

ALPESCARB Informe de implementación

Curso: SISTRANS / ISIS2304

Equipo: Juan David Chavez - 202323264

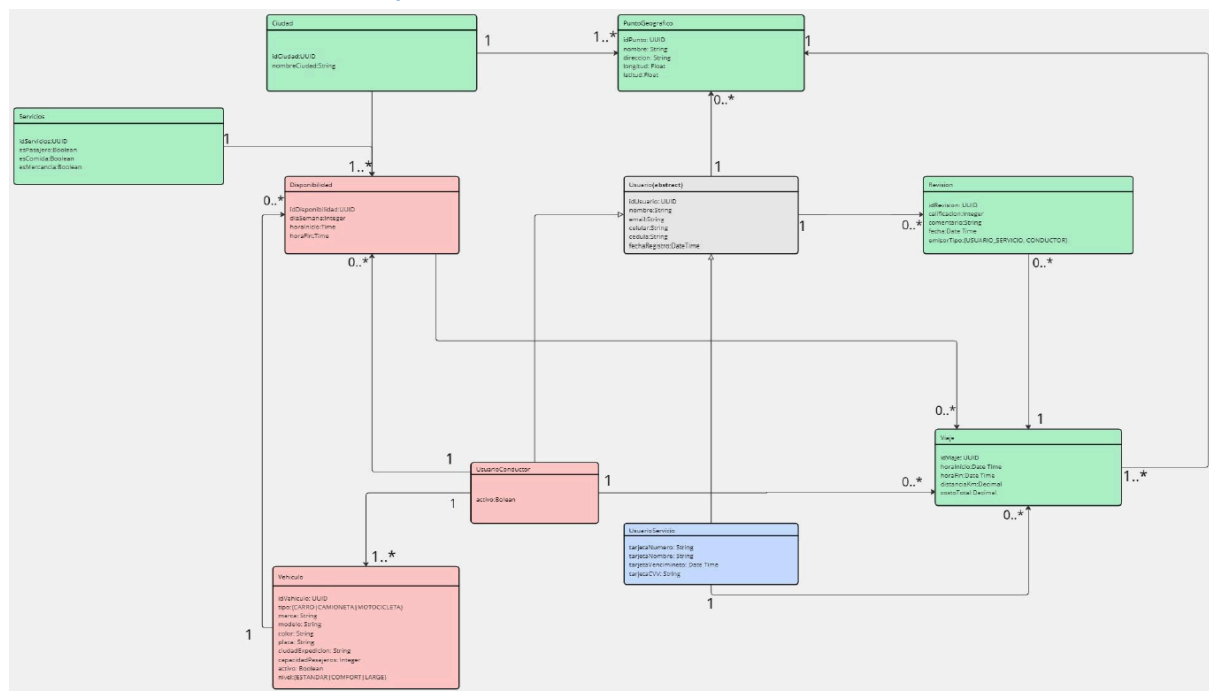
David Ortiz -

Santiago López -

En este documento se presentará el modelo de datos, la arquitectura e implementación de la aplicación del proyecto. correspondiente a los requerimientos funcionales RF 1-RF 11 y los requerimientos de consulta RFC 1-RFC 4

UML(Se le aplicaron las correcciones sugeridas).

https://miro.com/app/board/uXjVluZA4JM=?share_link_id=844327526011



Tablas

- Usuario
- Usuario Servicio
- Usuario Conductor
- Vehículo
- Viaje
- Revisión
- Servicio
- Disponibilidad
- Punto geográfico
- Ciudad

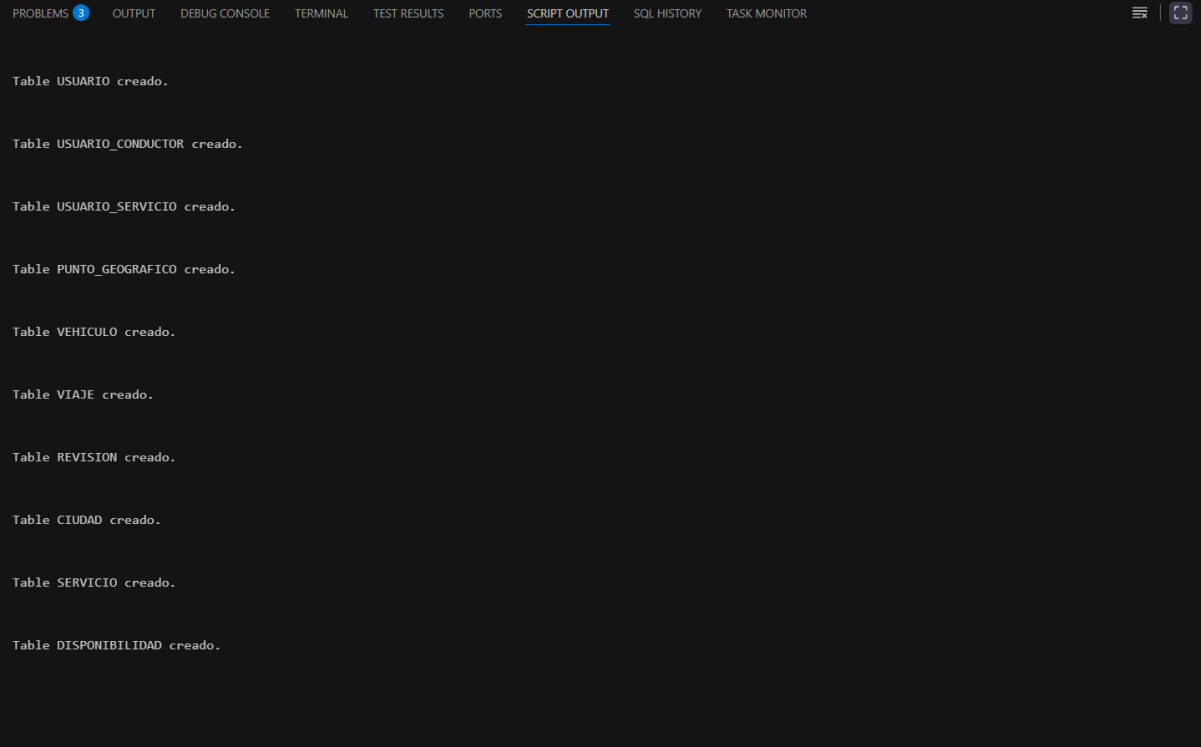
Justificación

Tabla Usuario: La tabla usuario fue creada para estar acorde la clase del modelo UML, esta tiene un identificador único, y sus respectivos atributos como lo son nombre, email, celular, cédula y fecha de registro. El tipo de cada uno de estos valores se respeto del modelo UML. Tabla Usuario Servicio: La tabla de usuario servicio, es la herencia de usuario, tiene el rol de poder almacenar los datos de todos los usuarios que son pasajeros.

Tabla Usuario Conductor: La tabla de usuarios conductores, es la herencia del usuario sobre todo la información de su actividad en la plataforma.

Tabla Vehiculo: Esta tabla fue creada, para almacenar la información de los vehículos, sobre su ubicación y su tipo

Evidencia Ejecución de las tablas:



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS SCRIPT OUTPUT SQL HISTORY TASK MONITOR

Table USUARIO creado.

Table USUARIO_CONDUCTOR creado.

Table USUARIO_SERVICIO creado.

Table PUNTO_GEOGRAFICO creado.

Table VEHICULO creado.

Table VIAJE creado.

Table REVISION creado.

Table CIUDAD creado.

Table SERVICIO creado.

Table DISPONIBILIDAD creado.
```

Normalización

Este documento está dedicado a evidenciar y justificar la correcta implementación de la normalización dentro de nuestro modelo E/R, demostrado en los niveles 1FN, 2FN, 3FN y FNBC. Con ello, demostramos la optimización de nuestro modelo y su utilidad en los casos de inserción y búsqueda de datos a través de una base de datos basada en este sistema, Además se han añadido los cambios sugeridos en la primera entrega incluyendo la tabla Servicio y Ciudad.

1FN

- Atomicidad: Este modelo posee atomicidad en el sentido en el que no se presentan listas o conjuntos en ninguna celda. Esto mayormente se presenta en base a restricciones ideadas por el grupo que eluden la posibilidad de que haya múltiples datos en una misma celda. Por ejemplo: en ciertas celdas como email o celular para la entidad “Usuario” se supone que los usuarios solo dan un email y un celular por persona. Y en cuanto al caso de los viajes donde puede haber múltiples usuarios pasajeros por viaje se repiten los datos del viaje por cada usuario en el viaje.
- Uniformidad: Este modelo posee uniformidad debido a que todas las celdas en una misma columna manejan un único tipo de dato. Esto se evidencia dentro de la construcción del UML, donde a cada atributo se le ha dado un único de dato al cual se apegan en todas las instancias de las entidades.

- Identificación única: Esta característica se cumple bajo el hecho en el que todas las tablas tienen una única llave primaria cada una, siendo esta llave en todas las tablas una estandarizada identificación numérica, o en otras palabras un ID numérico. Por ende, este modelo cumple con los requerimientos para cumplir con la Primera Forma Normal.

2FN

- Ya se ha demostrado que el modelo cumple con la Primera Forma Normal, por lo cual tiene la posibilidad de cumplir con la Segunda Forma Normal.

- Dependencias parciales: Este modelo no posee dependencias parciales y en donde toda tabla tiene todos sus atributos dependiendo de solo la llave principal. Esto se evidencia en la estrategia implementada, en la cual se dividen los atributos presentes en el sistema según las entidades evidenciadas (Usuario, conductor, viaje, vehículo, etc.) donde cada entidad posee una tabla con sus atributos referentes a este. Y adjunto a esto, y por lo evidenciado en la demostración del 1FN, cada tabla posee una única llave que identifica a todos los demás atributos en la tabla.

Por ello, este modelo cumple con los requerimientos para cumplir con la Segunda Forma Normal.

3FN

- Ya se ha demostrado que el modelo cumple con la Segunda Forma Normal, por lo cual tiene la posibilidad de cumplir con la Tercera Forma Normal.

- Dependencias transitivas: El modelo no posee dependencias transitivas en el sentido en el que todos los atributos dentro de una tabla dependen de una única llaveprimaria, y no de algún otro atributo que no sea clave. Esto se evidencia debido a la misma estrategia implementada para cumplir con la 2FN donde cada entidad posee solo una llave primaria y no existe la presencia de otros atributos que puedan causar una dependencia transicional. Por ello, este modelo cumple con los requerimientos para cumplir con la Tercera Forma Normal.

FNBC

- Ya se ha demostrado que el modelo cumple con la Tercera Forma Normal, por lo cual tiene la posibilidad de cumplir con la Forma Normal Boyce-Codd.

- Superclave: El modelo ideado cumple con la condición de las superclaves, mediante la condición en la que las tablas de las entidades poseen una única clave que identifica toda la fila, la cual es conocida por ser un único atributo, y este es siempre el identificador respectivo de la entidad mediante el cual se puede llegar a cualquier atributo del sistema sea primo o no.

- Por ello, este modelo cumple con los requerimientos para cumplir con la Forma Normal Boyce-Codd.

Arquitectura de la aplicación

- **Stack:** Spring Boot + Web + JDBC (JdbcTemplate) + Oracle.

- **Capas:** Controller → Service → DAO → DB.
- **config:** propiedades de la aplicación con el url de oracle, usuario y contraseña

Implementación por requerimientos (RF)

Para cada RF incluimos: endpoint, body ejemplo, SQL/resumen de lógica y espacio para pantallazo

RF1 - Registrar ciudad

- Post/ ciudades
- Body: {nombre → cali}
- Salida: id creado
- Pantallazo (Postman): [PEGA CAPTURA]

RF2 – Registrar usuario de servicios

- POST /usuarios/servicio
- Salida: id creado.
- Pantallazo: [PEGA CAPTURA]

RF3 – Registrar usuario conductor

- POST /usuarios/conductor
- Body: { "nombre":"Luis", "correo":"...", "telefono":"...", "cedula":"...", "licencia":"A123", "expira":"2030-12-31" }
- Salida: id creado.
- Pantallazo: [PEGA CAPTURA]

RF4 – Registrar vehículo

- POST /vehiculos
- Body: {
"tipo":"CARRO", "marca":"Chevy", "modelo":2018, "color":"Rojo", "placa":"ABC123", "capacidad":4, "conductorId":1 }

RF5 – Registrar disponibilidad del conductor

- POST /disponibilidades
- Body: { "conductorId":1, "inicio":"2025-09-29T08:00:00Z", "fin":"2025-09-29T12:00:00Z", "servicioId":1 }
- Validación: no solape (se rechaza con 409 si se cruza).
- Pantallazo: [PEGA CAPTURA]

RF6 – Modificar disponibilidad

PUT /disponibilidades/{id}

Body: igual a RF5.

Pantallazo: [PEGA CAPTURA]

RF7 – Registrar punto geográfico

POST /puntos

Body: { "nombre":"Origen", "direccion":"Calle 1", "lat":3.451, "lon":-76.532, "ciudadId":1 }

RF8 – Solicitar servicio

POST /servicios/solicitar

Body: { "usuarioServiciold":2, "serviciold":1, "origenId":1, "destinold":2 }

Salida: id del VIAJE en estado ASIGNADO.

Pantallazo: [PEGA CAPTURA]

RF9 – Registrar viaje finalizado

- POST /viajes/{id}/finalizar
- Efecto: set FECHA_LLEGADA, calcula DISTANCIA_KM (Haversine), PRECIO = km * TARIFA_BASE, ESTADO='FINALIZADO'.
- Pantallazo: [PEGA CAPTURA]

RF10 – Dejar revisión

- POST /revisiones
- Body: { "vehiculold":1, "fecha":"2025-09-29T09:00:00Z", "tipo":"MANTENIMIENTO", "aprobada":true, "observacion":"..." }
-
- Nota: Si el docente exige reviews usuario⇌conductor separadas, se dividiría en tablas específicas.
- Pantallazo: [PEGA CAPTURA]

RF11 – Segunda revisión/otra

- POST /revisiones (mismo endpoint)
- Pantallazo: [PEGA CAPTURA]

5. Requerimientos de consulta (RFC)

RFC1 – Histórico de servicios de un usuario

- GET /reportes/historico-usuario?usuarioServiciold={id}
- Salida: lista de viajes, fechas, distancia, precio, estado, vehículo y servicio.
- Pantallazo: [PEGA CAPTURA]

RFC2 – Top 20 conductores por número de servicios

- GET /reportes/top-conductores
- Pantallazo: [PEGA CAPTURA]

RFC3 – Total de dinero por conductor y vehículo

- GET /reportes/ingresos-conductor?conductorId={id}
- Nota: considera 60% para el conductor sobre PRECIO.
- Pantallazo: [PEGA CAPTURA]

RFC4 – Utilización de servicios en una ciudad y rango de fechas

- GET/reportes/uso-ciudad?ciudadId={id}&desde=YYYY-MM-DDTHH:MM:SSZ&hasta=...
- Salida: conteos y porcentaje por tipo/nivel, ordenado desc.
- Pantallazo: [PEGA CAPTURA]

Escenarios y planes de pruebas

hay que preparar los datos primero, entonces se ejecuta el sql en oracle y los ids explícitos.

Caso	RF/RFC	Precondición	Acción	Resultado esperado
TC01	RF1	-	POST /ciudades	201 + id
TC02	RF2	-	POST /usuarios/servicio	201 + id
TC03	RF3	-	POST /usuarios/conductor	201 + id
TC04	RF4	TC03	POST /vehiculos	201 + id
TC05	RF5	TC03	POST /disponibilidades	201 + id
TC06	RF6	TC05	PUT /disponibilidades/{id}	200
TC07	RF7	TC01	POST /puntos	201 + id
TC08	RF8	TC02, TC04, TC07	POST /servicios/solicitar	201 + id VIAJE
TC09	RF9	TC08	POST /viajes/{id}/finalizar	200 + DISTANCIA/PRICE
TC10	RF10	TC04	POST /revisiones	201 + id
TC11	RFC1	TC02	GET /reportes/historico-usuario	200 + lista
TC12	RFC2	TC08..09	GET /reportes/top-conductores	200 + top
TC13	RFC3	TC09	GET /reportes/ingresos-conductor	200 + agregados
TC14	RFC4	TC01, TC08	GET /reportes/uso-ciudad	200 + porcentajes

