Estimados, está adjuntado en la solapa de EVALUACIÓN PARA LA ACREDITACIÓN el parcial Nro 2 de tecnología. Tiene fecha límite el 9 de julio a última hora y se corregirá durante esa semana. Entregar la solución propuesta dentro de un archivo .zip o .rar, aclarando en un documento word en qué parte del código fuente se resuelve cada uno de los puntos requeridos en la sección de práctica.

# Notas

* En Wolny.P.Api, dentro de su appsettings.json, está la connection string. Para facilitar, se decidió usar al SQL Local Db que viene con el Visual Studio, puede usarse SQL Server indistintamente.
* Se puede utilizar el comando “update-database” para generar la base de datos, las tablas y los datos. El proyecto por default/predeterminado debe ser Wolny.P.Infrastructure. Si tiene puesto el startup de múltiples proyectos, puede que el comando falle por limitaciones de EF Core, solamente para ejecutar el “update-database”, fijar a Wolny.P.Api como proyecto startup
* Alternativamente, se deja un script “Wolny parcial dos.sql” que se puede ejecutar en un SQL Server que tiene la creación de la base, las tablas y sus datos
* Una vez que la bd esté cargada, se puede establecer como startup múltiple a los proyectos Wolny.P.Api y Wolny.P.Web
* En la landing de Wolny.P.Web está (no muy brevemente disculpas) explicado el funcionamiento del sistema

Punto 1

En Wolny.P.Application/Services/RecorridoService por la línea 109 se encuentra el método GenerarRecorrido que recibe un modelo con los datos necesarios

* Listado de Pedidos/Ciudades [Ciudades que se pueden repetir]
* Id de un camión: es opcional ya que un recorrido puede o no puede tener necesariamente en un inicio un camión asignado
* AlgoritmoEnum: con qué algoritmo se realiza el recorrido
* OrigenCiudadId: en qué ciudad debe iniciar el recorrido (será el principio y fin de éste)

El método hace las validaciones correspondientes y en la línea 156 se dirige al método ElegirAlgoritmo que se encuentra en la línea 191 y que esencialmente es un switch que apunta a los distintos algoritmos posibles, el primero de fuerza bruta y el resto heurísticos. Se eliminan las ciudades duplicadas, se recorren todos los destinos y se retorna al origen.

Punto 2

En Wolny.P.Application/Services/RecorridoService por la línea 33 se encuentra el método GetPuntoDos que recibe un modelo con los datos necesarios

* Top: para traer solamente 10 recorridos
* Ascending: para el ordenamiento
* Patente
* Finalizado: para indicar si el recorrido fue realizado

Se hace uso de EF Core para buscar con Linq/Where según los parámetros requeridos

Punto 3

En Wolny.P.Application/Services/CamionService por la línea 30 se encuentra el método GetPuntoTres que recibe un booleano disponible que indica si el Camion se encuentra disponible para hacer un viaje (o no). En este caso, el booleano será true.

Se hace uso de EF Core para buscar con Linq/Where según la disponibilidad del camión

Punto 4

Aprovechando el punto anterior, se usa el mismo método pero con el booleano disponible en false.

Punto 5

En Wolny.P.Web/Views/Home/Details.cshtml por la línea 68 inicia el script que carga los datos pertinentes en Google Maps Platform haciendo uso de GCP para poder ver la información de un recorrido que fue realizado en el punto 1. Link de ejemplo ([Details - Wolny.P.Web](https://localhost:7056/Home/Details/1) <https://localhost:7056/Home/Details/1>)

Punto 6

Se hace uso del mismo recurso que en el punto anterior. Para navegar hacia allí, se parte tal como dice el enunciado, de la vista del punto 2 (Wolny.P.Web/Views/Home/PuntoDos.cshtml, link de ejemplo [PuntoDos - Wolny.P.Web](https://localhost:7056/Home/PuntoDos?patente=XYZ0) <https://localhost:7056/Home/PuntoDos?patente=XYZ0>) y se hace clic dentro del apartado “Id e info del recorrido” que redirige a la funcionalidad del punto 1. Link de ejemplo ([Details - Wolny.P.Web](https://localhost:7056/Home/Details/1) <https://localhost:7056/Home/Details/1>)

Punto 7