API TEST AND CORRECTIONS

Todo el avance está subido a github, rama: dianaBranch
Todas las pruebas de postman están ordenadas por carpetas en el environment compartido

---

Orden de pruebas (endpoints y dependencias)

0) Preparación (una vez)

- Crea en Postman un **Environment** con:
  - base_url = http://127.0.0.1:8000
  - token = (vacío al inicio; luego lo pagarás)

---

1) Salud del servicio (público)

1. **GET** {{base_url}}/api/health
   - Esperado: 200 { status: "ok", ... } OK

2) Autenticación básica (público)

2. **POST** {{base_url}}/api/register OK
   - Crea usuario A (forwarder) y usuario B (carrier).

3. **POST** {{base_url}}/api/login OK
   - Con cada usuario creado: obtener token (Sanctum).
   - Guarda el token en el env (token) para el resto de pruebas.

3) Catálogos públicos (público)

4. **GET** {{base_url}}/api/ofertas (ruta) — listado (público) OK (arreglado, pedía phone en user cuando no existe eso en la tabla, por tanto se quito)
5. **GET** {{base_url}}/api/ofertas/{id} — detalle (público) OK
6. **GET** {{base_url}}/api/ofertas_carga — listado (público) OK (tenia mismo problema con phone)
7. **GET** {{base_url}}/api/ofertas_carga/{id} — detalle (público) OK (arregle que siempre devolvía puro null, ahora devuelve información correctamente)
8. **GET** {{base_url}}/api/cargo-types OK
9. **GET** {{base_url}}/api/truck-types OK

4) Subida de documentos (solo Sanctum, sin CheckUserStatus)

Estas rutas deben funcionar **aunque el usuario no esté verificado**, pero sí logueado.

10. **POST** {{base_url}}/api/documents/upload-batch OK
    - Body (form-data): cedula (File), licencia (File)
    - Esperado: 201 con documents[], path relativo y url absoluta /documents/...

11. **POST** {{base_url}}/api/documents/upload OK
    - Body (form-data): document_key = cedula|licencia, file (File)
    - Esperado: 201 con document

12. **Errores controlados OK**:
    - Sin token → 401
    - Extensión inválida / >2MB → 422
    - Faltar un archivo en batch → 422

Esta subida de documentos es nueva e implementada recientemente, por tanto ya sabía su funcionamiento y se hicieron todas las pruebas correctamente

5) Rutas protegidas con auth:sanctum + CheckUserStatus

- A partir de aquí, si el usuario está bloqueado, debe bloquear/redirigir (según tu middleware).
- Probe con usuarios bloqueados también, y arregle errores del middleware, ahora devuelve el mensaje correctamente cuando está bloqueado

5.1 Usuario actual y logout

13. **GET** {{base_url}}/api/user OK
    - Con token de usuario **bloqueado**→ valida el comportamiento del middleware.
    - Con token de usuario **activo**→ debe responder 200 con datos del usuario.

14. **POST** {{base_url}}/api/auth/logout OK
    - Token invalida → verificar respuesta.

(Vuelve a loguearte para seguir.)

5.2 Imágenes (solo administradores, según tu controlador)

15. **GET** {{base_url}}/api/images/{id} OK
16. **GET** {{base_url}}/api/images/{id}/check OK

- Esperado: no deja por no autorización (tuve que arreglar que no existía ninguna función verificación de si es admin o no)

5.3 Ofertas (CRUD autenticado)

Estas son **protected** salvo index/show.

17. **POST** {{base_url}}/api/ofertas — crear oferta **ruta OK**
18. **PUT** {{base_url}}/api/ofertas/{id} — actualizar OK

19. **DELETE** {{base_url}}/api/ofertas/{id} — borrar OK
20. **POST** {{base_url}}/api/ofertas_carga — crear oferta **carga OK**
21. **PUT** {{base_url}}/api/ofertas_carga/{id} — actualizar OK
22. **DELETE** {{base_url}}/api/ofertas_carga/{id} — borrar OK

- Probar con usuario bloqueado y no bloqueado (debe permitir).
- Validar 422 por campos requeridos faltantes.

- Tuve que arreglar la función authorize en general del proyecto para que verifique si el usuario que creó la oferta sea cual sea, sea el mismo que edita, borra, o lo que sea, antes daba error en estos métodos, ahora funcionan bien.
- Se agregó descripción en ambas tablas para dar consistencia.
- En ofertas carga no daba el authorize dejaba a todos actualizar y borrar, se arreglo eso y también que de descripción en ofertas ruta.

5.4 Bids (propuestas)

23. **GET** {{base_url}}/api/bids — mis bids OK
24. **POST** {{base_url}}/api/bids — crear bid OK
25. **PUT** {{base_url}}/api/bids/{bid} — actualizar bid OK
26. **POST** {{base_url}}/api/bids/{bid}/status — cambiar estado OK
27. **POST** {{base_url}}/api/bids/{bid}/accept — aceptar OK
28. **POST** {{base_url}}/api/bids/{bid}/reject — rechazar OK
29. **DELETE** {{base_url}}/api/bids/{bid} — eliminar OK
30. **GET** {{base_url}}/api/bids/received — bids recibidos OK

- Probar validaciones, autorizaciones (dueño vs no dueño), y transiciones válidas/inválidas.

- No había la función de borrado de un bid en el controller, se corrigió esto
- POST /api/bids/{bid}/status: Puede “aceptar” aunque ya exista otro aceptado. El endpoint /accept sí protege esto (con transacción). Lo que se hizo: alinear updateStatus para validar unicidad de aceptados igual que accept()

5.5 Chats

31. **GET** {{base_url}}/api/chats — lista OK
32. **GET** {{base_url}}/api/chats/{chat} — detalle OK
33. **POST** {{base_url}}/api/chats/{chat}/message — enviar mensaje OK
34. **GET** {{base_url}}/api/chats/{chat}/messages — recibir nuevos OK

- Probar acceso solo a chats donde el user participa.

5.6 Work Progress

35. **GET** {{base_url}}/api/work/{bid} — detalle progreso OK
36. **GET** {{base_url}}/api/work/{bid}/check-status — estado OK
37. **POST** {{base_url}}/api/work/{bid}/request-completion — solicitar finalización OK

38. **POST** {{base_url}}/api/work/{bid}/confirm-completion — confirmar OK
39. **POST** {{base_url}}/api/work/{bid}/reject-completion — rechazar OK

- Probar reglas de negocio (quién puede confirmar/rechazar, estados coherentes).

- Se corrigió el nombre de las rutas porque sino, no funcionaba correctamente el solicitar la finalización de trabajo, también se corrigió el que se guardaban columnas inexistentes en chats para mantener consistencia, ahora en la api funciona correctamente las peticiones.
- De igual forma, en las aceptaciones, no funcionaba porque la tabla de bids no tenía fecha de finalización, se agregó esto con migración, y ahora funciona correctamente todo el proceso de work progress.

OBSERVACIONES POR EL MOMENTO

- En la API, el usuario se crea sin rol definido, no se puede definir, y por defecto le pone forwarder, donde si se define es en el controlador de la web pero en el de la api no así que lo crea como por defecto así.

- Además, en la web pide datos de la empresa del usuario, pero en la web también la empresa es manejada como a parte,entonces hay inconsistencia ahí por que en la API en el usuario no pide nada de empresa, y en la api no hay empresa. Es por eso, que en la parte de registrar y empresas, debemos igualar la api, con la web.

- No puedo loguearme con el admin por ningún lado para obtener su token, ya que el login es para users nomas y no hay nada para administrators, se que esto está en otro proyecto por tanto no se puede hacer pruebas con esto, lo que si en imagenes verificamos que un usuario cualquiera no puede ingresar.

- Parece que falta un crud para las empresas, por que ahorita no existe

- Se arreglaron varios errores e inconsistencias de toda la API que se mencionan en el documento

COSAS QUE FALTAN

Luego de revisar toda la API queremos ver que controllers más existen en la web que puedan faltar en la api para ver de implementarlas o sugerir implementación.