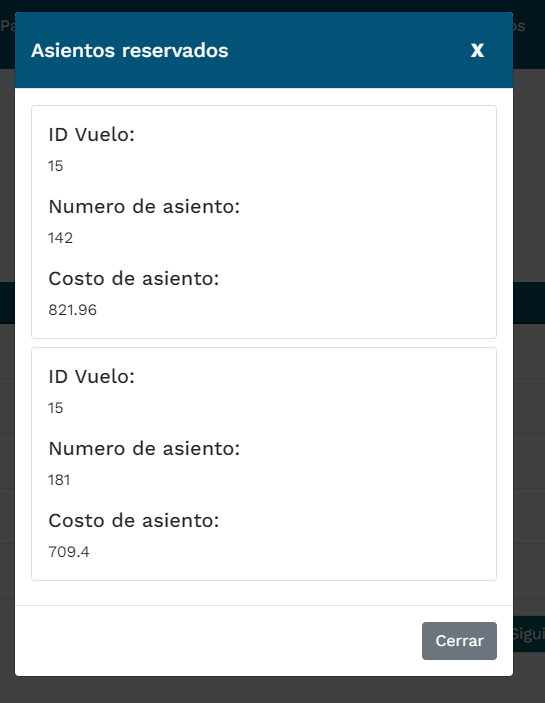
**Vista de vuelos**

Como vista inicial tenemos los registros de vuelos, los cuales son almacenados dentro de una tabla, generada con una librería llamada DataTables para frameworks como Vue y React.

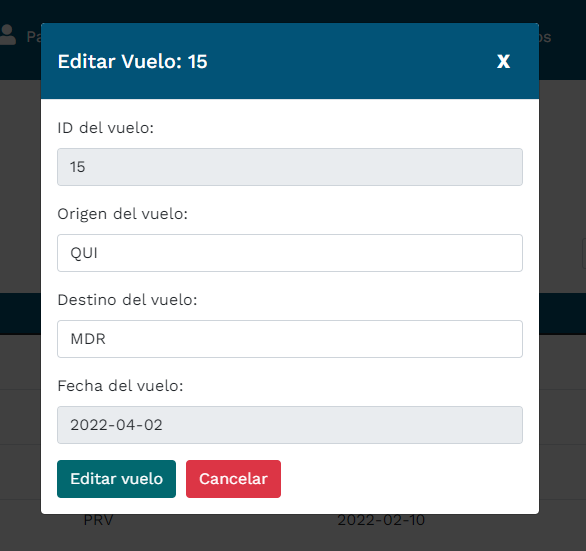
* **Opción más detalles.**

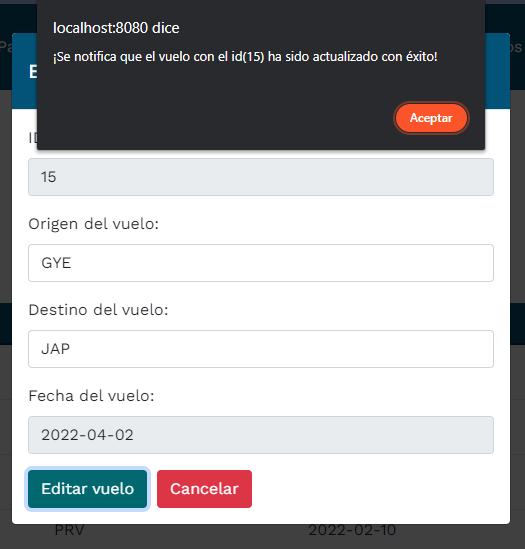
Todas las vistas contienen acciones los cuales pueden visualizarse en cada registro de la tabla, suelen contener tres opciones que son **más detalles, editar y eliminar**. Para este caso, la opción de más detalles presenta la información relacionada entre ese vuelo y sus asientos reservados, esto es posible debido a las sentencias SELECT establecidas en la parte del Back-end de la aplicación web.



La sentencia SELECT anteriormente mencionada, presenta los números de los asientos, el id del vuelo, y el costo de vuelo por asiento reservado, a través de un INNER JOIN realizado entre la tabla flight y seat.

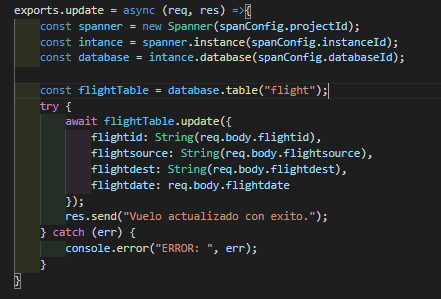
* **Opción editar vuelo**

Por otra parte, al desear editar un registro es necesario dar clic en el ícono de lápiz, el cual presentará un formulario modal que recuperará los datos del vuelo seleccionado. El formulario de editar tiene la configuración de deshabilitar la edición del id del vuelo y la fecha de vuelo, dado que conforman la clave primaria.

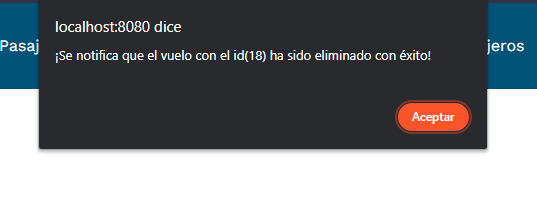
Para el ejemplo, se cambia el origen y destino del vuelo, posteriormente, clic en “Editar vuelo” y nos presentará una alerta de confirmación.

Al dar clic en la opción de “aceptar” se cerrará la alerta y el formulario, y se presentar los cambios realizados.



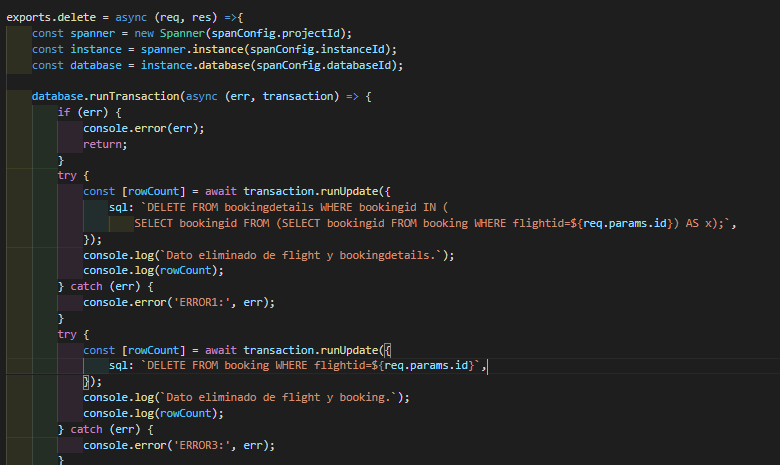
Para editar el vuelo seleccionado en la base de datos, en la parte del Back-end se estableció el siguiente procedimiento con la librería Spanner.

* **Opción eliminar vuelo**

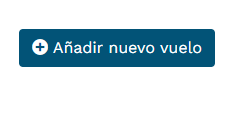
Para eliminar un vuelo en específico es necesario dar clic en el ícono de papelera. Para el ejemplo se eliminará el vuelo con id 18. Una vez realizado ese proceso se presentará una alerta de confirmación.

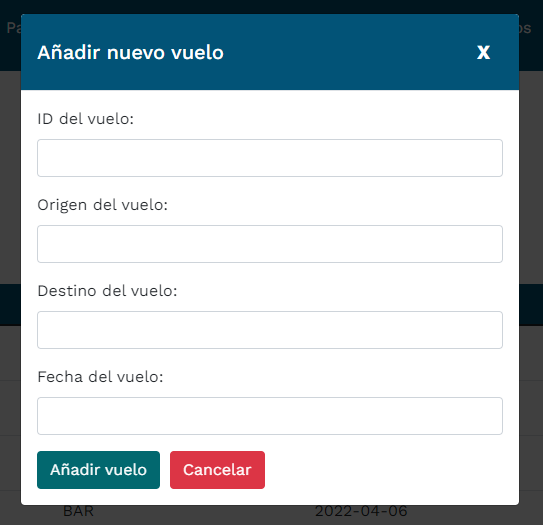
Al dar clic en la opción “aceptar” se cerrará la alerta y serán visibles los cambios en la tabla.

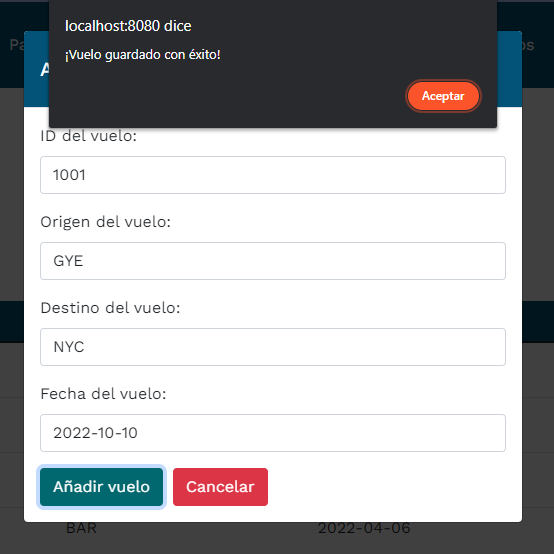


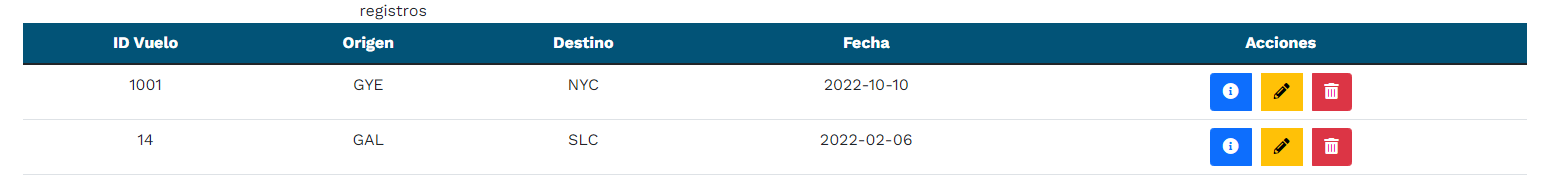
Para eliminar el vuelo seleccionado en la base de datos, en la parte del Back-end se estableció el siguiente procedimiento con la librería Spanner. A diferencia de los controladores realizados para las otras tablas, en el controlador de vuelos fue necesario implementar el uso de subquerys como se puede observar en la figura, dado a que si se elimina un vuelo es necesario eliminar el/los asientos relacionados, la reserva y el detalle de reserva, sin embargo, la tabla de detalle de reserva no tiene relación con la tabla vuelo, por tanto, fue necesario eliminar primero el detalle de vuelo, después la reserva, luego el/los asientos y finalmente el vuelo, con la condición de que el flightid sea el especificado.

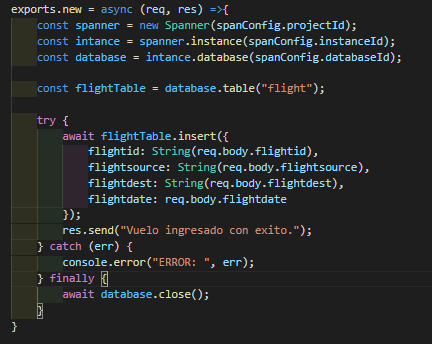
* **Opción añadir vuelo**

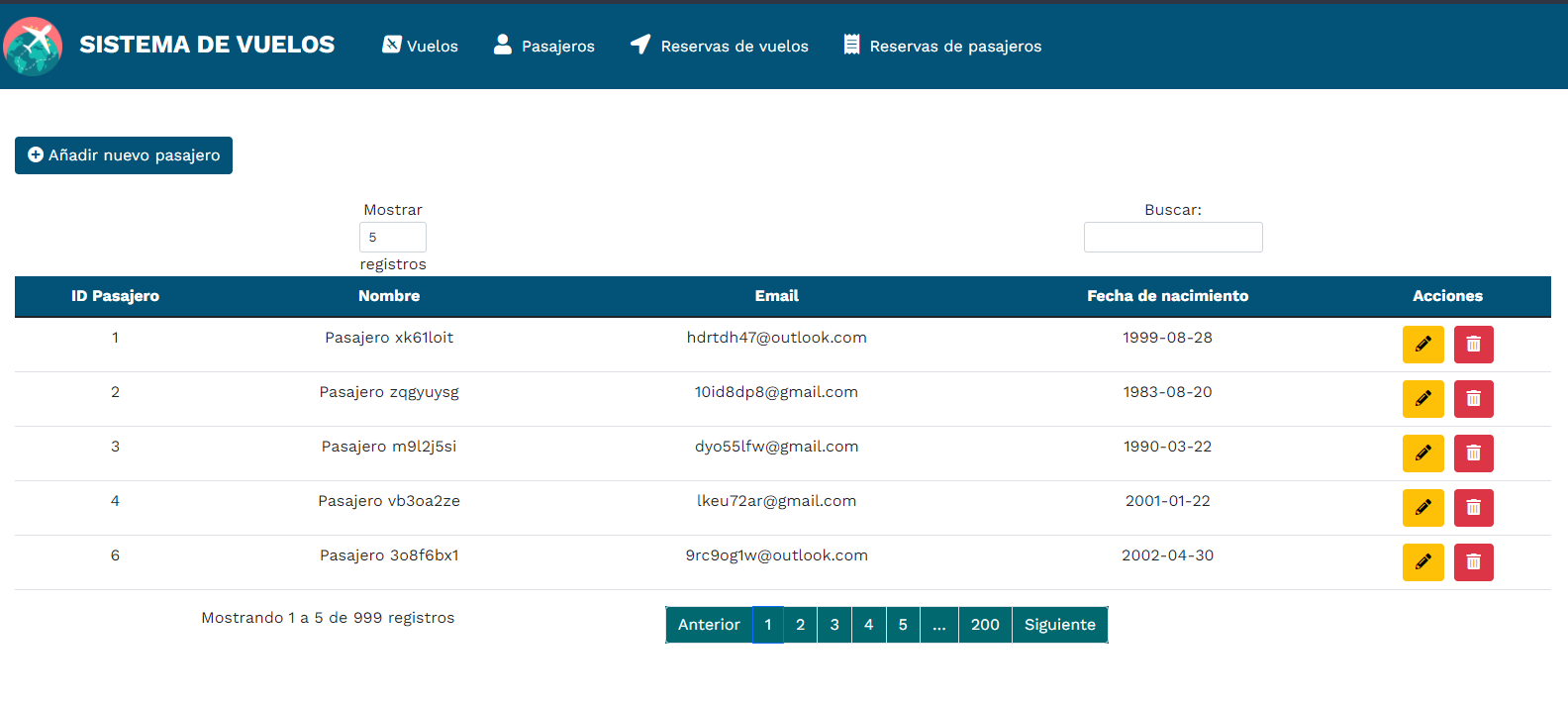
Adicionalmente, se tiene la opción de agregar un nuevo vuelo, con la opción de “añadir nuevo vuelo”.

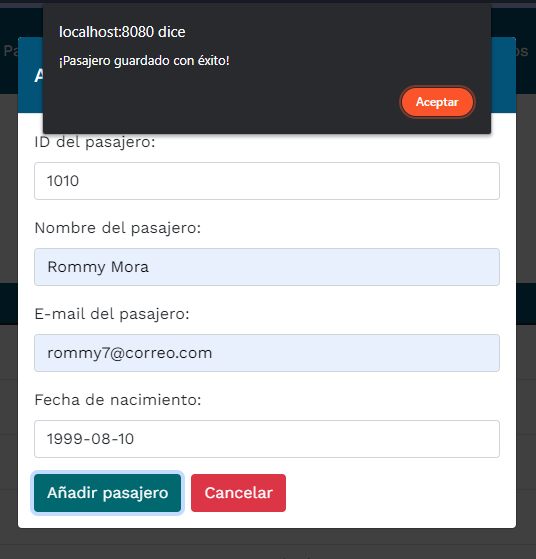
La opción despliega el siguiente formulario de registro.

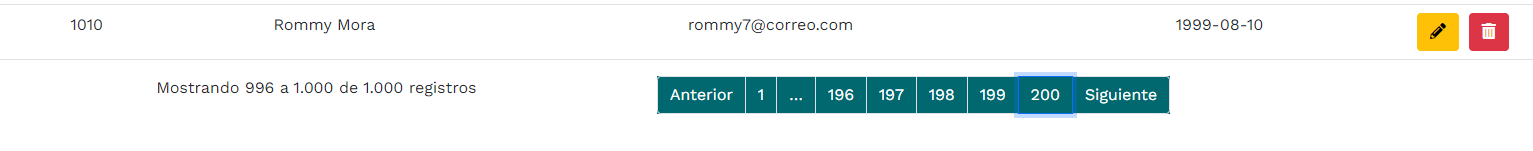
Completando los datos del formulario y posteriormente al dar clic en el botón “añadir vuelo”, se presentará una alerta de confirmación.

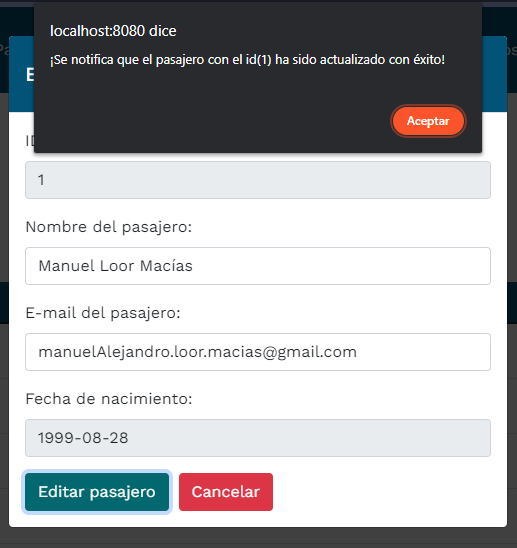
Al dar clic en la opción de “aceptar” de la alerta del navegador, se limpiará el formulario, y aparecerá el nuevo vuelo ingresado en la tabla.

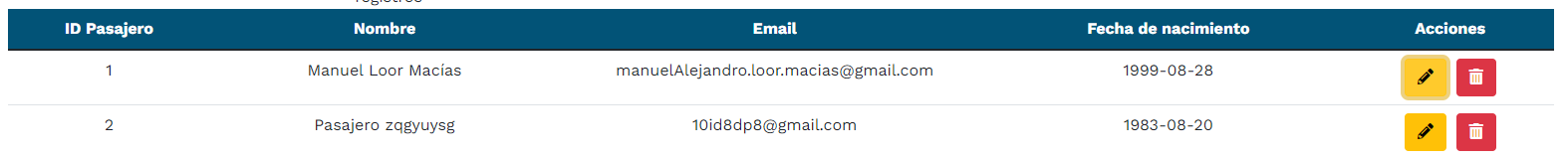
Para añadir un nuevo vuelo en la base de datos, en la parte del Back-end se estableció el siguiente procedimiento con la librería Spanner.

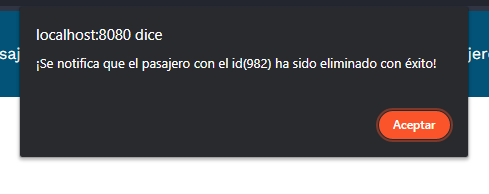
**Vista de pasajeros**

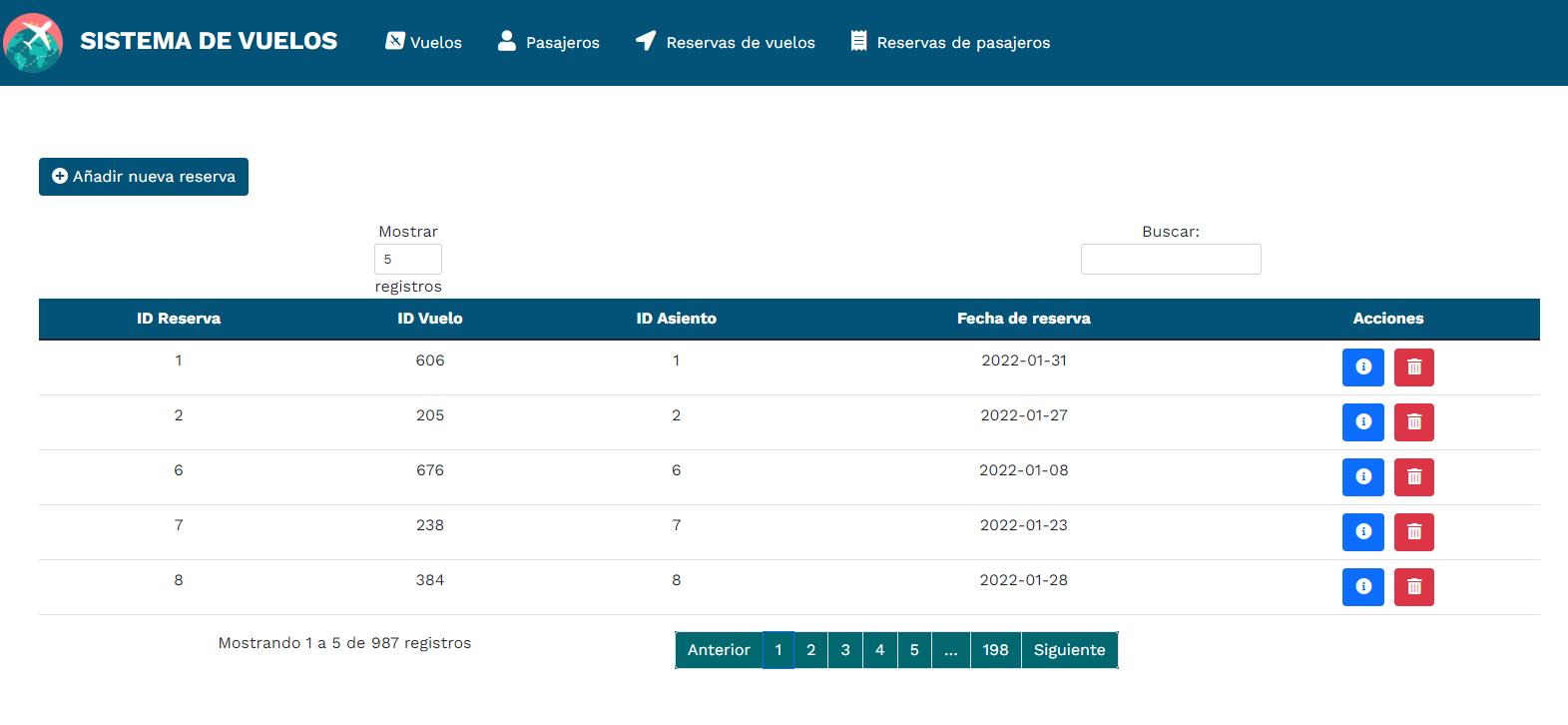
* **Opción agregar pasajero**



* **Opción editar pasajero**

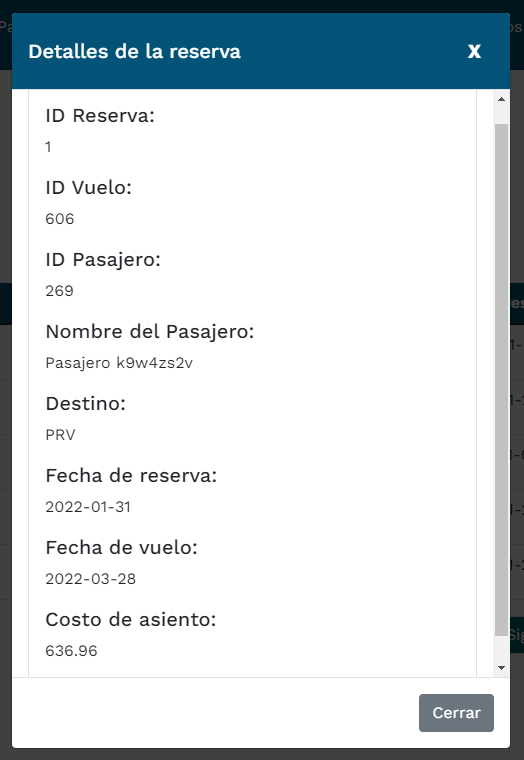


* **Opción eliminar pasajero**

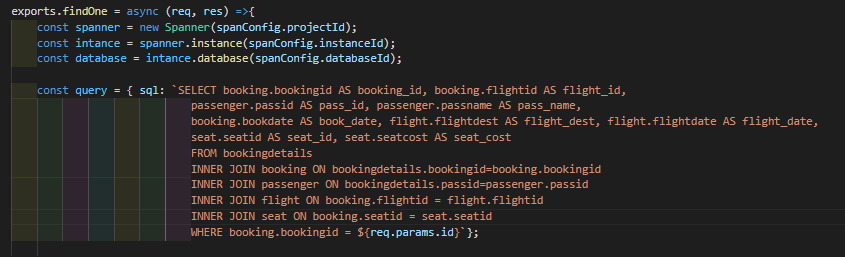
**Vista de reservas**

* **Opción más de detalles de la reserva**

La vista de reservas fue configurada de tal forma que presente información específica de todas las tablas de la base de datos, es decir, de forma adicional se presentarán datos de las demás tablas deacuerdo su relación con la reserva seleccionada. Por lo tanto, se presentará en formulario los datos de id de reserva, id de vuelo, id de pasajero, nombre del pasajero, destino, fecha de reserva y el costo de asiento.

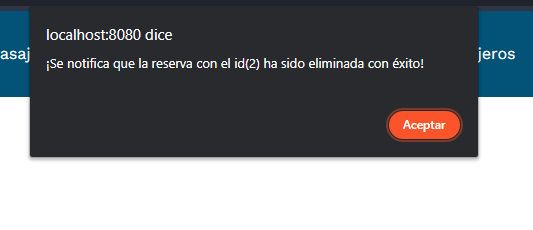


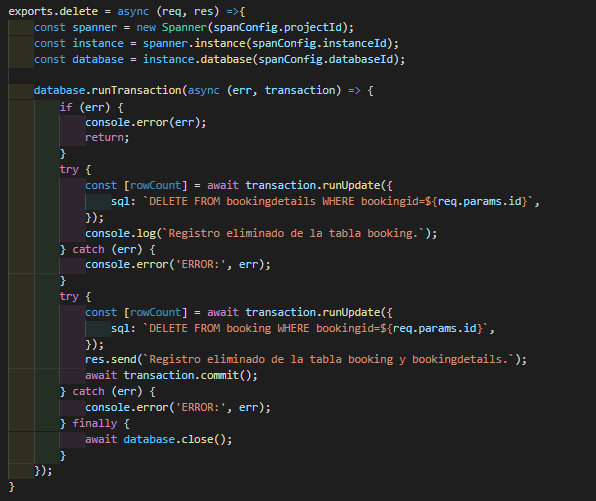
Esto es realizado a través del código escrito en la parte del Back-end, el cual utiliza una sentencia SELECT que relaciona a todas las tablas utilizando INNER JOIN.

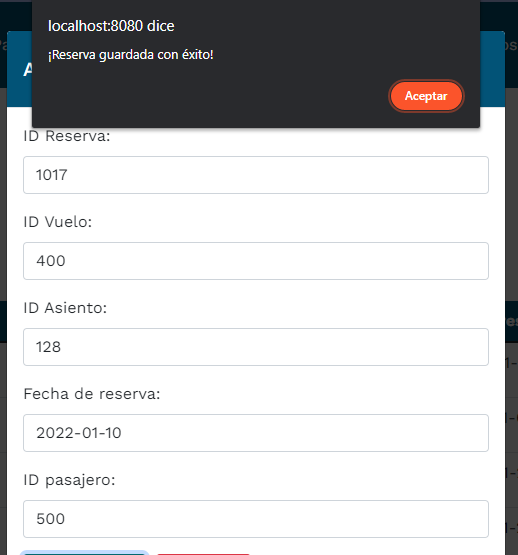


* **Opción editar reserva**

No fue habilitada dado que todas las columnas de la tabla reserva conforman la clave primaria, y su edición podría ocasionar conflictos con el servidor local y la base de datos.

* **Opción eliminar reserva**

Antes de eliminar la reserva es necesario eliminar primeramente su relación con la tabla de detalles de reserva, de acuerdo al id de reserva. Esto es posible por el siguiente código escrito en la parte del backend.

* **Opción añadir nueva reserva**



**Conclusiones:**

* Cloud Spanner es una opción viable al considerar un servidor de datos en la nube, el costo por lectura/escritura y el manteniemiento de la instancia en Google Cloud puede ser un inconveniente, pero sigue siendo una excelente opción.
* Vue es una framework capaz de soportar gran cantidad de datos, al igual que sus competidores Angular y React.

**Recomendaciones:**

* Si es un proyecto personal, procurar eliminar la instancia en Google Cloud, caso contrario se descontarán rápidamente los créditos que se tenga.
* Aprender a utilizar múltiples herramentas de los serviicos de Google Cloud ya que puedenc ser de mucha utilidad para un profesional de informática.