



Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Curso: CE-3101 Bases de Datos

Proyecto I - TECAir

Profesor: Marco Rivera Meneses

Estudiantes:

Gabriel Vargas López

Víctor Castrillo Muñoz

Yendry Badilla Gutiérrez

Mauricio Calderón Chavarría

Grupo: 1

Fecha: 24 de Abril del 2022

Índice

Modelo relacional y descripción detallada del mapeo	3
Descripción	3
Descripción de las estructuras de datos desarrolladas (Relaciones)	7
Descripción detallada de la arquitectura desarrollada	8
Problemas conocidos	8
Error en responsive	8
Error selección de asientos	8
Error al comprar promoción	8
Error al crear BottomNavigationView en el menú de la app	9
Problemas encontrados	9
Problema conexión API – Base de datos	9
Problema realizando el POST	9
Problema al ejecutar las solicitudes	10
Problema creación de base de datos SQLite	11
Conclusiones	12
Recomendaciones	12
Bibliografía	13

Modelo conceptual

El modelo conceptual se adjunta al final del documento y en PDF en el repositorio, para mayor calidad de imagen.

Modelo relacional y descripción detallada del mapeo

Descripción

Como primer paso se identificaron las entidades fuertes y se realizó el mapeo de estas escribiendo los atributos simples y compuestos en las tablas, identificando con subrayado las primary keys de cada una de las entidades.

Las tablas de usuario, vuelo, promoción, pasajero y tiquete poseían atributos compuestos, los cuales al momento del mapeo se dividieron en atributos simples:

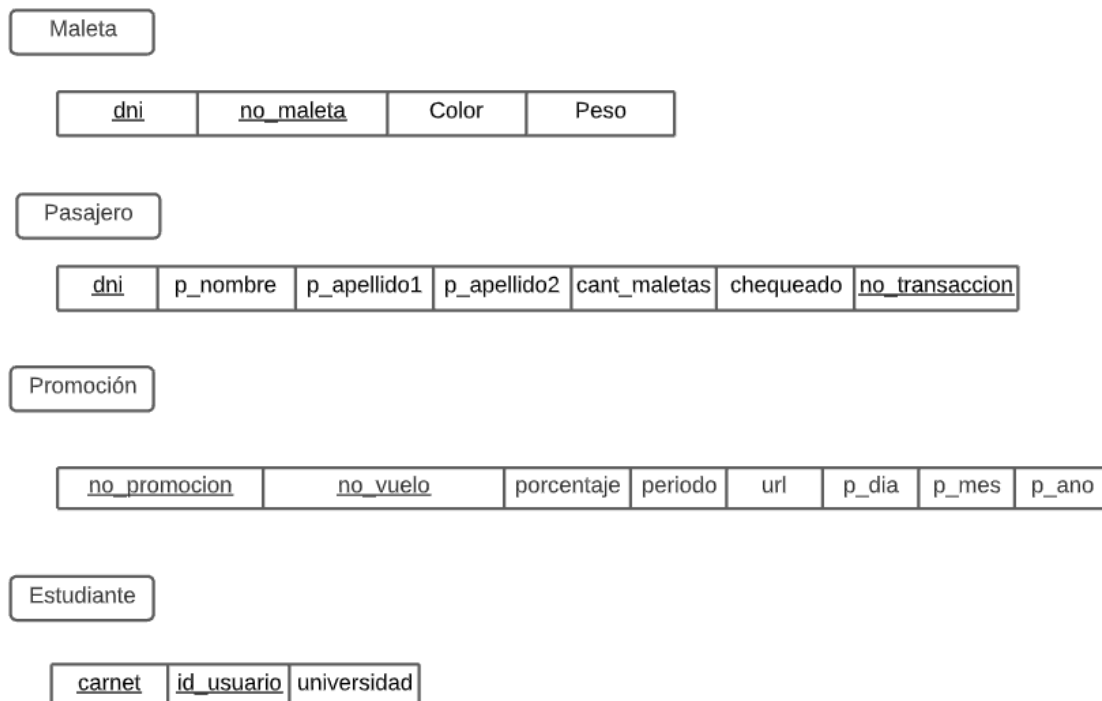
- En la tabla usuario, se dividió el nombre completo en u_nombre, u_apellido1 y u_apellido2.
- En la tabla vuelo, se dividió la fecha_vuelo en v_dia, v_mes y v_año.
- En promoción, se dividió la fecha_inicio en p_dia, p_mes y p_año.
- En la tabla pasajero, se dividió el nombre completo en p_nombre, p_apellido1 y p_apellido2.
- En tiquete, se dividió la fecha_emision en t_dia, t_mes y t_año.



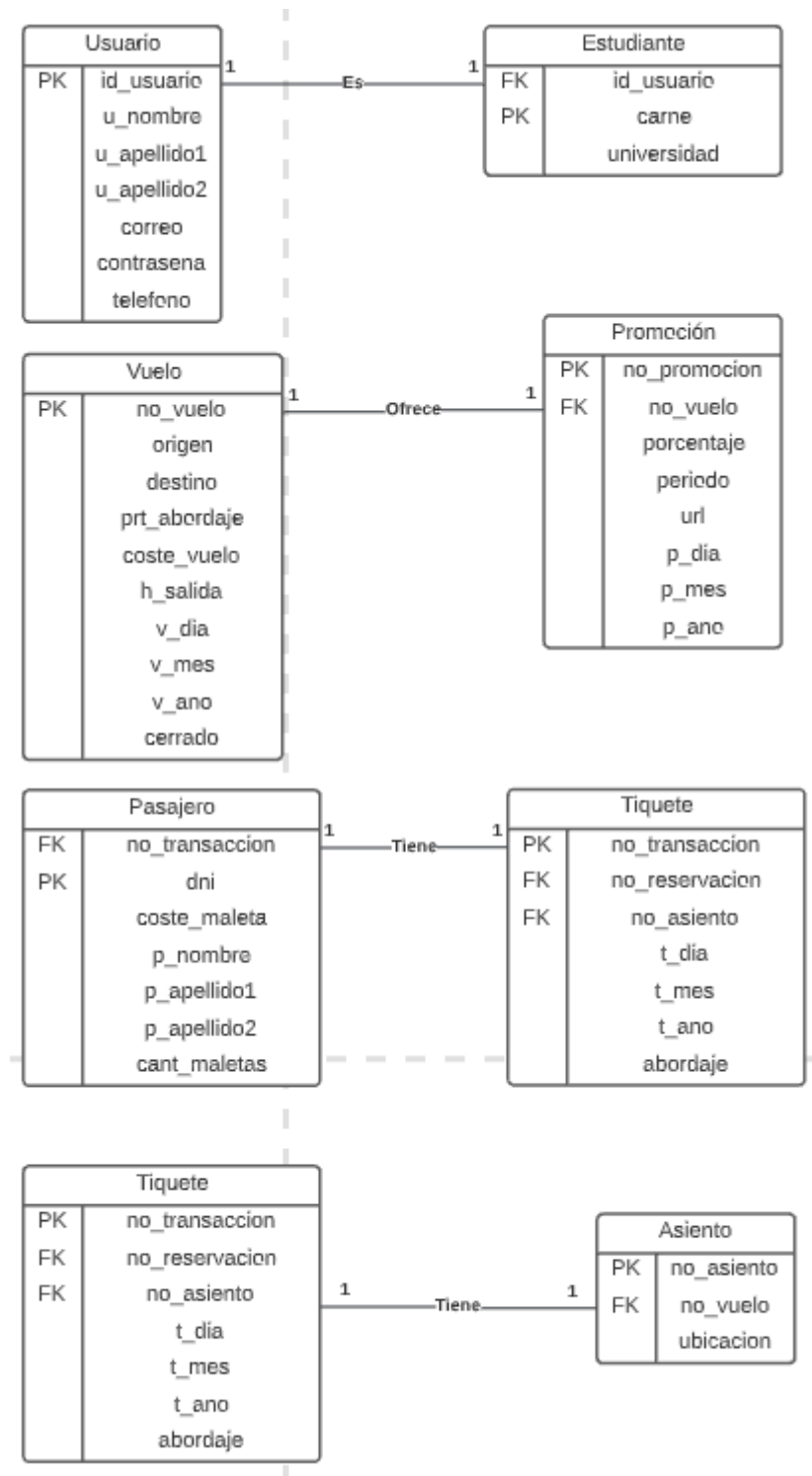
El segundo paso fue identificar y mapear las entidades débiles, en este caso se definieron como débiles:

- Estudiante dependiente de la entidad usuario, ya que un estudiante es un pasajero, por lo tanto, no puede existir sin la entidad de usuario.
- Pasajero dependiente de la entidad tiquete porque no puede existir un pasajero si no se ha generado un número de transacción de tiquete comprado.
- Maleta dependiente de la entidad pasajero, ya que no puede existir una maleta si no tiene un pasajero como dueño.
- Promoción dependiente de la entidad vuelo, ya que no puede crearse una promoción si no existe un vuelo primero.

Al escribir los atributos de estas entidades, se agregan como llaves foráneas (FK) las llaves primarias (PK) de las entidades de las cuales dependen. Como se puede observar en la siguiente imagen, maleta tiene como FK el dni de pasajero, pasajero tiene como FK el no_transaccion de tiquete, promoción tiene FK a no_vuelo de la entidad vuelo y estudiante tiene FK a id_usuario de la entidad usuario.



Seguidamente se realizó el mapeo de cuatro asociaciones binarias 1:1, en el cual cambia la disposición de las tablas para poder mostrar el tipo de llave PK o FK. Se identificaron las relaciones 1:1: usuario y estudiante asociadas por la llave foránea id_usuario, vuelo y promoción por medio de la FK no_vuelo, pasajero y tiquete por medio del no_transacción, y entre tiquete y asiento por la FK no_asiento.



Como cuarto paso, se mapearon diez asociaciones binarias 1:N, colocando llaves foráneas del lado de N a la llave primaria del lado 1:

- Relación 1 usuario reserva N pasajeros: Pasajero posee una FK a id_usuario de la entidad usuario.
- Relación 1 usuario realiza N reservaciones: Reservación tiene una FK a id_usuario de entidad usuario.
- Relación 1 pasajero posee N maletas: Maleta tiene una FK a dni de entidad pasajero.
- Relación 1 vuelo almacena N maletas: Maleta tiene una FK a no_vuelo de entidad vuelo.
- Relación 1 avión es asignado a N vuelos: Vuelo tiene una FK a matricula de entidad avion.
- Relación 1 vuelo realiza escala en N escalas: Escala tiene una FK a no_vuelo de entidad vuelo.
- Relación 1 vuelo gestiona N asientos: Asiento tiene una FK a no_vuelo de entidad vuelo.
- Relación 1 vuelo asigna N reservaciones: Reservacion tiene una FK a no_vuelo de entidad vuelo.
- Relación 1 reservación genera en N tickets: Ticket tiene una FK a no_reservacion de entidad reservacion.
- Relación 1 trabajador realiza N reservaciones: Reservacion tiene una FK a id_trabajador de entidad trabajador.
- Relación 1 trabajador realiza N reservaciones: Reservacion tiene una FK a id_trabajador de entidad trabajador.

Los últimos dos pasos serían realizar el mapeo de asociaciones N:M y de atributos multivaluados, sin embargo, en este caso no se presentan dicho tipo de relaciones ni de estos atributos.

En el modelo conceptual, existe un atributo derivado en la entidad pasajero, coste_maletas, el cual no se debe tomar en cuenta en el mapeo a modelo relacional.

El **modelo relacional completo** se adjunta al final del documento y en PDF en el repositorio, para mayor calidad de imagen.

Descripción de las estructuras de datos desarrolladas (Relaciones)

La base de datos está compuesta por las entidades usuario, estudiante, trabajador, avión, vuelo, escala, asiento, promoción, reservación, tiquete, pasajero y maleta.

La entidad tiene atributos que guardan el nombre completo, el correo, la contraseña y el teléfono de cada usuario que se registra en el sistema. Además, pregunta si el usuario es un estudiante para poder los datos de la universidad en la que estudia y su carnet. El trabajador, del cual solo se guarda su contraseña, se encarga de administrar varias de las demás entidades, como el vuelo y las promociones para mencionar algunas. La última de las entidades sin llave foránea es el avión, que contiene su nombre y la capacidad de personas que puede contener.

Las demás entidades contienen una llave foránea que las relaciona con alguna otra entidad. Por ejemplo la entidad vuelo, que contiene información del avión al que se le asigna el vuelo que vaya a realizar, además de contener datos de la ruta que recorre (el punto de partida y el punto de llegada), el puerto de abordaje, la hora de salida y de llegada, la fecha del vuelo, su costo y si es cerrado o no.

Las siguientes tres entidades que se mencionan contienen a vuelo como llave foránea para obtener sus datos. La escala guarda el aeropuerto al que debe ir para hacer escala en la ruta del vuelo y el orden de las paradas que realice. El asiento debe guardar el número que lo ubica. Finalmente, la promoción es un descuento que el trabajador le asigna a ciertos vuelos. Guarda la fecha en la que es válida, el porcentaje de rebaja y un URL que tenga más detalles al respecto.

Finalmente, estas entidades contienen una gran variedad de llaves foráneas, con varias teniendo asignada más de una llave. La entidad reservación se encarga de reservar un espacio para que el usuario realice la compra de un tiquete, por lo que contiene datos del vuelo a abordar, del usuario que hace la reservación y del trabajador que administra la reservación. El tiquete contiene la fecha en la que se compró y un booleano que indica cuándo el usuario vaya a abordar el vuelo, además de datos de la reservación que se realizó y el asiento que se le asigna en el vuelo. La entidad pasajero guarda su nombre completo, la cantidad de maletas que lleve al viaje y un booleano que indica si ha realizado el check-in, además de los datos del tiquete que compró. Y la entidad maleta guarda el color y el peso que tiene la maleta, además del pasajero que la carga y el vuelo que vaya a abordar.

Descripción detallada de la arquitectura desarrollada

La arquitectura desarrollada consiste en dos vistas de una aplicación Web, vista reservaciones que utilizarán los clientes y vista aeropuerto que utilizarán los trabajadores de la aerolínea. La aplicación Web se conecta a un servicio REST API por medio de solicitudes http como GET, POST y PUT, con el fin de enviar y recibir datos.

Los datos se organizan en tablas en la base de datos PostgreSQL, la cual se conecta al REST API por medio del contexto y modelos de Entity Framework. El API interactúa con la base de datos por medio de directivas de PostgreSQL como SELECT, JOIN, UPDATE para obtener y guardar datos en las tablas.

Se cuenta con una aplicación móvil, la cual tiene una base de datos local en SQLite, en la cual almacena la información y realiza la sincronización con la base principal en PostgreSQL.

Problemas conocidos

Error en responsive

La página web presenta errores de responsive en ciertas secciones, principalmente cuando se reduce en exceso el navegador. Sin embargo, existen otras secciones que si realizan bien este proceso.

Error selección de asientos

En la página web el usuario puede seleccionar el asiento que desee sin embargo este actualmente no verifica si otra persona se encuentra en el mismo asiento y el layout de asientos que se muestra es estático por lo que no se puede cambiar independientemente de que el avión posea más o menos asientos.

Error al comprar promoción

En la web es posible crear promociones y que estas se vean reflejadas con anuncios en un carrusel sin embargo no es posible acceder a dichas ofertas, esto porque no se consideró realizar una tabla que almacenara este tipo de reservaciones especiales por lo que con las condiciones

actuales de la base de datos es imposible realizar esta modificación, debido al tiempo restante el equipo a preferido descartar esta función y únicamente se mostrara como escaparate.

Error al crear BottomNavigationView en el menú de la app

Se consideró poner un BottomNavigationView en la ventana principal de la aplicación móvil, pero se acudió a un menú con botones que resolvía el problema con mayor facilidad.

Problemas encontrados

Problema conexión API – Base de datos

Al realizar la conexión API – Base de datos PostgreSQL, se encontró el problema de que los recursos como tutoriales en video y en páginas web solo realizaban la conexión con Code First, y en el caso del proyecto, ya se tenía una base de datos existente en Postgres. Realizando más investigación, en la documentación del proveedor Npgsql.EntityFrameworkCore.PostgreSQL se encontró una manera de realizar el proceso inverso, es decir Database First utilizando el siguiente comando:

```
dotnet ef dbcontext scaffold "Host=my_host;Database=my_db;Username=my_user;Password=my_pw" Npgsql.EntityFrameworkCore.PostgreSQL
```

Sin embargo, al ejecutarlo en la terminal del proyecto, presentó el error de no reconocer Npgsql.EntityFrameworkCore.PostgreSQL como un comando válido.

```
PS C:\Users\yendr\Documents\2022\Bases de Datos\TECAir-CE-3101\TECAir_API> dotnet ef dbcontext scaffold "Host=localhost;Database=my_db;Username=my_user;Password=my_pw" Npgsql.EntityFrameworkCore.PostgreSQL
Could not execute because the specified command or file was not found.
Possible reasons for this include:
 * You misspelled a built-in dotnet command.
 * You intended to execute a .NET program, but dotnet-ef does not exist.
 * You intended to run a global tool, but a dotnet-prefix executable with this name could not be found on the PATH.
```

El problema fue solucionado ejecutando el siguiente comando, el cual instala .NET entity framework tool:

```
PM> dotnet tool install --global dotnet-ef
You can invoke the tool using the following command: dotnet-ef
Tool 'dotnet-ef' (version '6.0.4') was successfully installed.
```

Problema realizando el POST

Al realizar la solicitud POST a la base de datos, se encontraba la tabla y la columna respectiva, sin embargo, no se podía referenciar con el nombre asociado.

Error: response status is 500

Response body

```
Npgsql.PostgresException (0x80004005): 42703: no existe la columna «no_promocion»
```

```
Exception data:
Severity: ERROR
SqlState: 42703
MessageText: no existe la columna «no_promocion»
Hint: Hay una columna llamada «no_promocion» en la tabla «promocion», pero no puede ser referenciada desde esta parte de la consulta.
Position: 177
File: parse_relation.c
Line: 3599
Routine: errorMissingColumn
```

El problema se presentó por un error en el INSERT, cuando se insertaban los valores, el nombre de que se pasaba de parámetro en VALUES era distinto al nombre de los atributos del modelo generado por Entity Framework.

```
sql = @"
INSERT INTO promocion (no_promocion, porcentaje, periodo, url, p_dia, p_mes, p_ano, no_vuelo)
VALUES (@no_promocion,@porcentaje, @periodo, @url, @p_dia, @p_mes, @p_ano, @no_vuelo);
";
```

Fue solucionado utilizando el nombre de los atributos del modelo:

```
namespace TECAir_API
{
    9 references
    public partial class Promocion
    {
        [Key]
        4 references
        public decimal NoPromocion { get; set; }
        3 references
        public decimal Porcentaje { get; set; }
        3 references
        public decimal Periodo { get; set; }
        3 references
        public string Url { get; set; } = null!;
        3 references
        public string PDia { get; set; } = null!;
        3 references
        public string PMes { get; set; } = null!;
        3 references
        public string PAño { get; set; } = null!;
        4 references
        public decimal NoVuelo { get; set; }
    }
}

sql = @"
INSERT INTO promocion (no_promocion, porcentaje, periodo, url, p_dia, p_mes, p_ano, no_vuelo)
VALUES (@noPromocion,@porcentaje, @periodo, @url, @pDia, @pMes, @pAño, @noVuelo);
";
```

Problema al ejecutar las solicitudes

Cuando se ejecutaron POST o GETS iniciales se presentó el siguiente error:

Error: response status is 500

Response body

```
System.InvalidOperationException: Unable to resolve service for type 'TECAir_API.Database.Interface.IReservacion' while attempting to activate 'TECAir_API.Controllers.ReservacionController'.
at Microsoft.Extensions.DependencyInjection.ActivatorUtilities.GetService(IServiceProvider sp, Type type, Type requiredBy, Boolean isDefaultParameterRequired)
at lambda_method10(Closure , IServiceProvider , Object[] )
at Microsoft.AspNetCore.Mvc.Controllers.ControllerActivatorProvider.<>c__DisplayClass7_0.<CreateActivator>b__0(ControllerContext controllerContext)
at Microsoft.AspNetCore.Mvc.Controllers.ControllerFactoryProvider.<>c__DisplayClass6_0.<CreateControllerFactory>g__CreateController|0(ControllerContext controllerContext)
at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Object& state, Boolean& isCompleted)
at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeInnerFilterAsync()
--- End of stack trace from previous location ---
```

La operación no era válida porque no se había agregado el archivo de interfaz a los servicios

en el archivo Program.cs, el problema se solucionó agregando los archivos de interfaz para cada solicitud con AddScope en Program.cs:

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashvower
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var postgresSQLConnectionConfiguration = new PostgreSQLConfiguration(builder.Configuration);
builder.Services.AddSingleton(postgresSQLConnectionConfiguration);
builder.Services.AddScoped<IPromocion, PromocionRepository>();
builder.Services.AddScoped<IVuelo, VueloRepository>();
builder.Services.AddScoped<IMaletum, MaletumRepository>();
builder.Services.AddScoped<IReservacion, ReservacionRepository>();
```

Problema creación de base de datos SQLite

Al crear la base de datos en SQLite, el archivo generado por la app móvil, no mostraba ninguna tabla. Para resolver este problema, se probó generar la base de datos para cada tabla individualmente, y de esta manera entender cuál CREATE TABLE generaba el problema.

Realizando este proceso, se logró encontrar que el error estaba en la definición de foreign keys de la siguiente forma:

```
"FOREIGN KEY("+V_MATRICULA+") REFERENCES "+TABLE_AVION+"("+MATRICULA+")";
```

Por lo que se investigó sobre otra manera para declararlas y se encontró la siguiente:

```
V_MATRICULA + " TEXT REFERENCES " + TABLE_AVION + "(" + MATRICULA + ")";
```

Utilizando esta definición las tablas sí fueron creadas de manera correcta:

Name	Type	Schema
▼ Tables (13)		
> android_metadata		CREATE TABLE android_metadata
> asiento		CREATE TABLE asiento(no_asie
> avion		CREATE TABLE avion(matricule
> escala		CREATE TABLE escala(no_esce
> estudiante		CREATE TABLE estudiante(carr
> maleta		CREATE TABLE maleta(no_mal
> pasajero		CREATE TABLE pasajero(dni IN
> promocion		CREATE TABLE promocion(no_
> reservacion		CREATE TABLE reservacion(no
> tickete		CREATE TABLE tickete(no_trar
> trabajador		CREATE TABLE trabajador(id_t
> usuario		CREATE TABLE usuario(id_usu
> vuelo		CREATE TABLE vuelo(no_vuelo
Indices (0)		
Views (0)		
Triggers (0)		

Conclusiones


La arquitectura es de un nivel superior, permite resolver el problema planteado y cumplir los objetivos establecidos en el proyecto, implementando servicios de aplicación Web con tecnologías actuales como Angular, conexión por medio de un REST API desarrollado como un WEB API en ASP.net core, bases de datos de gran calidad y fáciles de utilizar por ser open source como Postgres, utilizando Entity Framework para realizar la conexión con el REST API, desarrollo de aplicaciones móviles en Android Studio, uno de los sistemas operativos más usados en dispositivos móviles.

De acuerdo con lo anterior, cabe mencionar que esta arquitectura y tecnologías son muy utilizadas en la industria de desarrollo de software tanto a nivel nacional como internacional, por lo que se adquirieron habilidades técnicas que se podrán utilizar en el ámbito profesional en el futuro.

Recomendaciones

- Mantener todas las branches del repositorio actualizadas para evitar conflictos de los archivos.
- Evitar subir archivos que generan los IDEs que contienen rutas locales de la computadora de cada desarrollador para evitar conflictos en los archivos.
- Crear las tablas de la base de datos en orden en el script de base de datos puede evitar errores de creación, ya que la base de datos no puede crear primero una tabla que posee una llave foránea a otra tabla que se crea después en el script.
- Se debe analizar si el proceso de conexión de base de datos con REST API por medio de Entity Framework es code first o database first, ya que los procesos se realizan de manera distinta.
- Es importante consultar la documentación de los lenguajes y tecnologías utilizadas en el desarrollo, ya que entre una versión y otra puede haber cambios en la estructura y forma de estos. Por ejemplo, en el presente proyecto, .NET 6 ya no cuenta con el archivo Startup.cs sino que cambió por Program.cs y cambió la estructura de este archivo y la forma de añadir algunos elementos como los servicios.

Bibliografía

- ASP.NET Core Dependency Injection error: Unable to resolve service for type while attempting to activate.* (2016, 30 noviembre). Stack Overflow.
<https://stackoverflow.com/questions/40900414/asp-net-core-dependency-injection-error-unable-to-resolve-service-for-type-whil>
- Calingasan, A. (2021, 13 abril). *.Net 5 Web API With Repository Pattern Docker and PostgreSQL*. Alexcodetuts. <https://alexcodetuts.com/2021/04/10/net-5-web-api-with-repository-pattern-docker-and-postgresql/>
- Chanphakeo, R. (2018, 17 mayo). *Getting Started with Entity Framework Core (PostgreSQL)*. Medium. <https://medium.com/@RobertKhou/getting-started-with-entity-framework-core-postgresql-c6fa09681624>
- DB Browser for SQLite*. (2021, 25 julio). DB4S. <https://sqlitebrowser.org>
- FlightConnections. (2022, 25 abril). *Vuelos directos desde San José (SJO)*.
<https://www.flightconnections.com/es/vuelos-desde-san-jos%C3%A9-sjo>
- God, P. [Patrick God]. (2021, 23 noviembre). *CRUD with a .NET 6 Web API & Entity Framework Core*  *Full Course* [Vídeo]. YouTube.
https://www.youtube.com/watch?v=Fbf_ua2t6v4
- Munonye, K. (2021, 5 junio). *How to Create REST API in .Net Using C# and Visual Studio*. Kindson The Genius. <https://www.kindsonthegenius.com/how-to-create-rest-api-in-net-using-c-and-visual-studio/>
- Munonye, K. [Kindson The Tech Pro]. (2021, 9 junio). *Build your First REST API in Net with C# in Visual Studio - Step by Step* [Vídeo]. YouTube.
<https://www.youtube.com/watch?v=SDHILmkdd3s>
- Npgsql Entity Framework Core Provider / Npgsql Documentation*. (s. f.). Npgsql.
<https://www.npgsql.org/efcore/#using-an-existing-database-database-first>

Robles, M. [Códigos de Programación]. (2021, 10 marzo). *Crear base de datos en Android Studio (SQLite)* [Vídeo]. YouTube.

<https://www.youtube.com/watch?v=iWQIXjQ8ucA>

Tshibanda, P. [TechWithPat]. (2022, 5 enero). *How to connect to a PostgreSQL using Entity Framework Core (.NET 6)* [Vídeo]. YouTube.

<https://www.youtube.com/watch?v=yGwGLcqcJ6Q>

