Prueba Técnica Desarrollador Shipit

Versión 2.0

https://shipit.cl

Problemática

Debemos generar envíos con ciertos datos, para ello, dejamos más abajo toda la información necesaria para llevar a cabo este objetivo.

Para llevar a cabo lo mencionado anteriormente, necesitamos cumplir algunos pre-requisitos, como conocer los orígenes, destinos, poder cotizar el costo de envío hacia un x destino e implementar una interfaz gráfica que nos permita interactuar con todas estas aristas para que finalmente, podamos guardar el registro del envío generado con todos los datos y validaciones necesarias, todo esto se encuentra explicado a detalle más abajo.

Información útil sobre la API

En este link, está la documentación oficial de la API, sin embargo, en este apartado adjuntamos información específica que será útil para desarrollar la prueba.

Autenticación

Para consultar la API, es necesario contar con una autenticación que debe ser enviada vía headers, utilizando la siguiente información:

API Key

- X-Shipit-Email: prueba_front@shipit.cl
- X-Shipit-Access-Token: MWhEAdkHKYdscen_4cxR
- Accept: application/vnd.shipit.v4

Endpoints útiles

Consultar destinos

GET https://api.shipit.cl/v/communes

Consultar orígenes

GET https://api.shipit.cl/v/origins

Consultar costo de envío

POST https://api.shipit.cl/v/rates

Body ejemplo

```
{
   "parcel": {
        "length": 41,
        "width": 41,
        "height": 50,
        "weight": 9.5,
        "destiny_id": 18,
        "origin_id": 312,
        "is_payable": false,
        "destiny": "Domicilio",
        "courier_selected": false,
        "courier_for_client": "",
        "request_from": "calculator"
      }
}
```

Tecnologías a utilizar

Lenguajes

- Ruby.
- Javascript (Opcional).

Frameworks

- Rails.
- ReactJS (Opcional).

Motor de base de datos

- Postgresql.

Requerimientos funcionales

1. Modelar una tabla llamada ‘shipments’, con al menos las siguientes columnas:
    - name (Nombre del destinatario) # varchar
    - last_name (Apellido del destinatario) # varchar
    - email (Email del destinatario) # varchar
    - length (Largo del pedido) # int
    - height (Alto del pedido) # int
    - width (Ancho del pedido) # int
    - weight (Peso del pedido) # int
    - courier (Courier que llevará el pedido) # varchar
    - price (Precio del envío) # int
2. Modelar una tabla llamada ‘addresses’ con al menos las siguientes columnas:
    - destiny_id (Fk de la comuna hacia donde va el pedido) # int
    - shipment_id (Fk para relacionar a la tabla ‘shipments’) # int
3. Modelar una tabla llamada ‘destinies’ con al menos las siguientes columnas:
    - country_name (País al que pertenece el destino) # varchar
    - external_id (Id de la comuna) # int
    - name (Nombre de la comuna) # varchar
4. Poblar la tabla ‘destinies’ con la información obtenida vía API.
5. Generar una interfaz gráfica, con un desplegable donde se listen todos los destinos obtenidos desde la tabla ‘destinies’, además de 4 inputs donde se pueda ingresar: largo, alto, ancho y peso.
6. La interfaz gráfica debe tener 2 botones
    - Botón 1: Llamado ‘cotizar’. Debe obtener el costo de enviar el pedido al destino seleccionado. (Para el origin_id de la cotización, se deben consultar los orígenes y utilizar el sub atributo ‘commune_id’ del origen que tenga el sub atributo ‘default’ como ‘true’.
    - Botón 2: Llamado ‘Crear envío’. Debe abrir una interfaz en la misma página (O no) que permita ingresar los datos faltantes para ser almacenados en la tabla ‘shipments’. (Se deben conservar: largo, alto, ancho, peso e id de destino utilizado en la cotización) + 1 botón llamado ‘Guardar’ que se encargará de crear el registro en la tabla ‘shipments’ junto a sus registros relacionados.
7. Generar una tabla donde se listen todos los envíos creados. La tabla debe contener una columna por cada columna de la tabla ‘shipments’ + una columna para mostrar el nombre de la comuna (destino) hacia donde va el envío.

Requerimientos no funcionales

1. Debes utilizar git para versionar tu código
2. Debe haber una rama por cada funcionalidad/mejora/bug implementada.
3. Los commits deben ser descriptivos en base al código contenido.
4. Desplegar el proyecto en un proveedor cloud para que sea accesible desde internet. (Opcional).
5. Generar un readme para poder levantar el proyecto en localhot y poder probar los requerimientos solicitados.
6. Todas las consultas a la base de datos deben ser realizadas vía ORM.

Detalle de la entrega

- El plazo para entregar el proyecto es hasta las 23:59 horas del 7 día, contando a partir del día siguiente en que fue enviada la prueba vía correo electrónico.
- El repositorio debe ser privado, y se debe enviar una invitación como colaborador a @HaNdlezz y @gcamposm a través de GitHub.

Notas informativas

Para dudas o comentarios, puedes responder el mismo correo donde recibiste la prueba adjunta utilizando “responder a todos”.

Si no terminas la prueba, te pedimos enviarla igualmente, ya que posiblemente en base a tu entregable, tendremos una evaluación de tipo analítica en conjunto.