Repair Stations

Inés Alarcón - 16008420

Objetivo

Las Repair Stations o centros de repuestos tienen como objetivo el registro y mantenimiento de inventario de los repuestos recibidos en cada bodega así como el historial y manejo de ordenes realizadas por los talleres.

Alcance

- Desarrollo de la plataforma web
- Diseño y creación de bases de datos que permita alto nivel de disponibilidad de la información
- Diseño y creación de formularios web para cada uno de los requerimientos funcionales

---

Funcionalidades del Producto

- Replicación: “Para reducir la latencia y aumentar la disponibilidad, almacenamos los datos cerca de los usuarios y servicios que los necesitan (la región más cercana para noostros es nam5 = United States). Esta configuración de ubicación es la ubicación de recursos predeterminada de Google Cloud Platform (GCP) de nuestro proyecto.
  Una ubicación de varias regiones, como lo es nam5, consta de un conjunto definido de regiones donde se almacenan varias réplicas de la base de datos. Cada réplica es una réplica de lectura y escritura que contiene todos los datos de la base de datos o una réplica testigo que no mantiene un conjunto completo de

datos pero participa en la replicación. Al replicar los datos entre varias regiones, los datos pueden continuar sirviéndose incluso con la pérdida de una región completa. Dentro de una región, los datos se replican en todas las zonas para que los datos puedan continuar sirviéndose dentro de esa región incluso con la pérdida de una zona."[1]

- Mantenimiento Web: El sistema debe de contar con la capacidad de eliminar, crear, actualizar y consultar registros por medio de una interfaz gráfica
- Servicios: El sistema debe contar con un servicio (servidor) que habilite la posibilidad de consumir información de nuestra base de datos.

[1] https://firebase.google.com/docs/firestore/locations?hl=en

---

Arquitectura

https://www.dropbox.com/s/fe4xil0hy0yt89y/Arquitectura%20Ines%20%281%29.pdf?dl=0

Modelo de Datos

Nuestros datos están organizados en la estructura de datos de Firebase, colecciones. Anidamos objetos complejos como mapas en los documentos y dentro de los documentos tenemos subcolecciones que permiten anidar detalles adicionales del documento principal.

- Colección "stations" → esta colección hace referencia a los centros de repuestos disponibles

```
{
        "name": "Repuestos ABC",
 (String)
        "latitud": 123456789,
(double)
        "longitud": 789456123
(double)
```

```
}
```

  - Nomenclatura estación: R - de repair, S - de station, “Tu inicial”
  - e.g: RSI (Repair Station Ines)

- Colección “gruas” → esta colección hace referencia a las grúas creadas para el servicio de mensajería

```
{
        "name": "Grua X",
 (String)
        "url": "gruax.com"
(String)
}
```

- Colección “linea” → esta colección hace referencia a todas las líneas de carros para las cuales tenemos repuestos

```
{
        "brand": "toyota"
 (String)
        "models": [
 (Array de Strings)
                "corolla",
                "yaris",
                "rav4",
        ]
}
```

- Colección “pieces” → esta colección hace referencia a todos los respuestos que tenemos en el inventario

```
{
        "name": "Bumper",
   (String)
        "articleId": "TY01",
  (String)
```

```
        "brand": "Toyota",
  (String)
        "model": "Yaris",
   (String)
        "year": 2010,
   (int)
        "price": 165,
   (double)
        "weight": 150,
  (double)
        "stock": 8,
   (int)
        "description": "Bumper Delantero"
   (String)
        "station": "stationCode",
   (String)
}
```

  - Nomenclatura respuestos: Inicial Marca, Inicial Modelo, ID número
  - e.g: Bumper Delantero, Toyota Yaris = TY01

- Colección “orders” → esta colección hace referencia a las ordenes creadas por taller

```
{
        "orderId": "Order-20123",
  (String)
        "issueDate": "03/11/2022",
 (String)
        "clientName": "Taller ABC",
  (String)
        "clientLatitud": 123456789,
  (double)
        "clientLongitud": 789456123,
 (double)
```

```
        "delivery": "Grúa XYZ"              (STRING si una rea
lizará la entrega o NULL
                                             si todavía falta
por asignar una grúa)
        "status": true,                      (TRUE si la orden
ya ha sido completada o
        FALSE falta por asignar grúa mensajera)
        "station": "stationCode",
  (String)
}
```

- Cada documento en “orders” tiene una subcolección “items” → esta lista todos los articulos agregados a la orden

```
{
        "item": "Bumper",
  (String)
        "itemId": "TY01",
  (String)
        "brand": "Toyota",
 (String)
        "model": "Yaris",
  (String)
        "year": 2010,
  (int)
        "price": 165,
  (double)
        "weight": 150,
 (double)
        "quantity": 123
  (int)
}
```

Anexos

- ¿Qué aprendimos?
  - A diseñar y manejar bases de datos
  - Como establecer protocolos de comunicación
    - Aprendimos a comunicar mejor nuestras necesidades con nuestros compañeros para que el proyecto funcionará
  - Como podemos aplicar lo aprendido en clase a ejemplos que serían válidos en un ambiente laboral o fuera de clase.
- ¿Qué podemos mejorar?
  - Agregar funcionalidades a nuestro proyecto:
    - Una sección de promociones:
      - Formulario para agregar promociones
      - Rutas en el servidor para el manejo de estas consultas
  - La comunicación entre nosotros, no dejar que miembros del equipo se queden atrás.

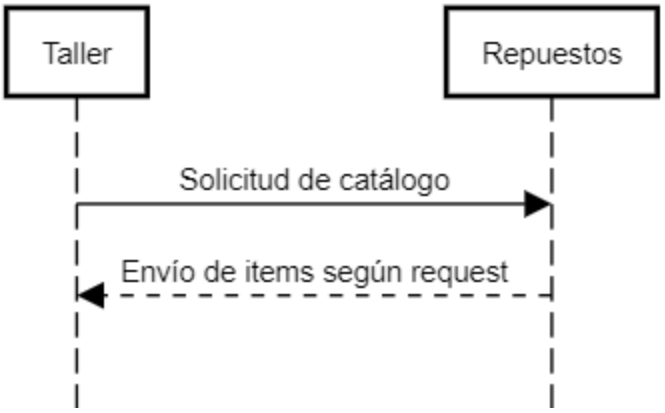
Repositorios

- Inés: https://github.com/InesAlarcon/CC6_RepairStation

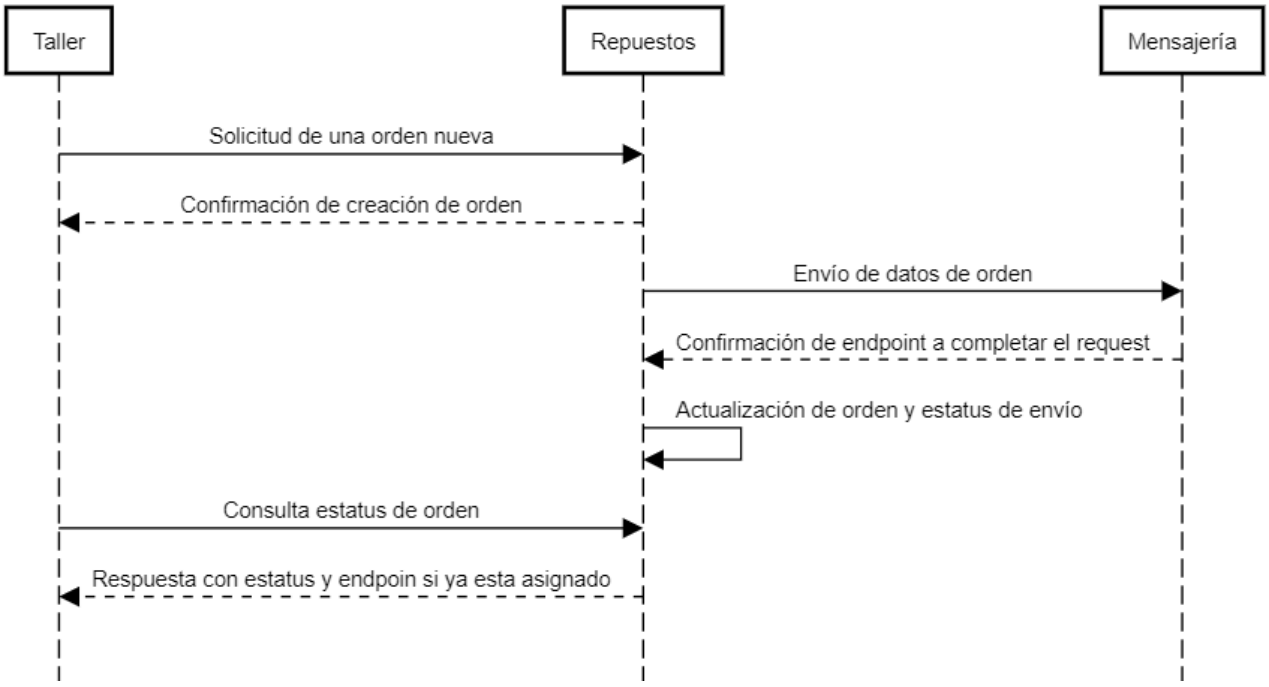
API Use

UML

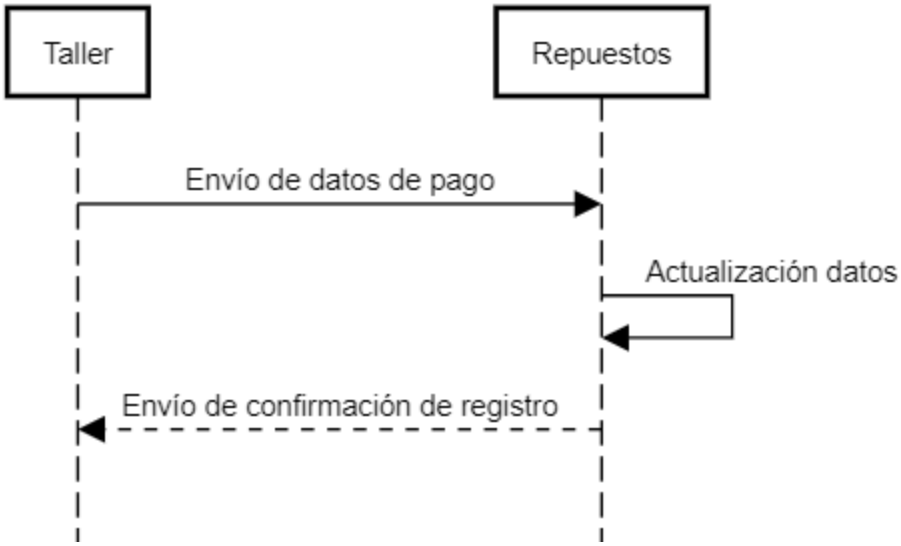
- Solicitud de catálogo



- Solicitud de orden nueva



- Registro de pago de una orden



Taller → Repuestos

- ¿Cómo solicitar el catálogo de nuestros productos?

```
POST a: /catalog
{
        "year": null,                (int para especificar año
o nulo si qui ren todo)
        "model": "Yaris",            (String o nulo si quieren
todo)
        "brand": "Toyota"            (String OBLIGATORIO)
}
```

Respuesta

```
{
        "data": [          (NULL si no existen artículos que ha
gan match con el
                            criterio o un ARRAY de OBJETOS si e
xisten artículos)
                {
```

```
                        "articleId": "TY01",
 (String)
                        "name": "Bumper",                (Stri
ng)
                        "price": 165,
  (double)
                        "stock": 10,
 (int)
                        "year": 2010,
  (int)
                        "weight": 15,
                        "model": "Yaris",
  (String)
                        "brand": "Toyota",
 (String)
                        "description": "asdfqwer"
  (String)
                        "station": "RSI",
                },
                .
                .
                .
                {
                        "articleId": "TY02",
 (String)
                        "articleName": "Retrovisor Izq.",
  (String)
                        "price": 65,
 (double)
                        "stock": 12,
 (int)
                        "year": 2014,
  (int)
```

```
                        "model": "Yaris",
  (String)
                        "brand": "Toyota",
 (String)
                        "description": "asdfqwer"
  (String)
                }
        ]
}
{
    "data": [
        {
            "year": 2010,
            "description": "Bumper Delantero",
            "weight": 15,
            "articleId": "TY01",
 (String)
            "station": "RSI",
  (String)
            "stock": 8,
  (int)
            "price": 111,
            "name": "Bumper",
            "model": "Yaris",
            "brand": "Toyota"
        },
        .
        .
        .
        {
            "articleId": "TH03",
```

```
            "weight": 1.2,
            "station": "RSI",
            "description": "Retrovisor Derecho",
            "name": "Retrovisor Derecho",
            "year": 2011,
            "brand": "Toyota",
            "model": "Hilux",
            "stock": 4,
            "price": 65
        }
    ]
}
```

- ¿Cómo hacer una orden?

```
POST a /order
{
        "items": [
                {
                        "articleId": "article-200",
  (String)
                        "quantity": 3
  (int)
                },
                .
                .
                .
                {
                        "articleId": "article-300",
  (String)
                        "quantity": 4
  (int)
```

```
                }
        ],
        "clientName": "Taller ABC",
  (String)
        "clientLatitud": 123456789,
  (double)
        "clientLongitud": 789456123
  (double)
}
```

Respuesta

```
{
        "orderId": "order-300",
  (String)
}
```

- ¿Cómo consultar si tu orden ya tiene una grúa asignada para el servicio de mensajería?

```
POST a /status
{
        "orderId": "order-300"
 (String)
}
```

Respuesta

```
{
        "status": true,                    (NULL si no existe
la orden, FALSE si sigu
      e procesandose y TRUE si ya existe un men
                            sajero asignado)
        "delivery": "Grua X"               (String si ya exis
te o null si no)
        "deliveryCost": "5.00",            (double NULL si au
n no tenemos el valor de delive
              ry)
```

```
        "paymentStatus": "Pagado",          (String NULL si au
n no tenemos el valor de delive
             ry)
        "deliveryDate": "14/11/2022",       (String NULL si au
n no tenemos el valor de delive
             ry)
        "deliveryServer": "Grua Test Ines", (String NULL si au
n no tenemos el valor de delive
             ry)
        "deliveryETA": "0.5"                (double NULL si au
n no tenemos el valor de delive
             ry)
        "totalCost": "50.50"                (double NULL si au
n no tenemos el valor de delive
}
```

- ¿Cómo hacer el registro de pago sobre una orden?

```
POST a /payment
{
        "orderId": "order-300"
 (String)
        "paymentId": 123
 (int)
}
```

Respuesta

```
{
        "orderId": "order-300"
 (String)
        "paymentId": 123,
  (int)
        "status": true
 (bool)
}
```

Respuestos → Mensajería

- Envío de orden para solicitud de servicio

```
POST a /delivery
{
        "orderId": "Some Order ID",
   (String)
        "quantityArticles": 123
   (int)
        "totalPrice": 1234596789
  (double)
        "totalWeight": 123456789,
   (double)
        "pickupLatitud": 123456789,
   (double)
        "pickupLongitud": 123456789,
  (double)
        "pickupName": "RSI",
  (String)
        "clientName": "Taller XYZ",
   (String)
        "deliveryLatitud": 123456789,
   (double)
        "deliveryLongitud": 123456789,
  (double)
}
```

Respuesta

```
status code 200      (si fue asignado correctamente a una grú
a)
```

- Solicitud de estatus de una orden

```
POST a /orden
{
```

```
        "orderId": "Some Order ID",
   (String)
}
```

Respuesta

```
Sin error:
{
        "orderId": "Some Order ID",
   (String)
        "estado": NUEVO
   (String)
        "costoTotal": 1234596789
  (double)
        "costoEnvio": 123456789,
  (double)
        "fechaEntrega": "17/11/2022",
   (String)
        "servidor": "SERVER_XYZ",
   (String)
        "deliveryETA": 123456789
  (double)
}
Con error:
{
    "success": false,
    "message": "Ocurrio un error al tratar de encontrar el pag
o con ordenID RSI-1668728630277",
    "errors": {
        "error": "No se encontro ningun pago con ese identific
ador de orden"
    }
}
```

- Envío de pago

```
POST a /pago
{
        "orderId": "Some Order ID",
   (String)
}
```

Respusta

```
{
        "orderId": "Some Order ID",
   (String)
        "estado": PAGADO
  (String)
        "costoTotal": 1234596789
  (double)
        "costoEnvio": 123456789,
  (double)
        "fechaEntrega": "17/11/2022",
   (String)
        "servidor": "SERVER_XYZ",
   (String)
        "deliveryETA": 123456789
  (double)
}
```

Errores

- Si por alguna razón ocurre un error en el servidor, se enviará un status code = 500 o 400
- Si todo fue un éxito, la respuesta siempre será un JSON