Microservicio API carnet

Autores: Jesús Torres Mateo y Daniel Castilla Marín

Nivel de acabado: Básico

Análisis del esfuerzo:

Jesús → 83,5 horas dedicadas

Dani → 88 horas dedicadas

Justificación de cómo han ido consiguiendo cada uno de los requisitos del microservicio

1. **El backend debe ser una API REST tal como se ha visto en clase implementando al menos los métodos GET, POST, PUT y DELETE y devolviendo un conjunto de códigos de estado adecuado.**

El backend está compuesto por 8 métodos de los 4 tipos. Estos métodos están situados en el archivo *server.js* y se dividen en:

- 3 métodos GET:
  - "/": la página de inicio del proyecto
  - "/api/v1/carnets": muestra un listado de todos los carnets
  - "/api/v1/carnets/:DNI": muestra un dni
- 1 método POST:
  - "/api/v1/carnets": añade un carnet
- 3 métodos PUT:
  - "/api/v1/carnets/retire/:DNI": retira (poner a no válido) un carnet. Se hace cuando un conductor se ha quedado sin puntos
  - "/api/v1/carnets/revalidate/:DNI": vuelve a validar un carnet. Se hace cuando un conductor pasa de 0 puntos a >=1 puntos
  - "/api/v1/carnets/edit/:DNI": edita un carnet
- 1 método DELETE:
  - "/api/v1/carnets/remove/:DNI": elimina un carnet

Estos métodos devuelven un conjunto de códigos que se basan que serán:

- 200: Ok
- 201: Created
- 403: Forbidden
- 404: Not found
- 409: Conflict
- 500: Internal server error

2. **Debe tener un frontend que permita hacer todas las operaciones de la API (este frontend puede ser individual o estar integrado con el resto de frontends)**

   Tenemos un frontend desarrollado con react que es individual de nuestro microservicio

3. **Debe estar desplegado en la nube y ser accesible en una url.**

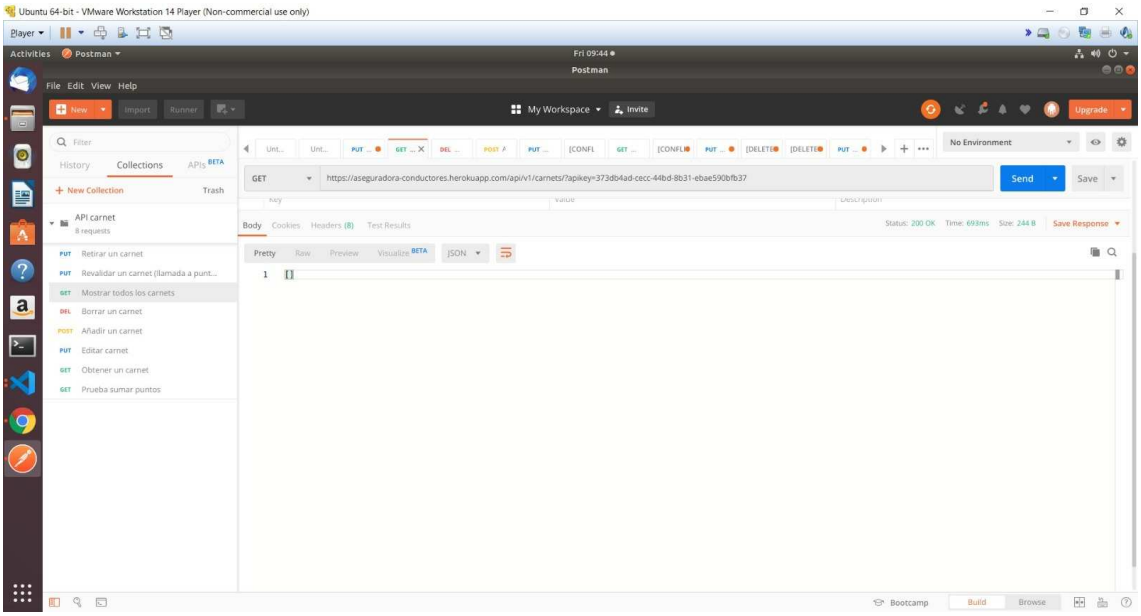
   El proyecto está desplegado en heroku y es accesible desde esta url: https://frontend-api-carnet.herokuapp.com/

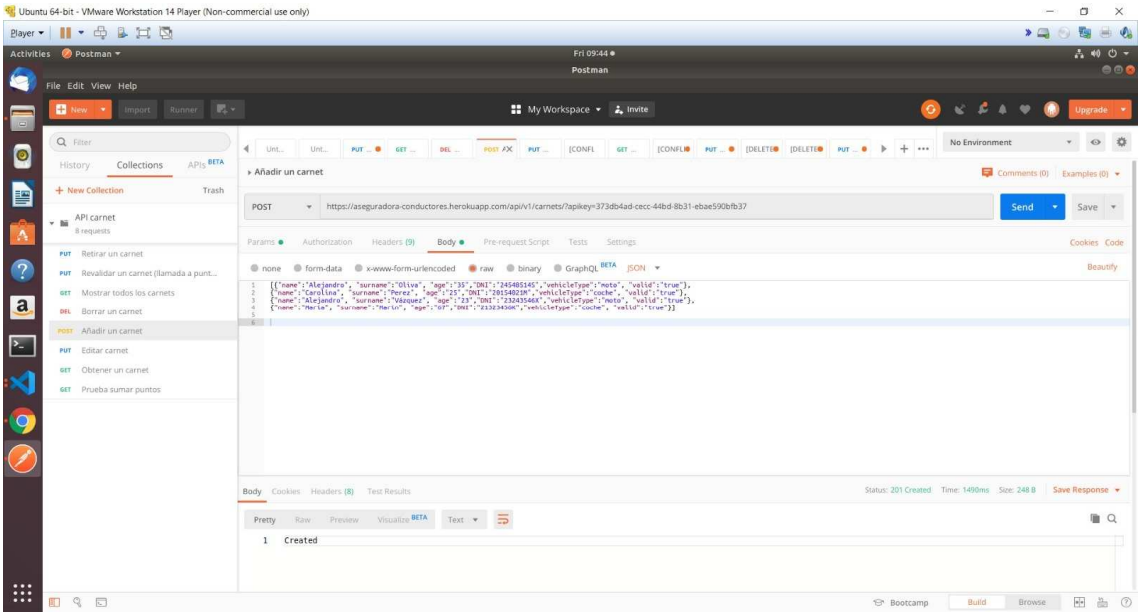
4. **La API que gestione el recurso también debe ser accesible en una dirección bien versionada.**

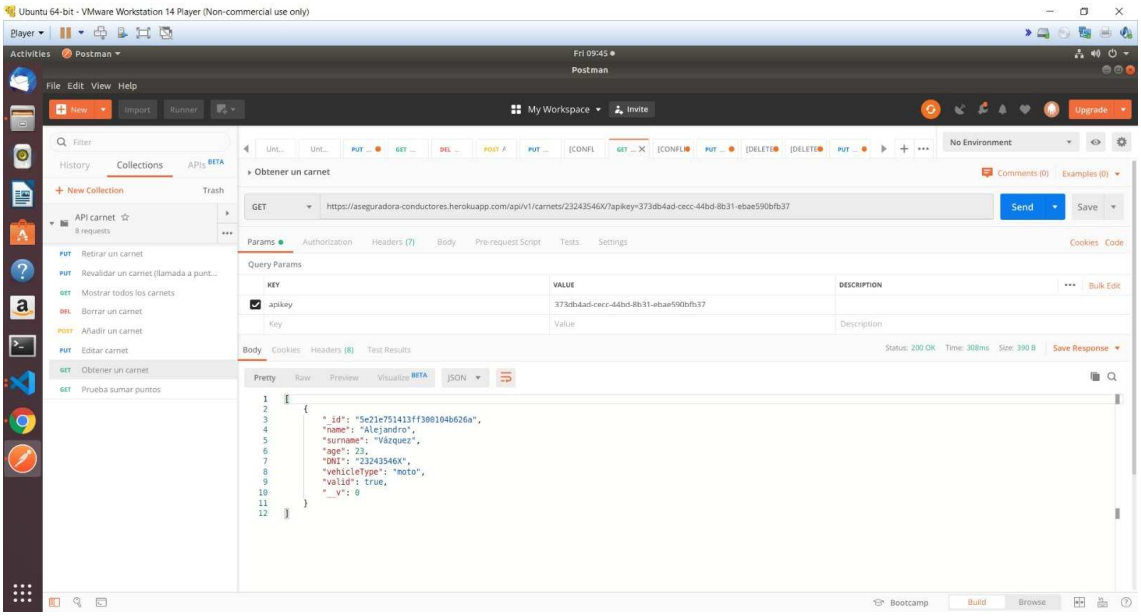
   Como puede observarse en las urls del apartado 1, la api está versionada y en todas partes contiene en su url base la versión en la que se encuentra siendo la base de la api "/api/v1"

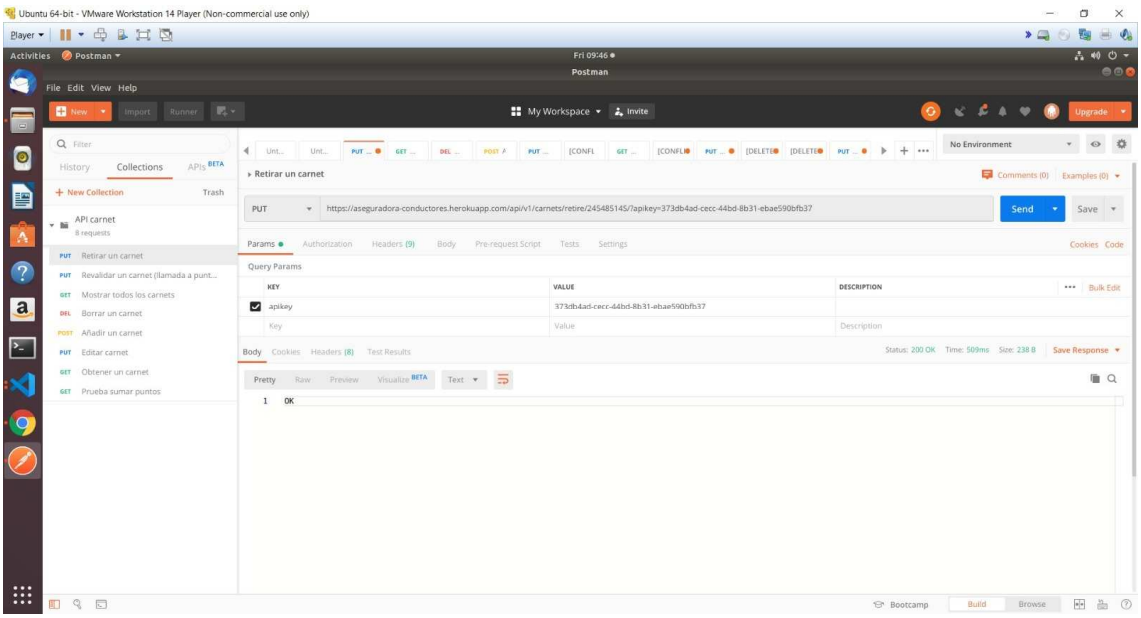
5. **Se debe tener un conjunto de ejemplos de uso del API en Postman de todas las operaciones de la API.**

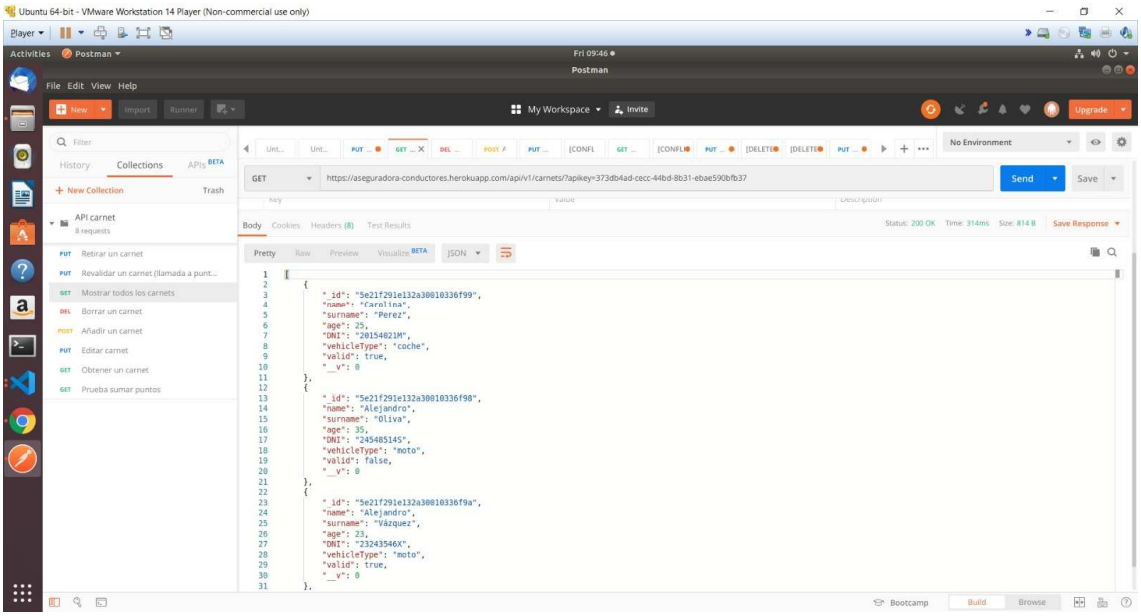
   Postman se ha usado para realizar comprobaciones de la api teniendo todos los métodos guardados como se observa a continuación

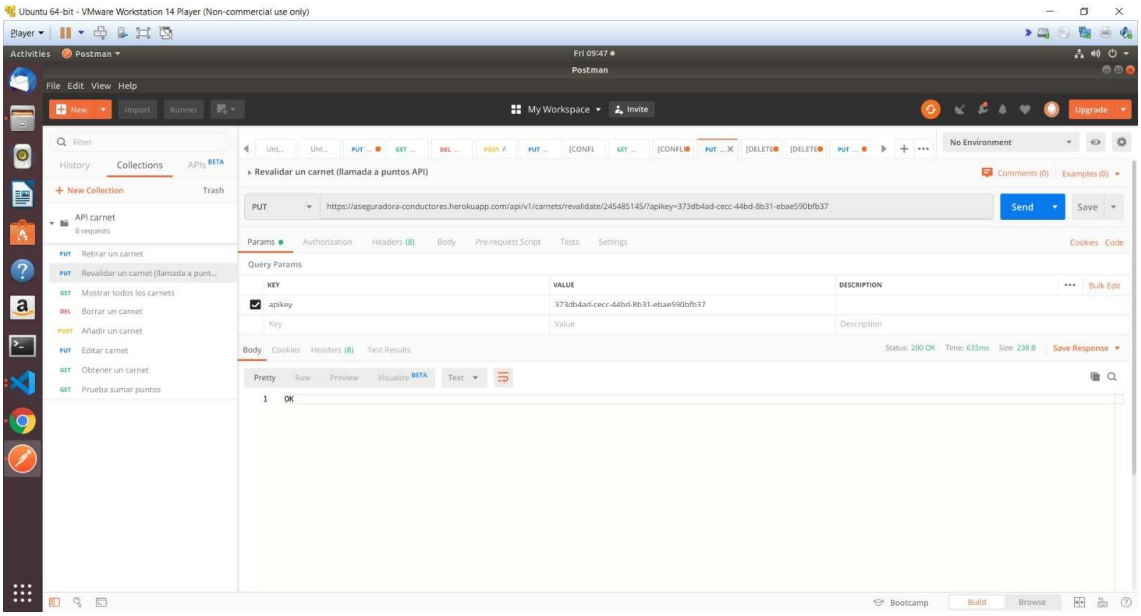


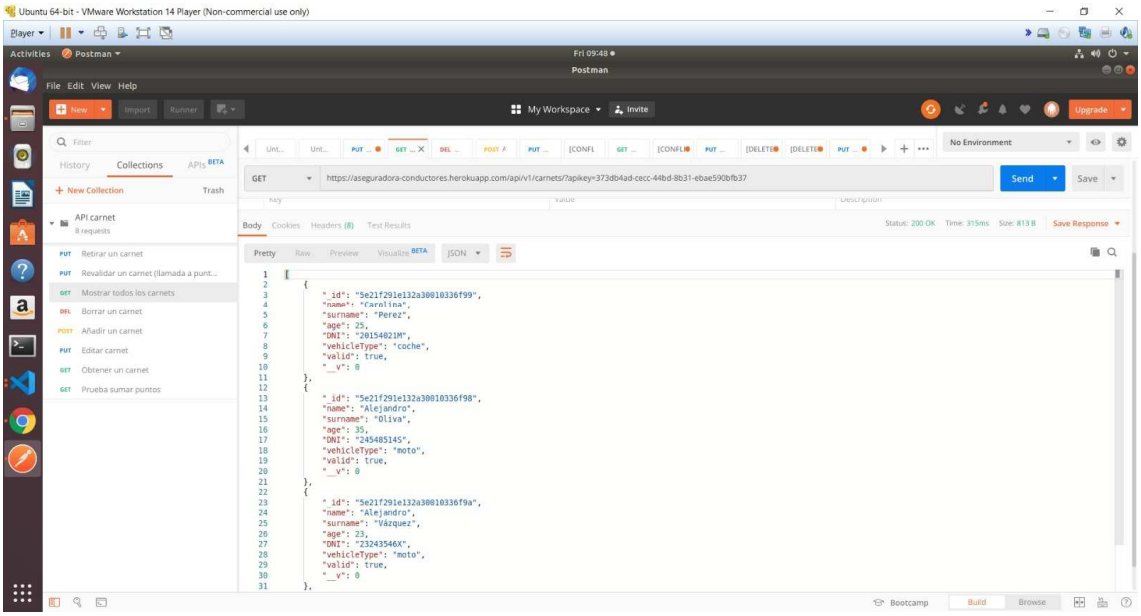

Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
Añadir un carnet
POST
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Created

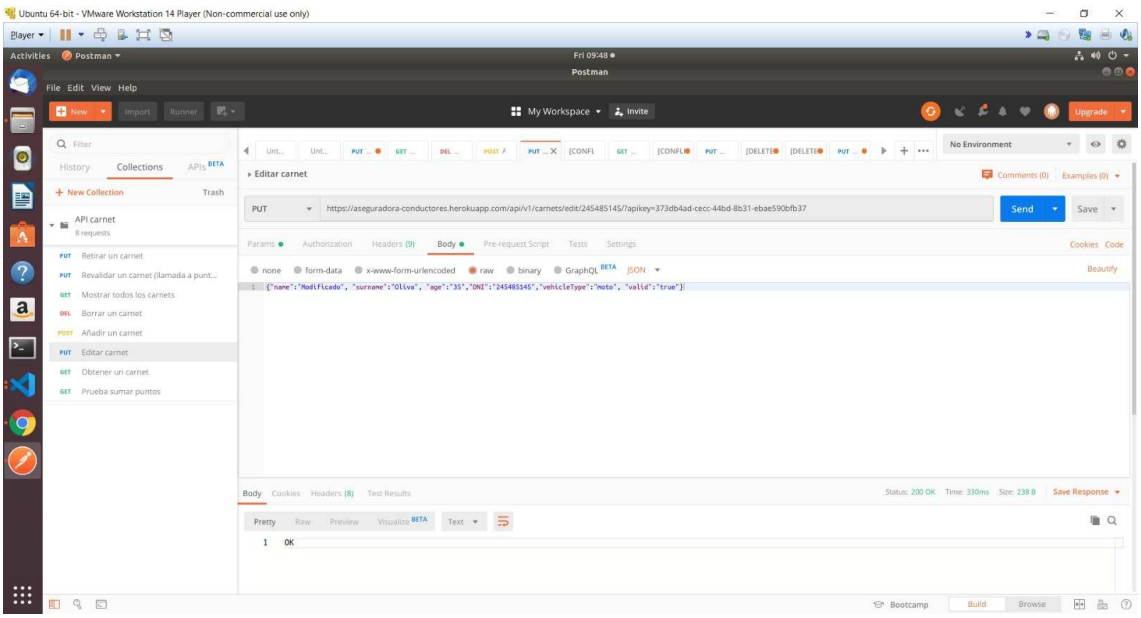

Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
Obtener un carnet
GET
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/23243546X/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
apikey
373db4ad-cecc-44bd-8b31-ebae590bfb37

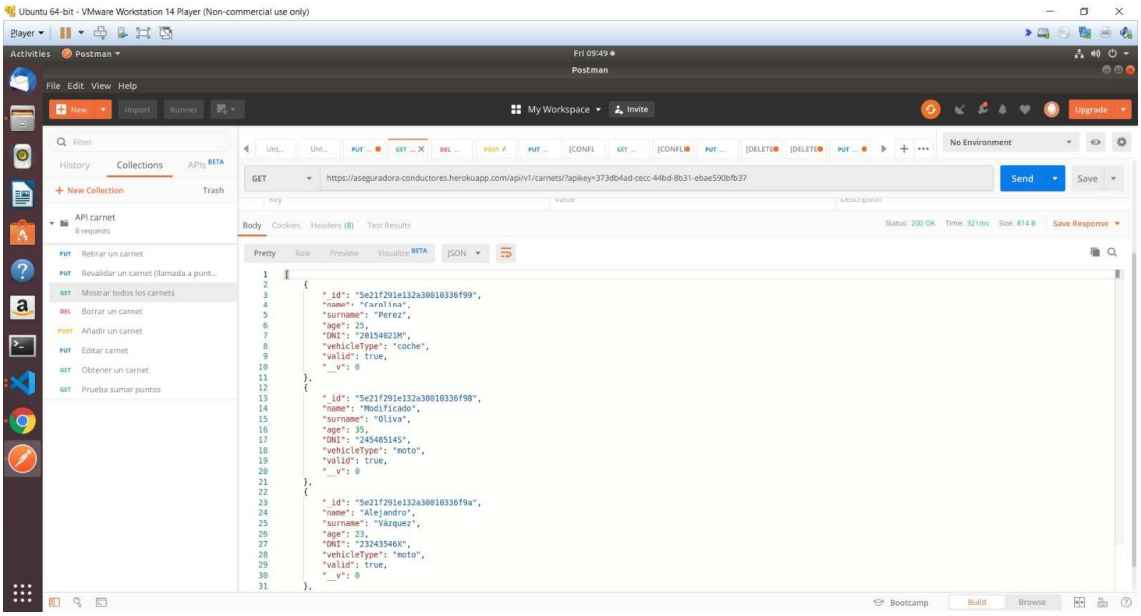

Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
File Edit View Help
Retirar un carnet
PUT
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/retire/245485145/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Status: 200 OK
OK

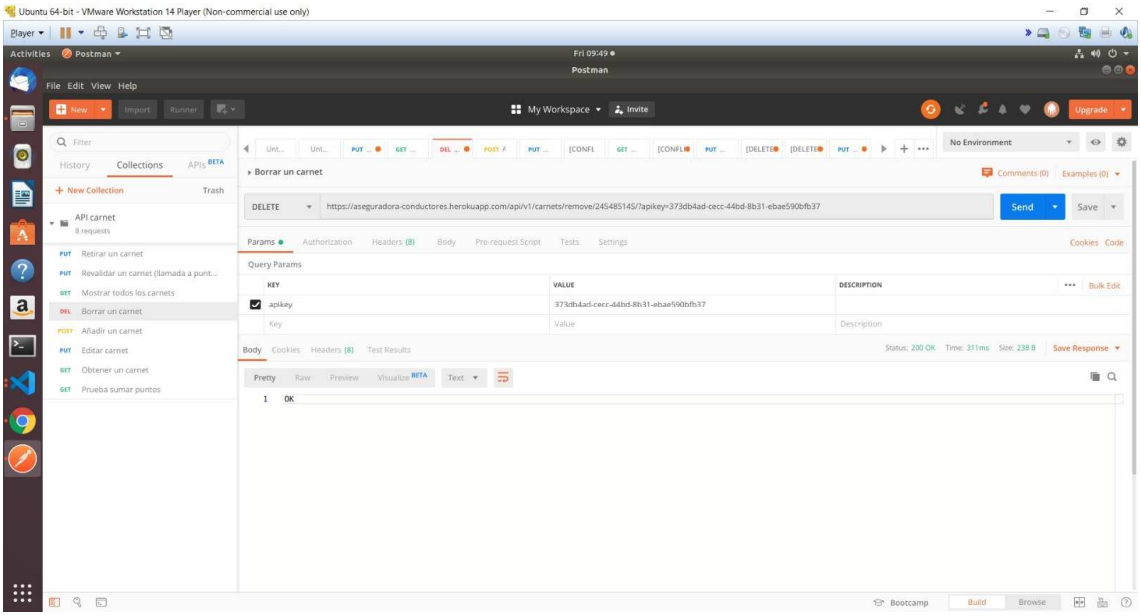

Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
File Edit View Help
GET
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Status: 200 OK
[
{
"_id": "5e21f291e132a30010336f99",
"name": "Carolina",
"surname": "Perez",
"age": 25,
"DNI": "20154021M",
"vehicleType": "coche",
"valid": true,
"__v": 0
},
{
"_id": "5e21f291e132a30010336f98",
"name": "Alejandro",
"surname": "Oliva",
"age": 35,
"DNI": "24548514S",
"vehicleType": "moto",
"valid": false,
"__v": 0
},
{
"_id": "5e21f291e132a30010336f9a",
"name": "Alejandro",
"surname": "Vázquez",
"age": 23,
"DNI": "23243546X",
"vehicleType": "moto",
"valid": true,
"__v": 0
},

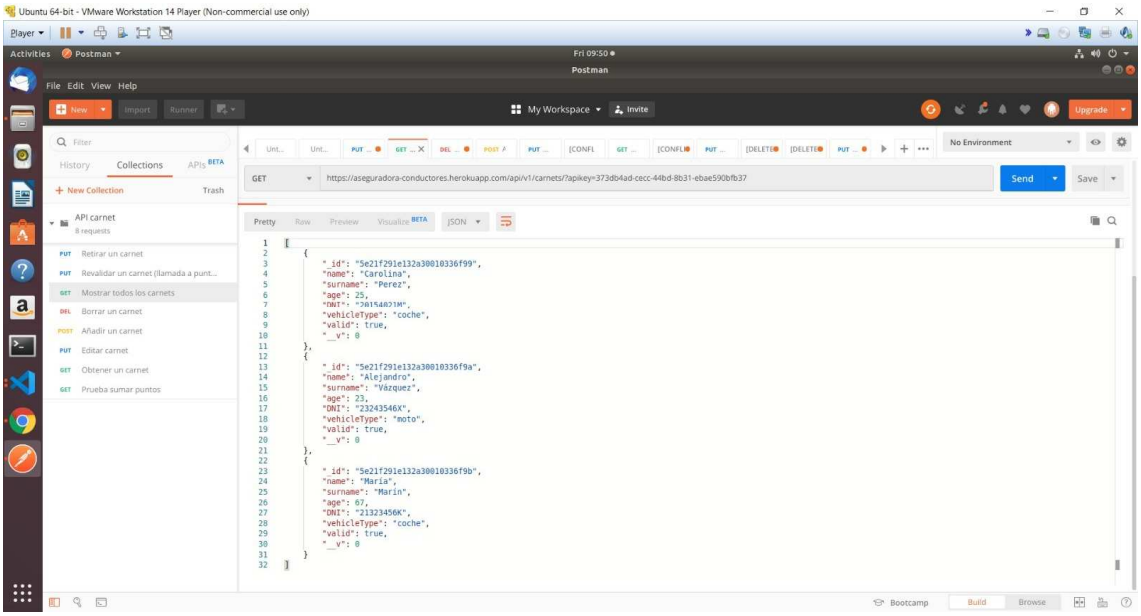

Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
Revalidar un carnet (llamada a puntos API)
PUT
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/revalidate/24548514S?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Query Params
KEY
VALUE
DESCRIPTION
apikey
373db4ad-cecc-44bd-8b31-ebae590bfb37
Body
Pretty
OK


Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
GET
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Body
Pretty
JSON
[
{
"_id": "5e21f291e132a30010336f99",
"name": "Carolina",
"surname": "Perez",
"age": 25,
"DNI": "20154021M",
"vehicleType": "coche",
"valid": true,
"__v": 0
},
{
"_id": "5e21f291e132a30010336f98",
"name": "Alejandro",
"surname": "Oliva",
"age": 35,
"DNI": "24548514S",
"vehicleType": "moto",
"valid": true,
"__v": 0
},
{
"_id": "5e21f291e132a30010336f9a",
"name": "Alejandro",
"surname": "Vázquez",
"age": 23,
"DNI": "23243546X",
"vehicleType": "moto",
"valid": true,
"__v": 0
},


Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
File Edit View Help
My Workspace
Invite
Upgrade
History
Collections
New Collection
Trash
API carnet
8 requests
Retirar un carnet
Revalidar un carnet (llamada a punt...
Mostrar todos los carnets
Borrar un carnet
Añadir un carnet
Editar carnet
Obtener un carnet
Prueba sumar puntos
Editar carnet
PUT
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/edit/24548514S/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Body
raw
{"name":"Modificado", "surname":"Oliva", "age":"35","DNI":"24548514S","vehicleType":"moto", "valid":"true"}
Status: 200 OK
Time: 330ms
Size: 238 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1 OK
Bootcamp
Build
Browse


Ubuntu 64-bit - VMware Workstation 14 Player (Non-commercial use only)
Postman
File Edit View Help
My Workspace
Invite
Upgrade
History
Collections
New Collection
Trash
API carnet
8 requests
Retirar un carnet
Revalidar un carnet (llamada a punt...
Mostrar todos los carnets
Borrar un carnet
Añadir un carnet
Editar carnet
Obtener un carnet
Prueba sumar puntos
GET
https://aseguradora-conductores.herokuapp.com/api/v1/carnets/?apikey=373db4ad-cecc-44bd-8b31-ebae590bfb37
Send
Save
Status: 200 OK
Time: 321ms
Size: 814 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"_id": "5e21f291e132a30010336f99",
"name": "Carolina",
"surname": "Perez",
"age": 25,
"DNI": "20154021M",
"vehicleType": "coche",
"valid": true,
"__v": 0
},
{
"_id": "5e21f291e132a30010336f98",
"name": "Modificado",
"surname": "Oliva",
"age": 35,
"DNI": "24548514S",
"vehicleType": "moto",
"valid": true,
"__v": 0
},
{
"_id": "5e21f291e132a30010336f9a",
"name": "Alejandro",
"surname": "Vázquez",
"age": 23,
"DNI": "23243546X",
"vehicleType": "moto",
"valid": true,
"__v": 0
},
Bootcamp
Build
Browse

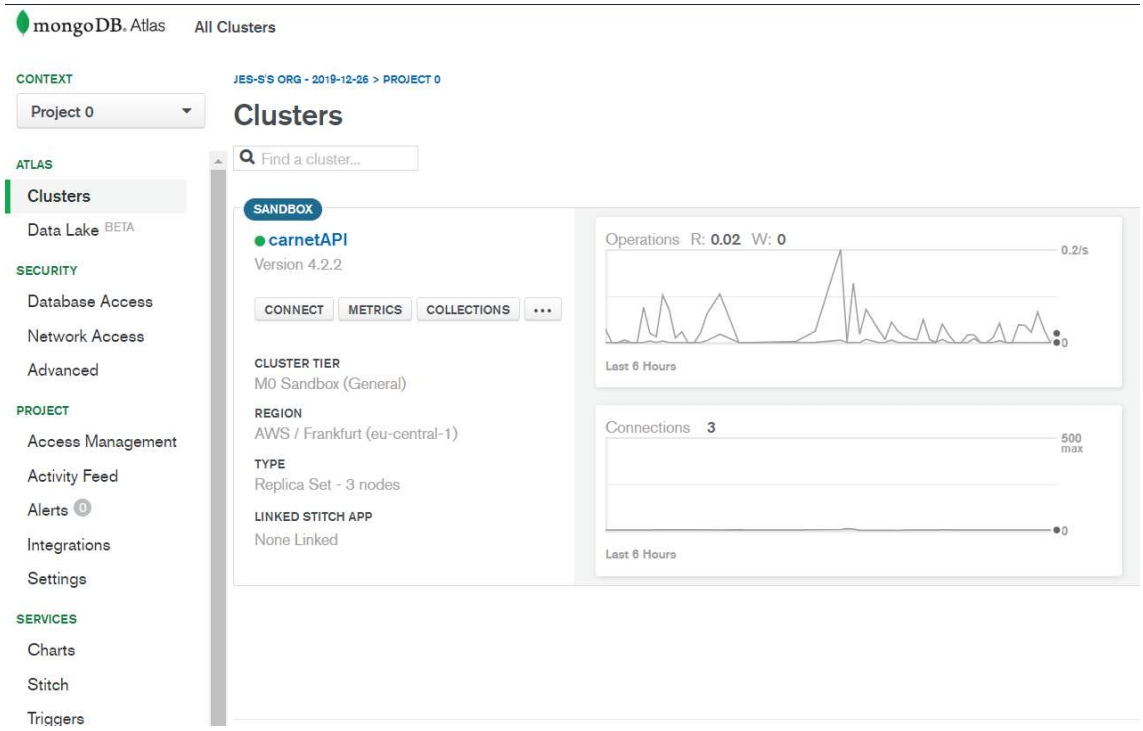




6. Debe tener persistencia utilizando MongoDB

Utilizamos MongoDB como base de datos, aunque en nuestro caso, en vez de tener instalado MongoDB hemos utilizado MongoDB Atlas. Esta es una versión en la nube de la base de datos que nos permite tener todas las ventajas de esta sin necesidad de tener que instalar MongoDB en nuestra máquina.

Todo esto de la conexión con MongoDB se lleva a cabo en el archivo *db.js*.



7. **Utilizar gestión del código fuente y mecanismos de integración continua:**
   a. **El código debe estar subido a un repositorio de Github siguiendo Github Flow**

      El repositorio está subido en GitHub y gestionado con el cli de Git. Se ha seguido el modelo de GitHub Flow creando varias ramas y trabajando sobre ellas en cada uno de los apartados del proyecto.

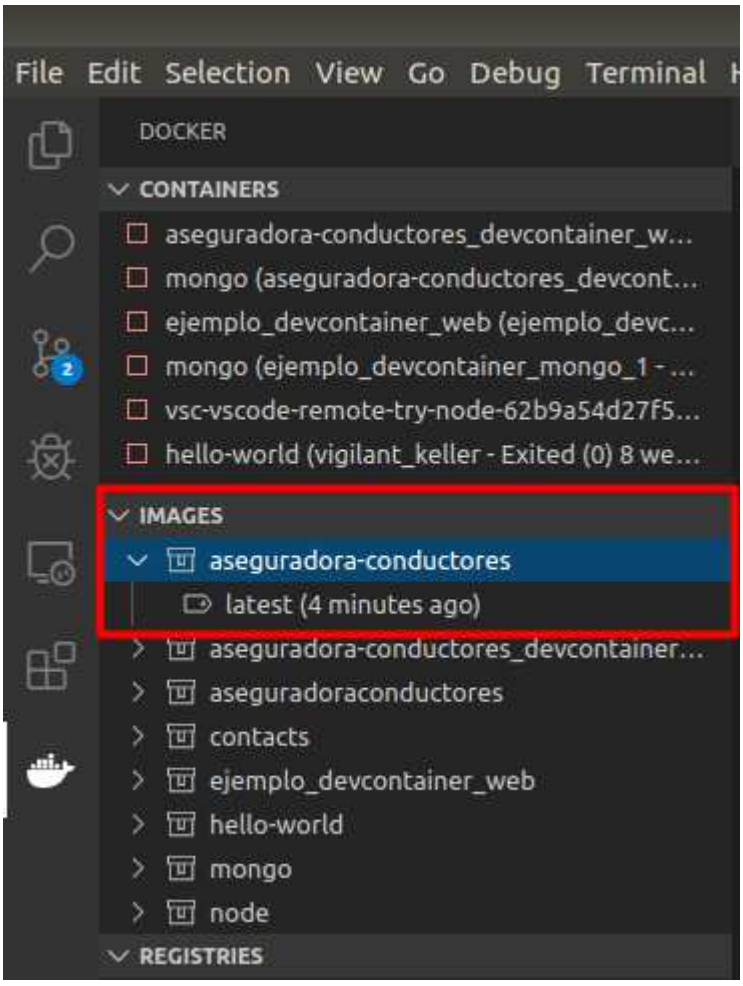
      Dirección del repositorio: https://github.com/DGT-CARNET/Aseguradora-Conductores

   b. **El código debe compilarse y probarse automáticamente usando Travis.ci en cada commit**

      El código se prueba, crea una imagen de Docker y se despliega en heroku (este último paso solo si no ha habido errores previos) cada vez que se hace un commit.

8. Debe haber definida una imagen Docker del proyecto

El proyecto tiene una imagen de Docker definida que se sube junto a .travis.yml cada vez que se hace un commit. Podemos ver que está la imagen "aseguradora-conductores" construida.



9. Debe haber pruebas unitarias implementadas en Javascript para el código del backend utilizando Mocha y Chai o similar

La api contiene unos test básicos realizados en javacript. Dichos test se encuentran en el archivo *server.test.js* incluido en la carpeta *test*. Tanto estos test como para los de integración, se puede comprobar que funcionan correctamente ejecutando el comando *npm test* en el terminal de VScode.

10. Debe haber pruebas de integración con la base de datos.

La api tiene pruebas de integración con la base de datos de MongoDB hechas en el archivo *integración-db.test.js* que se encuentra en la carpeta *test*.

11. Debe tener un mecanismo de autenticación en la API

La api tiene un mecanismo de autenticación mediante passport en el archivo passport.js. Requiere de una apikey para poder acceder a la url tal y como se explicó en clase. Todos los métodos de la api requieren de la apikey para poder funcionar o de lo contrario devolverá *Unauthoraized*.