TEC | Tecnológico de Costa Rica

Documentación Técnica

Proyecto I - Tec Air

Curso: Base de datos

Estefanny Villalta Segura

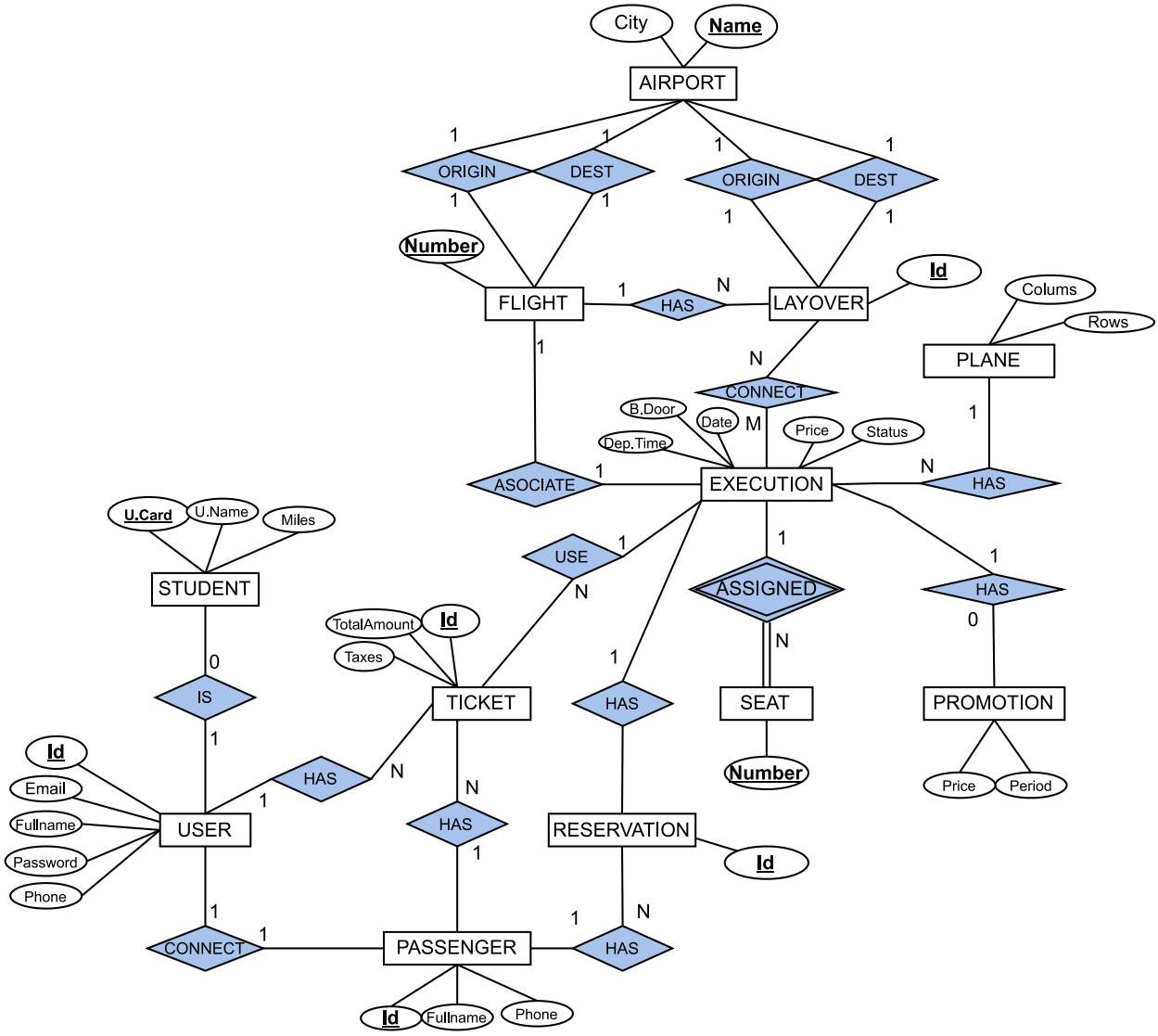
Sebastián Hidalgo Vargas

Valeria Morales Alvarado

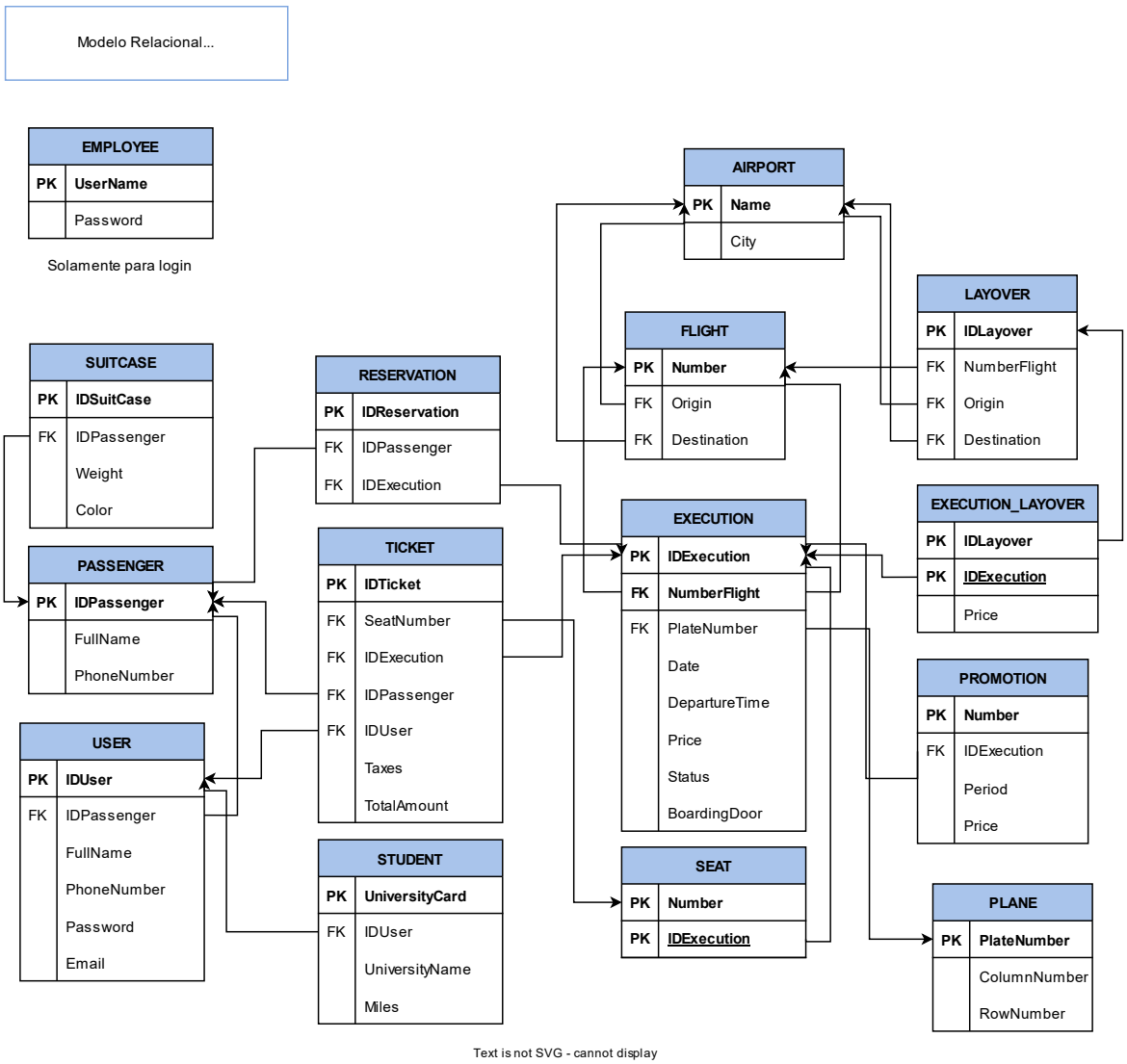
Juan Daniel Rodríguez Montero

Meibel Ceciliano Picado

Modelo Conceptual con Notación de Chen

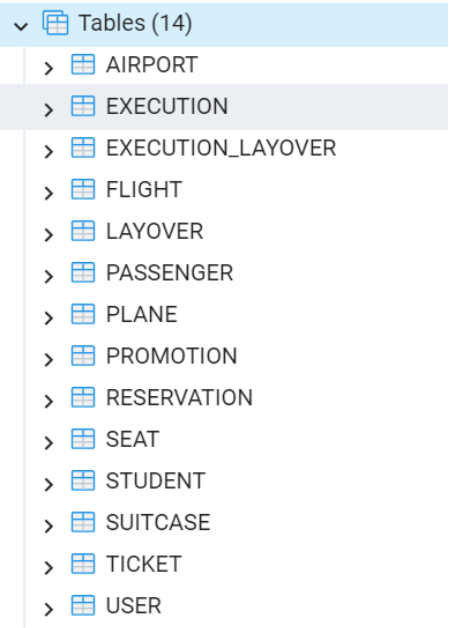


Modelo Relacional



Descripción de las estructuras de datos desarrolladas

Las estructuras de datos utilizadas en este primer proyecto fueron principalmente tablas y listas. La base datos fue desarrollada en PostgreSQL donde fue almacenada la información de cada entidad. Los atributos fueron colocados como columnas en su respectiva tabla indicando minuciosamente cuál de ellos eran primary key o foreign key. A manera de tuplas se iban generando las instancias en el script de población. Para el desarrollo de este proyecto se trabajan las siguientes tablas de entidades:



Airport: esta tabla contiene la información de cada aeropuerto que maneja la aerolínea AirTEC, se almacena para cada uno su “Name” el cual será su primary key y la ciudad donde se encuentra.

Flight: se refiere a los vuelos que se generan, los cuales poseen un numero identificador (PK), y dos atributos principales, “Origin” y “Destination” que hacen referencia a los lugares de donde parte el vuelo hasta donde aterriza.

Execution: se refiere a los viajes que pueden comprar los usuarios o pasajeros, contiene un ID que se genera de forma automática y será su llave primaria, además que se relaciona con plane y flight mediante FK, contiene como atributos fecha y hora ambos de tipo Date, precio, estado del viaje ya sea abierto o cerrado y puerta de abordaje.

Execution_Layover: Esta relación 1:N permite vincular las escalas que puede contener un viaje, se compone de una clave primaria compuesta, el PK de Layover y Execution, además del precio.

Layover: Cada escala maneja su propio identificador, generado de manera automática, contiene tres columnas de llaves foráneas con flight, origen y destino.

Passenger: esta tabla contiene la información de cada pasajero que viaja en la aerolínea, se almacena para cada uno su ID el cual será su PK, por ende, no se permite un valor nulo. Se requiere su nombre completo, y número telefónico el cual será un entero de 8 dígitos.

User: si un pasajero así lo desea puede crearse una cuenta en nuestra plataforma, para ello se requiere un correo electrónico y una contraseña, además de su nombre completo, y número telefónico. Una vez su cuenta es generada se le asigna un numero de ID que tiene restricción de integridad.

Student: en caso de que un usuario sea estudiante puede ingresar su información para almacenar millas en su programa, se necesita el numero de su carnet siendo este un string, el nombre de su universidad y el ID de el usuario, el cual se permite ser nulo pues es una relación de 1:0, claramente no todo usuario que utilice la plataforma tiene esta profesión.

Suitcase: aquí se acumulan las maletas que carga un pasajero, cada una posee un identificador propio y se vincula a su respectivo dueño mediante una llave foránea que hace referencia al ID del pasajero, además contiene como atributos su color y peso, siendo el primero un string y el segundo un entero.

Plane: esta tabla engloba los aviones de AirTEC cada uno con su número de placa, y su capacidad como un atributo que se deriva del número de filas y columnas, ambos de tipo entero.

Seat: se refiere a los asientos que posee un avión, cada asiento contiene un numero que se genera de manera automática al crear una reservación de un viaje, esta relación de 1:N incluye en su tabla la PK de Execution.

Promotion: nuestra aerolínea es capaz de generar promociones en caso de que un vuelo no sea muy ofertado, cada una con su respectivo número funcionando como clave, además de un periodo que

indica su tiempo de vigencia, y evidentemente su precio. Esta relación es de 1:0 junto a Execution pues puede suceder que un viaje no requiera de promoción.

Reservation: Una vez el usuario seleccione su viaje e ingrese sus credenciales se genera el ID de una reservación, la cual permite apartar un asiento en el vuelo respectivo, en la vista administrador deberá llevarse a cabo el check in de esta reservación, para lo cual fueron colocadas como llaves foráneas el identificador de pasajero y ejecución.

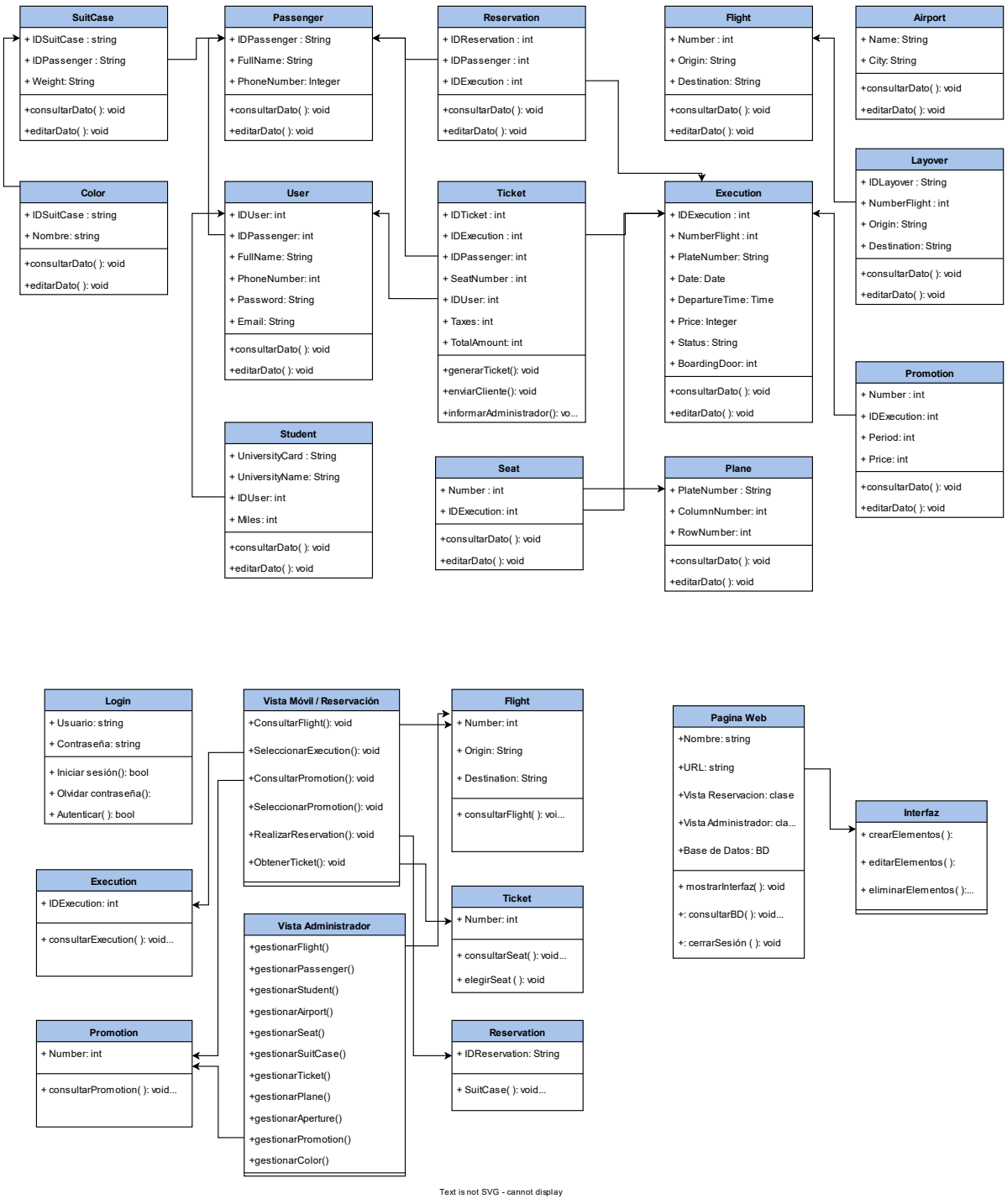
Ticket: finalmente, una vez realizado el check in se obtienen los datos que irán en el tiquete, este reporte será entregado al pasajero para su abordaje, por lo que los datos allí contenidos son muy importantes, entre ellos se encuentran numero de asiento, IDExecution, IDPassenger, IDUser que podría ser nulo, pues no es necesario "loguearse" para comprar un vuelo, impuestos y el precio total. Cabe mencionar que este atributo de precio total ya incluye el precio adicional de cada maleta extra.

Diagrama de Clases:

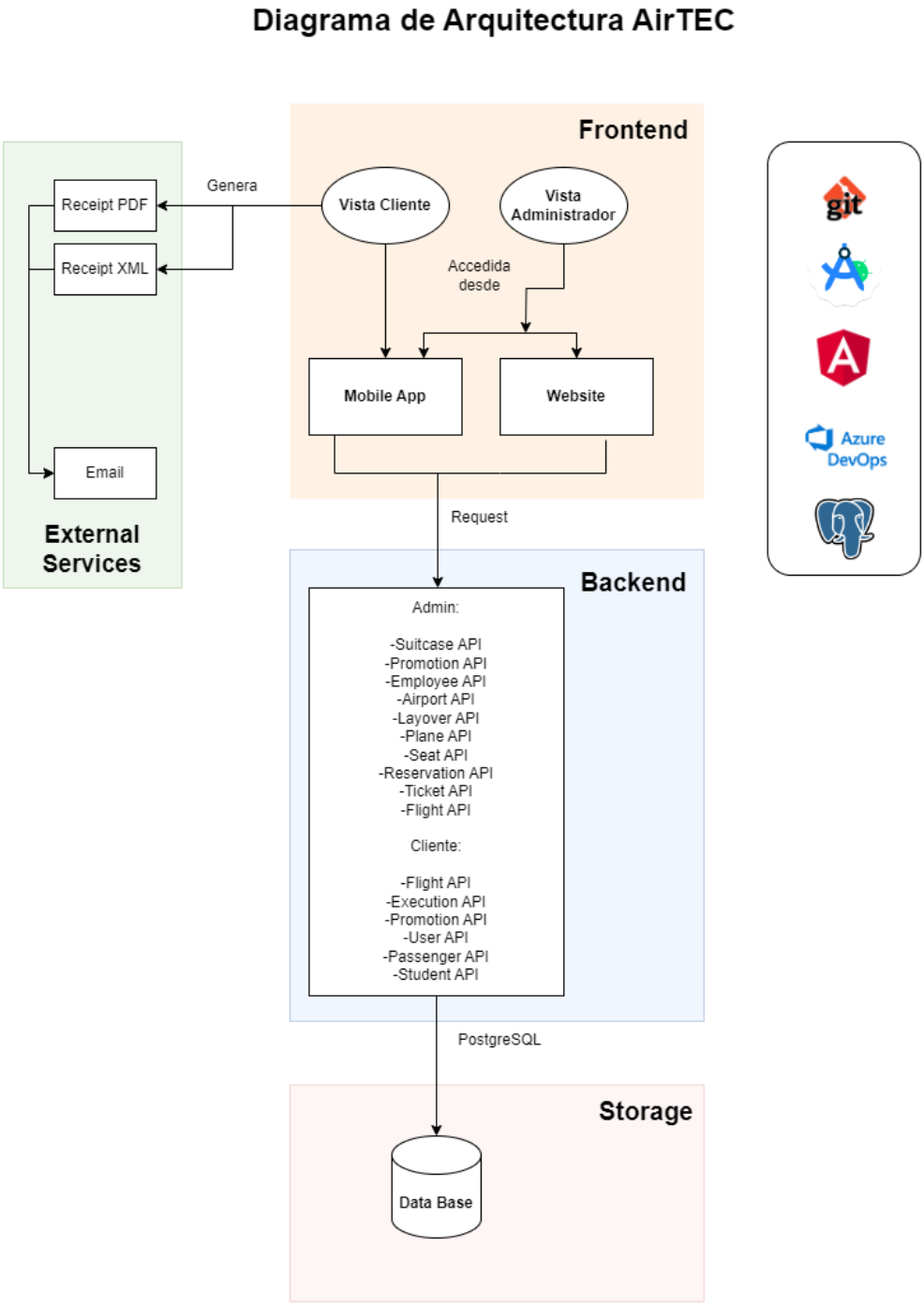
Descripción:

Para el diseño del diagrama, se consideró que estuviera presente todos los elementos esenciales del sistema que se está modelando, se utilizan nombres descriptivos y etiquetas claras para garantizar que el diagrama es fácil de entender y que las relaciones sean identificables de inmediato. De igual manera, se buscó organizar el código en unidades lo más independientes posibles, para facilitar su mantenimiento y modificación a futuro. Al trabajar con bases de datos su enfoque de programación es muy similar a POO, por lo que se colocan las entidades como clases, y los atributos con su respectivo tipo de dato, esto facilita al momento de programar ya que se evitan posibles ambigüedades y malentendidos.

Diagrama de Clases AirTEC



Descripción detallada de la arquitectura desarrollada.



Problemas Encontrados

En este apartado se describen los problemas que se presentaron durante la realización del proyecto y pudieron ser solucionados en el tiempo establecido:

- El principal problema se presentó con el diseño del modelo conceptual y posteriormente el modelo relacional, ya que, las relaciones que se habían considerado eran incorrectas y dificultaban considerablemente la programación del Backend, hubo que generar en reiteradas ocasiones la base de datos, pues las modificaciones eran significativas. Esto causo bastantes retrasos, sin embargo, con la asesoría del profesor fue posible producir un modelo que se ajustara adecuadamente a las especificaciones del proyecto.
- Se enfrentaron algunos inconvenientes al poner a prueba los endpoints en el Swagger, pues se hicieron llamadas a claves primarias o atributos inexistentes, para solucionarlo se apego a la restricción de integridad referencial, de esta manera se aseguró que al hacer una referencia a una tabla los datos eran validos y no había información incompleta o inconsistente.
- Respecto a la página web se encontraron diversos problemas referentes a la obtención, edición y envío de datos. Esto se veía reflejado en errores en el inspector web y la consola para ello se hizo uso de Swagger para poder identificar como interactuar correctamente con la base de datos, así como realizar las conexiones desde los archivos de typescript.
- También se encontraron algunos problemas a la hora de obtener y agregar elementos a la base de datos, tal que los servicios del API funcionaban de manera asincrónica, por lo que se requirió trabajar en la manera en la cual se almacenaban los datos de manera temporal. Se logró solucionar haciendo la llamada de los servicios de forma “anidada” tal que dentro del “.subscribe” de una llamada se agregaba la llamada del siguiente servicio. Además de esto se almacenaron los valores en arrays y variables para poder realizar las funcionalidades de la página web que requerían la información de la base de datos.

- También se tuvieron varios problemas respecto a las ligaduras de los componentes de la interfaz gráfica en el desarrollo móvil, debidas a la falta de experiencia y la curva inicial de aprendizaje pero fue solucionado de manera que todos los componentes gráficos son constantes a pesar de las dimensiones del dispositivo móvil.

Problemas Conocidos

De los problemas emergentes, se tuvo la dificultad de sincronizar la base de datos local de la aplicación móvil con la base de datos global. A la hora de realizar un POST o un GET desde la aplicación, no se actualizaban las tablas de SQLite, dejando solo la información local y no tomaba en cuenta actualizaciones desde el servidor. El proceso de sincronización requería de más pruebas para poder ser confiable y tener completa certeza de que no causaría un problema para el resto de servicios que dependían de la base de datos global.

Conclusiones

Se probo la utilidad del entity framework como mapeo objeto-relacional, EF fue capaz de generar automáticamente clases de modelos basándose en la estructura de base de datos creada en PostgreSQL. Mediante esta herramienta fue posible posteriormente crear controladores y DTO (objetos de transferencia de datos) en el contexto de la API, lo que ayudo a ahorrar tiempo y reducir la posibilidad de errores.

Los diagramas y modelos fueron herramientas clave al momento de programar la base de datos y el API, a través de ellos se visualizaron las tablas, sus columnas, llaves primarias y referencias a otras tablas mediante llaves foráneas. Inclusive al especificar los tipos de datos en el UML, verificando que fueran consistentes con la longitud e información que se deseaba almacenar. Dichos documentos ayudan a detectar posibles inconsistencias de diseño antes de iniciar con la implementación.

Bootstrap es una herramienta que mejoro el uso de Angular ya que proporciona una amplia colección de componentes, estilos predefinidos y un sistema que facilita la creación de interfaces de usuario atractivas. Esto acelera el desarrollo y garantiza un diseño coherente en toda la aplicación.

Recomendaciones

Se recomienda el uso de un servidor en la nube con Azure Database en PostgreSQL en lugar de una infraestructura local, ya que así las personas encargadas de la base de datos pueden tener acceso a la misma versión de los datos de manera inmediata. Además, que permite realizar copias de seguridad en caso de surgir algún accidente y requerir una recuperación.

Usar autogeneración de id’s específicos le puede resultar útil, en el caso de este proyecto se utilizó para los endpoints de vuelo, numero de asiento, factura, reservación, y usuario. La ventaja es que

al ser autogenerados suelen ser únicos en toda la base de datos, lo que garantiza que cada registro tenga una identificación única. Evitando conflictos de duplicación de claves primarias.

Se aconseja prestar especial atención cuando se diseñan los diagramas y modelos, como conceptual, relación y de clases, pues, aunque es un trabajo engorroso ahorrara a futuro aparición de problemas y retrocesos en el código.

Se sugiere hacer un boceto de lo que se desea tener en la página web y la información que se quiere desplegar, de esta manera es más sencillo conocer el flujo de las vistas web y diseñar la interfaz, a la vez que se programa la lógica del backend con los endpoint necesarios.

Bibliografía

Angular. (s. f.). Recuperado de https://angular.io/docs Code Skills. (2018, 4 abril). *Servicios Web - Web API REST* - Parte 1 [Vídeo]. YouTube.

Bootstrap templates and more on WrapBootstrap. (s. f.). WrapBootstrap. https://wrapbootstrap.com/

Code Skills. (2018b, abril 11). *Servicios Web - consumo API REST - parte 3* [Vídeo]. YouTube. https://www.youtube.com/watch?v=HtBX2j2befA

TecnoBinaria. (2017, 14 abril*). Cómo CREAR una BASE de DATOS* | PostgreSQL #2 [Vídeo]. YouTube. https://www.youtube.com/watch?v=GruJjmfm_gs

The Coder Cave esp. (2020, 28 junio). *Aprende Entity Framework | Entity Framework Core para principiantes* | Cuando y cómo usarlo [Vídeo]. YouTube. https://www.youtube.com/watch?v=jpshj-LiRig

Chapter 6. Data Manipulation. (2023, 14 septiembre). PostgreSQL Documentation. https://www.postgresql.org/docs/current/dml.html

Bootstrap. Getting started. Bootstrap. https://getbootstrap.com/docs/5.3/getting-started/introduction/