Documentación DCAAS

Democracy As A Service

Índice

Descripción

DCAAS (Democracy As A Service) es una API para realizar encuestas, se centra sobretodo en: libertad de datos, transparencia, anonimato y seguridad.

Los usuarios pueden registrarse como votantes o publicantes, lo cual hace que puedan votar en encuestas o crearlas, además las encuestas son relativamente personalizables con preguntas de distintos tipos y configuraciones, así como la capacidad de crear informes de los resultados.

Además hay un sistema de roles, entre los que se encuentra el de administrador, el cual puede hacer CRUD sobre casi todas las entidades (los únicos sitios donde no puede hacer estas acciones son en operaciones que vulnerarían la transparencia y justicia de la aplicación, por ejemplo un administrador no puede ver el dueño de una respuesta anónima o cambiar las respuestas/preguntas de una encuesta ya iniciada)

Al ser exclusivamente backend, no tiene nada de interfaz pero si viene con una serie de endpoints API REST para operar la aplicación.

Tecnologías y despliegue

DCAAS está desarrollado en Laravel (lo cual también implica PHP y Composer), principalmente utiliza las librerías propias de Laravel (Illuminate, Symfony, que principalmente provienen de Packagist)

Para el debug se utilizó, entre otras funciones de Laravel y PHP como dd(), dump() o Log::info(), además de una librería llamada Telescope, que permite lanzar un servidor alternativo exclusivamente para ver ciertos parámetros de la app, como peticiones, rutas, etc en modo debug.

Actualmente usa la base de datos SQL de MariaDB, pero está todo diseñado de tal manera que la aplicación es desacoplable, por ejemplo se podría cambiar a PostgreSQL sin cambiar nada de código.

Por otra parte, tanto durante el desarrollo como el despliegue se usa Docker para su funcionamiento. Se usa la función de docker compose para crear dos contenedores, uno con Apache2 que sirva la aplicación de Laravel y otro con la base de datos MariaDB, ambos accesibles tanto desde fuera como entre ellos (actualmente está configurado de tal manera que la app se sirve en el puerto 8081 y la base de datos en el 3310)

Y por último, no solo se usó VSCode para escribir el código, sino que además se usó Insomnia (un cliente de API) para probar los endpoints, tanto durante el desarrollo como durante la aplicación, Insomnia permite exportar listas de endpoints guardados así como automatizar ciertos parámetros con Javascript, dichas exportaciones están disponibles en el repositorio en GitHub

Instrucciones de despliegue

Desplegar este backend es relativamente sencillo, el único requisito es tener Docker instalado. En el repositorio hay un archivo de Bash llamado up.sh que abre el servidor automáticamente, esto lo hace mediante `docker compose up -d`, comando que se puede ejecutar en su lugar en caso de no estar en un entorno Linux

Hay que tener especial atención en que es posible que los archivos creados cualquiera de estos contenedores no sean accesibles fácilmente por problemas de permisos, en ese caso se recomienda hacer `sudo chmod 777 -R .` en la raíz del proyecto

Tal y como está configurado el docker-compose.yml ahora mismo, la API es accesible mediante el puerto 8081

Es posible que haga falta instalar dependencias y relacionados, para eso está el archivo comandos.sh que instruye sobre los comandos útiles relacionados con el proyecto, los comandos necesarios para iniciarlo en producción serían

```
docker exec -it apache-http bash ; cd dcaas-app ; composer install ; cp .env.example .env ; php artisan key:generate ; php artisan migrate ; composer dump-autoload
```

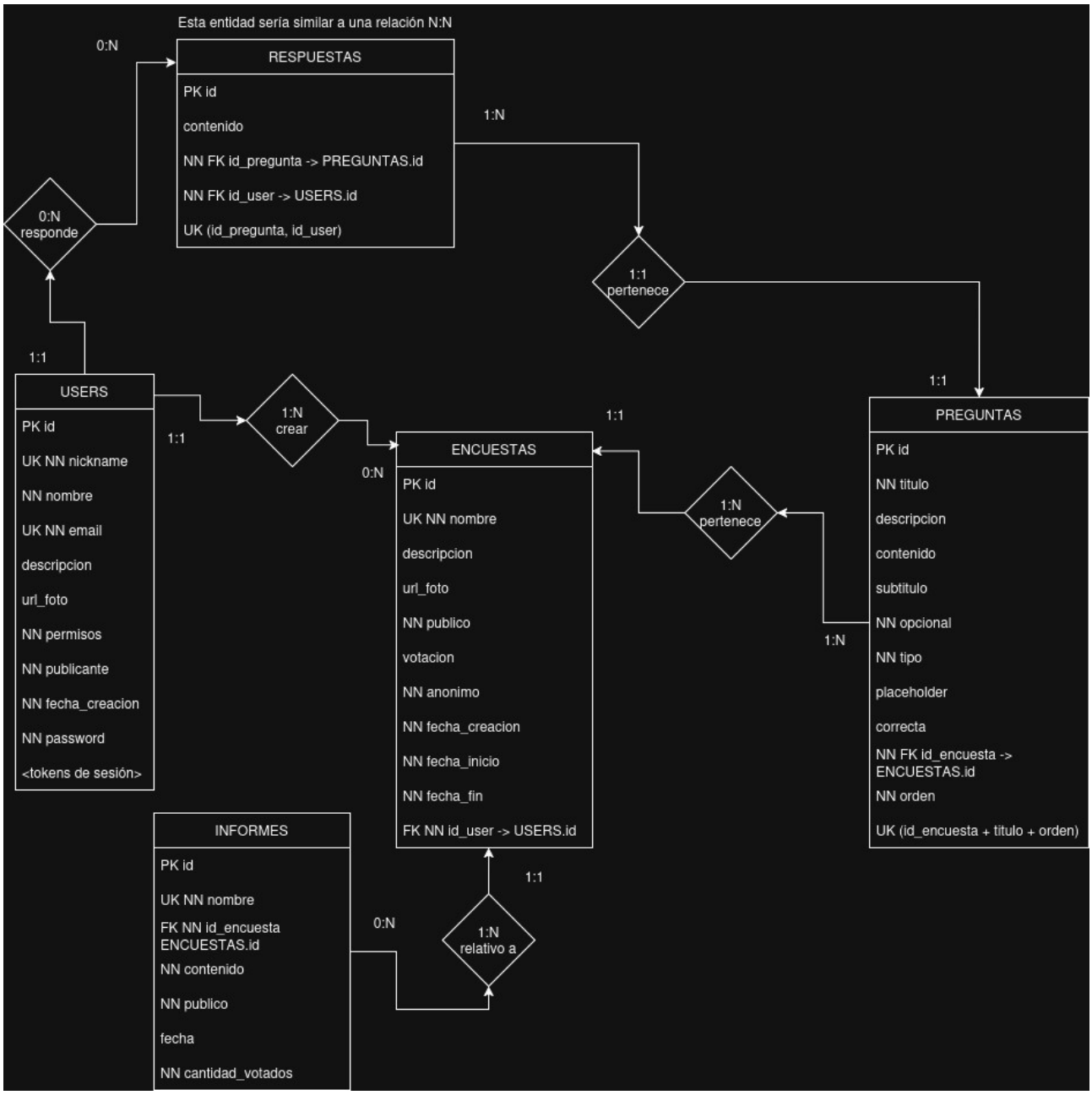
En caso de tener algún error con la base de datos, es importante migrar antes para crear todas las tablas, esto se puede hacer rápidamente con el comando

```
docker exec -it apache-http php dcaas-app/artisan migrate:fresh
```

Diagrama UML de la base de datos

Diagrama UML

Este es el diagrama UML de la base de datos, aunque parezca que no hay relación N:N, la tabla de respuestas estaría actuando como una en cierto modo



Definición de entidades

Users

Representan a los usuarios de la aplicación, que son los que realizan las acciones y lo que se usa para autenticar

id: Clave primaria, es un UUID

nickname: Nombre de slug, es único y representa al usuario, debe cumplir `/^[A-Za-z0-9\-\._]+$/`

nombre: Nombre formal, mínimo 3 caracteres

email: Dirección de correo válida y única, identificativa para el login

descripción: Descripción del usuario de hasta 512 caracteres

url_foto: Una URL válida que representa la foto del usuario

permisos: Entero del 0 al 3 que representa el nivel de permisos, funciona de la siguiente manera:

Normal (0): Usuario normal que puede realizar acciones

Admin (1): Usuario con permisos extendidos que puede hacer CRUD en más entidades

Bloqueado (2): Tiene restringidas ciertas acciones, como la de editar su usuario

Deshabilitado (3): Usuario sin permisos para nada, el equivalente a ser borrado

publicante: Booleano que si es true, es un usuario publicante que puede crear encuestas, y si es false es un usuario que puede votar (no se puede hacer ambas)

fecha_creacion: Almacena el timestamp de cuando se creó el usuario

password: Contraseña del usuario para iniciar sesión o realizar acciones sensibles, debe tener mínimo 8 caracteres, dígitos, mayúsculas, minúsculas y símbolos. Y lo que se almacena es su hash, no la contraseña en sí

Algo clave que identifica que acciones puede realizar el usuario es su “permisos” y su “publicante”, a la hora de ver los endpoints esto se explicará mejor.

Sessions

Las sesiones están manejadas automáticamente por Laravel, pero básicamente un token de sesión (volátil porque solo es válido por 1 día) se obtiene al iniciar sesión o al registrarse, y es necesario para acceder a la mayoría de endpoints (se debe colocar en el Auth Bearer Token en la petición HTTP)

Encuestas

Las encuestas son creadas por los usuarios y son también estos los que las modifican y las responden

id: Clave primaria, es un UUID

nombre: Nombre de la encuesta, único en todo el mundo para evitar plagio, mínimo 3 caracteres

descripcion: Descripción simple de la encuesta, preferentemente en formato Markdown, máximo 512 caracteres

url_foto: URL de la foto de la portada

certificacion: Es un campo opcional para poner una certificación oficial, no es usado para ninguna comprobación pero ayuda a mejorar la confianza de los usuarios ante la encuesta

publico: Booleano que define si la encuesta está visible al público o no, debe estarlo antes de empezar

votacion: Booleano que define si la encuesta está registrada como votación o no, esto no se usa para comprobar nada pero puede ser útil para ciertos formateos o algoritmos de búsqueda

anonimo: Booleano que, si es true, nadie (ni si quiera los admins) podrán ver quien ha votado el qué (aunque igualmente se almacena en la base de datos para comprobar que un usuario no haya votado ya antes)

fecha_creacion: Timestamp con la fecha en la que se creó la encuesta

fecha_inicio: Fecha en la que inició la encuesta, si está vacío es que aún no ha empezado

fecha_fin: Fecha en la que terminó la encuesta, si está vacío es que aún no ha terminado

estado: Entero que indica el estado de la encuesta (de 0 a 2)

- Sin iniciar (0): No se puede votar, puede ser privada, se puede editar
- Activa (1): Está iniciada, se puede votar, es pública, no se puede editar
- Terminada (2): Ya acabó, no se puede ni editar ni votar, se pueden hacer informes

id_user: UUID del usuario que creó la encuesta, el cual debe de ser publicante (es una relación 1:N)

Las sinergias entre estado y publico serán relevantes más adelante, cuando se expliquen los endpoints.

Cuando se borra el usuario, sus encuestas también por el borrado en cascada

Preguntas

Las preguntas son entidades asociadas a las encuestas, se añaden de forma separada a la información de la encuesta. Se hace de esta manera porque cada pregunta tiene su propia información y una encuesta puede tener x preguntas (hasta 127, pero mínimo una antes de empezarla).

id: Clave primaria, es un UUID

titulo: Título de la pregunta, normalmente haciendo la pregunta en sí, de 1 a 127 caracteres

descripcion: Descripción para extender su información, hasta 255 caracteres

subtitulo: Texto alternativo de la pregunta, hasta 255 caracteres

opcional: Booleano que indica si la pregunta puede ser omitida a la hora de responder a la encuesta

tipo: Entero que indica el tipo de la pregunta (del 0 al 3)

Desarrollar (0): Es una pregunta en la que el usuario tiene que escribir texto

Check (1): El usuario tiene que elegir una opción de entre varias

Radio (2): El usuario tiene que elegir 0 o múltiples opciones de entre varias

Numero (3): El usuario tiene que introducir un número, se admiten negativos y decimales

contenido: Indica en sí el contenido de la pregunta, dependiendo del tipo esto puede ser

Desarrollar (0) o Numero (3): Debe estar vacío

Check (1) o Radio (2): Debe de ser un array con las opciones posibles, aunque en la base de datos se guarda usando el separador "¬"

placeholder: Contenido por defecto que aparecería en la interfaz, depende del tipo de la pregunta

Desarrollar (0): Un texto placeholder simplemente
Check (1): Un número, que indica que respuesta estaría marcada por defecto

Radio (2): Un array de números que indican que respuestas están marcadas por defecto (se usa el separador "¬")

Numero (3): El usuario tiene que introducir un número, se admiten negativos y decimales

correcta: Representa que respuesta(s) sería(n) la(s) correcta(s) (depende del tipo), sigue un formato muy similar al de placeholder

orden: Número entero mayor que 0 que indica el orden de la pregunta dentro de esa encuesta, debe ser único por encuesta

id_encuesta: UUID de la encuesta a la que pertenece la pregunta, es una relación 1:N

Se define como campo único unificado al conjunto de id_encuesta, titulo y orden para que no se repitan preguntas dentro de una encuesta.

Por otra parte el campo contenido es demasiado complejo como para describirlo aquí, ya que depende del tipo y sigue unas reglas de validación muy estrictas

Cuando se borra una encuesta, también se borran sus preguntas por el borrado en cascada

Respuestas

La respuesta en cierto modo actuaría como tabla para una relación N:N, ya que es respondida por un usuario ante una pregunta, un usuario puede responder varias preguntas y la misma pregunta puede ser respondida por distintos usuarios (aunque no es tan simple, por ejemplo la misma pregunta solo puede ser respondida una vez por el mismo usuario, así como saltársela si fuese opcional)

Todas las preguntas de una encuesta se deben responder de golpe, por lo que al responder una encuesta se crearán tantos objetos de respuesta como preguntas tuviese la encuesta (incluso si la pregunta se ha dejado en blanco, su contenido sería "")

Además los usuarios no pueden responder si la encuesta asociada a la pregunta no está activa

id: Clave primaria, es un UUID

contenido: Es el contenido de la respuesta, puede ser dejada en blanco si la pregunta era opcional y su contenido depende del tipo de la pregunta

> Desarrollar (0): El texto que habría escrito el usuario
>
> Check (1): El número de la opción que ha marcado el usuario
>
> Radio (2): Un array (separador "¬") con elementos irrepetibles que serían los números de las opciones marcadas
>
> Numero (3): El número que ha escrito el usuario

id_pregunta: UUID de la pregunta a la que hace referencia este objeto de pregunta, gracias a esto se puede saber la encuesta, es una relación a preguntas

id_user: UUID del usuario que ha respondido la pregunta, es una relación a users

Si la encuesta era anónima, se guarda igualmente el campo id_user ya que es necesario para saber si el usuario ha respondido ya, sin embargo se garantiza que en ese caso absolutamente nadie pueda saber que ha respondido el usuario (para mayor seguridad se podría haber encriptado este campo, pero esto complicaría las relaciones entre tablas, y no está pensado que un administrador pueda manipular la base de datos con sentencias SQL)

Cuando se borra una pregunta, también se borran sus respuestas por el borrado en cascada

Eso sí, si se borra un usuario el campo id_user pasará a ser null pero la respuesta seguirá almacenada, ya que el borrado de usuarios no provoca borrado en cascada a la respuesta si es este usuario quien la respondió

Informes

Cuando una encuesta finaliza, se puede pedir crear un informe de dicha encuesta en base a las preguntas y respuestas, esto generará distintos datos estadísticos. Eso sí, solo se pueden generar hasta 5 informes por encuesta, aunque una vez revisados se pueden publicar para que todos lo vean (respetando el anonimato y la privacidad)

id: Clave primaria, es un UUID

nombre: Nombre del informe, debe de ser único para evitar plagios, hasta 255 caracteres

id_encuesta: UUID de la encuesta a la que hace referencia, es una relación a encuesta

contenido: Distintos datos estadísticos de la encuesta, como extensión opcional se podría haber hecho que el creador de la encuesta pueda introducir opciones para hacer las estadísticas a su gusto, es por eso que se pueden hacer múltiples informes sobre la misma encuesta. Esto ahora mismo no está implementado y de momento todos los informes de la misma encuesta se verán iguales, ya que está preparado para que se puedan configurar. Como tal este campo almacena un JSONB

publico: Booleano que indica si el informe es visible al público, se puede cambiar a true

fecha: Timestamp de la fecha en la que se generó el informe

cantidad_votados: También se almacena un entero con la cantidad de usuarios que votaron en la encuesta, este valor es necesario para hacer medias y otras operaciones

Además, un informe solo se puede generar cuando la encuesta ya está terminada

(Cada endpoint llama a una función en un controlador)

Documentación de la API

Actualmente solo se puede comunicar con el backend mediante API REST, usando HTTP. Todas las rutas que funcionen de esta manera empezarán con `/api/v1`, osea que una URL podría ser algo como http://localhost:8081/api/v1/test. A partir de este prefijo salen los siguientes endpoints

```
GET|HEAD  /
GET|HEAD  api/v1/admin
DELETE    api/v1/admin/encuesta/{id}/borrar ..... borrarEncuestaAjena › AdminController@borrarEncuestaAjena
PATCH     api/v1/admin/encuesta/{id}/editar ..... editarEncuestaAjena › AdminController@editarEncuestaAjena
GET|HEAD  api/v1/admin/encuesta/{id}/ver ..... verEncuestaAjena › AdminController@verEncuestaAjena
DELETE    api/v1/admin/informes/{id}/borrar ..... borrarInformeAjeno › AdminController@borrarInformeAjeno
GET|HEAD  api/v1/admin/informes/{id}/ver ..... verInformeAjeno › AdminController@verInformeAjeno
GET|HEAD  api/v1/admin/preguntas/{id}/ver/{pagina?} ..... verPreguntasEncuestaAjena › AdminController@verPreguntasEncuestaAjena
GET|HEAD  api/v1/admin/respuestas/{id}/ver/{pagina?} ..... verRespuestaDeEncuestaAjena › AdminController@verRespuestaDeEncuestaAjena
PATCH     api/v1/admin/usuario/alterPerms ..... editarPermisos › AdminController@editarPermisos
DELETE    api/v1/admin/usuario/{id}/borrar ..... borrarUsuarioAjeno › AdminController@borrarUsuarioAjeno
PATCH     api/v1/admin/usuario/{id}/editar ..... editarUsuarioAjeno › AdminController@editarUsuarioAjeno
GET|HEAD  api/v1/admin/usuario/{id}/ver ..... verUsuarioAjeno › AdminController@verUsuarioAjeno
GET|HEAD  api/v1/debug ..... debug
POST      api/v1/encuesta ..... crearEncuesta › EncuestaController@crear
DELETE    api/v1/encuesta/borrar/{id} ..... borrarEncuesta › EncuestaController@borrar
GET|HEAD  api/v1/encuesta/buscar/{busqueda}/{pagina?} ..... buscarEncuestaPublica › EncuestaController@buscar
PATCH     api/v1/encuesta/editar/{id} ..... editarEncuesta › EncuestaController@editar
POST      api/v1/encuesta/finalizar/{id} ..... finalizarEncuesta › EncuestaController@finalizar
POST      api/v1/encuesta/iniciar/{id} ..... iniciarEncuesta › EncuestaController@iniciar
GET|HEAD  api/v1/encuesta/ver/{id} ..... verEncuestaPublica › EncuestaController@verPublico
GET|HEAD  api/v1/encuesta/verCantidadVotos/{id} ..... verCantidadVotados › RespuestaController@verCantidadVotados
GET|HEAD  api/v1/encuesta/verMia/{id} ..... verEncuestaPrivada › EncuestaController@verPrivado
GET|HEAD  api/v1/encuesta/votado/{id} ..... verVotado › RespuestaController@verVotado
POST      api/v1/encuesta/votar/{id} ..... votar › RespuestaController@votar
DELETE    api/v1/informe/borrar/{id} ..... borrarInforme › InformeController@borrarInforme
POST      api/v1/informe/crear/{id} ..... crearInforme › InformeController@crearInforme
PATCH     api/v1/informe/publicar/{id} ..... publicarInforme › InformeController@publicarInforme
GET|HEAD  api/v1/informe/ver/{id} ..... verInforme › InformeController@verInforme
GET|HEAD  api/v1/informe/verEncuesta/{id} ..... verInformesDeEncuesta › InformeController@verInformesDeEncuesta
GET|HEAD  api/v1/informe/verMiEncuesta/{id} ..... verInformesMiEncuesta › InformeController@verInformesMiEncuesta
GET|HEAD  api/v1/informe/verMio/{id} ..... verMiInforme › InformeController@verMiInforme
PUT       api/v1/preguntas/establecer/{id} ..... establecer › PreguntaController@establecer
GET|HEAD  api/v1/preguntas/ver/{busqueda}/{pagina?} ..... verDeEncuesta › PreguntaController@verDeEncuesta
GET|HEAD  api/v1/preguntas/verMia/{id}/{pagina?} ..... verDeEncuestaPrivado › PreguntaController@verDeEncuestaPrivado
GET|HEAD  api/v1/preguntas/verUsuario/{id}/{pagina?} ..... verEncuestasDeUsuario › EncuestaController@verEncuestasDeUsuario
GET|HEAD  api/v1/respuesta/ver/{id} ..... verRespuestaDeEncuesta › RespuestaController@verRespuestaDeEncuesta
GET|HEAD  api/v1/respuestas/ver/{id}/{pagina?} ..... verRespuestasDeEncuesta › RespuestaController@verRespuestasDeEncuesta
GET|HEAD  api/v1/test ..... test
POST      api/v1/usuario ..... registrarse › UserController@registrar
DELETE    api/v1/usuario ..... borrarUsuario › UserController@borrar
PATCH     api/v1/usuario ..... editarUsuario › UserController@editar
DELETE    api/v1/usuario/cerrarSesion ..... cerrarSesion › UserController@cerrarSesion
POST      api/v1/usuario/login ..... login › UserController@login
GET|HEAD  api/v1/usuario/ver/{id} ..... verUsuarioPublico › UserController@ver
GET|HEAD  api/v1/usuario/yo ..... verUsuarioPrivado › UserController@verYo
GET|HEAD  api/v1/validarSesion ..... validarSesion › UserController@verYo
GET|HEAD  sanctum/csrf-cookie ..... sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieController@show
GET|HEAD  storage/{path} ..... storage.local
```

En algunas ocasiones se hará mención a <id>, esto representa el UUID del objeto con el cual hacer dicha acción. También se mencionará <pagina>, cuando un endpoint devuelve múltiples objetos, se usa una técnica de paginado de 50 en 50, por ejemplo si página vale 2 se devolverán las entidades desde la 51º hasta la 100º, si no se pone el valor por defecto es 1 (del 1º al 50º)
Además en cada ruta se enseñarán ejemplos del formato de la petición y la respuesta, con datos de ejemplo

Por último, para no alargar demasiado la documentación se omitirán las respuestas de error por objeto no encontrado, acción imposible, datos inválidos o error de permisos (se explicarán los requisitos del endpoint, pero seria redundante mostrar cada vez todos los posibles errores en cada endpoint)

(Si se omite la respuesta o la pregunta en una petición es porque no es relevante)
(Si se omite información sobre la validación de campos, es porque se usa la misma que la explicada en las entidades)

Peticiones de la API

Todos los endpoints de la API siguen un formato de respuesta concreto, este sería un mensaje de error


```
{
	"status": "failure",
	"timestamp": "2026-02-24T14:23:15.611090Z",
	"message": "Usuario no encontrado",
	"ok": false,
	"data": null,
	"code": 404
}
```


Este mensaje de error devuelve un código de error HTTP concreto, ok:false, un mensaje informativo, el estado de la petición y un timestamp indicando cuando respondió.

Por otro lado así se podría ver una petición exitosa


```
{
	"status": "success",
	"message": "Usuario encontrado correctamente",
	"data": {
		"usuario": {
			"nickname": "Usuario-1",
			"nombre": "usuario 1 hola",
			"descripcion": null,
			"url_foto": null,
			"id": "019c900c-12cd-735c-8d92-de34388dc9dd",
			"permisos": 0,
			"fecha_creacion": "2026-02-24 14:26:54"
		}
	},
	"timestamp": "2026-02-24T14:27:28.561656Z",
	"ok": true,
	"code": 200
}
```


Indica un mensaje, un status:”success”, ok:true, el código de éxito 200 HTTP, un timestamp indicando cuando se respondió y lo más importante, el campo data que incluye los datos devueltos por el servidor

Por otra parte, a la hora de mandar peticiones a esta API es recomendado usar el header

```
Content-Type: "application/json"
```

Rutas públicas

Estas son las rutas que no requieren nada en el Auth Bearer Token, ya que no requieren estar autenticado con ningún usuario. Se centran sobretodo en leer datos públicos

GET /test: Una ruta de test que solo funciona cuando la app está en modo debug

GET /debug: Igual que test

GET /usuario/ver/<id>: Ver los datos públicos de un usuario, buscando a partir de su id o su nickname

Respuesta:


```
{
	"usuario": {
		"nickname": "Usuario-1",
		"nombre": "usuario 1 usuariez",
		"descripcion": null,
		"url_foto": null,
		"id": "019c900c-12cd-735c-8d92-de34388dc9dd",
		"permisos": 0,
		"fecha_creacion": "2026-02-24 14:26:54"
	}
}
```


POST /usuario: Registrarse como usuario, requiere los datos del registro

Petición:


```
{
	"nickname": "Usuario-1",
	"nombre": "usuario 1 usuariez",
	"email": "usuario1@yangmail.com",
	"password": "123456789aA.",
	"publicante": true
}
```


Respuesta:


```
{
	"usuario": {
		"nickname": "Usuario-1",
		"nombre": "usuario 1 usuariez",
		"descripcion": null,
		"url_foto": null,
		"id": "019c9054-8f9c-7159-bac7-59d17a0aaea1",
		"permisos": 0,
		"fecha_creacion": null,
		"email": "usuario1@yangmail.com"
	},
	"access_token": "1|zDLaVYHWnb09fxClyshADOzW1wWfrYi5gkMzn4nt3f3407a7",
	"token_type": "Bearer"
}
```

POST /usuario/login: Inicia sesión, devolviendo el token de sesión

Petición:


```
{
	"auth": "Usuario-1",
	"password": "123456789aA."
}
```


auth admite tanto el nickanme como el correo del usuario

Respuesta:


```
{
	"usuario": {
		"nickname": "Usuario-1",
		"nombre": "usuario 1 usuariez",
		"descripcion": null,
		"url_foto": null,
		"id": "019c9054-8f9c-7159-bac7-59d17a0aaea1",
		"permisos": 0,
		"fecha_creacion": "2026-02-24 15:45:47",
		"email": "usuario1@yangmail.com"
	},
	"access_token": "2|KhlsBBFK55nVtbAh1TqEFEmQ1jZCc8yKQbjpUpbHe29921df",
	"token_type": "Bearer"
}
```


GET /encuesta/ver/<id>: Ver los datos públicos de una encuesta pública a partir de su id

Respuesta:


```
{
	"encuesta": {
		"nombre": "preguntas12 oficial",
		"descripcion": "unas cuantas preguntas",
		"url_foto": "http://yangmei.io/a.png",
		"id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
		"fecha_creacion": "2026-02-24 15:45:47",
		"certificacion": "1234a",
		"votacion": false,
		"anonimo": true,
		"fecha_inicio": "N/A",
		"fecha_fin": "N/A",
		"estado": 0,
		"id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1"
	}
}
```

GET /encuesta/buscar/<busqueda>/<pagina?>: Busca múltiples encuestas por su nombre, eligiendo una página concreta (se dividen de 50 en 50)

Respuesta:


```
{
	"encuestas": [
		{
			"nombre": "preguntas12 oficial",
			"descripcion": "unas cuantas preguntas",
			"url_foto": "http://yangmei.io/a.png",
			"id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
			"fecha_creacion": "2026-02-24 15:45:47",
			"certificacion": "1234a",
			"votacion": false,
			"anonimo": true,
			"fecha_inicio": "N/A",
			"fecha_fin": "N/A",
			"estado": 0,
			"id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1"
		}
	]
}
```


GET /preguntas/ver/<id>/<pagina?>: Ver las preguntas en cierta página (de 50 en 50) de cierta encuesta pública a partir de su id

Respuesta:


```
{
	"preguntas": [
		{
			"id": "2ed4545a-01af-43c8-9c55-a55b81807ab1",
			"titulo": "escribe texto",
			"descripcion": "descripcion hola hola",
			"contenido": null,
			"opcional": true,
			"tipo": 0,
			"id_encuesta": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
			"placeholder": "abc",
			"subtitulo": "abajo"
		}, ......
	]
}
```

GET /encuestas/verUsuario/<id>/<pagina?>: Ver las encuestas de un usuario publicante a partir de su id en cierta página

Respuesta:


```
{
	"encuestas": [
		{
			"nombre": "preguntas12 oficial",
			"descripcion": "unas cuantas preguntas",
			"url_foto": "http://yangmei.io/a.png",
			"id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
			"fecha_creacion": "2026-02-24 15:45:47",
			"certificacion": "1234a",
			"votacion": false,
			"anonimo": true,
			"fecha_inicio": "N/A",
			"fecha_fin": "N/A",
			"estado": 0,
			"id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1"
		}
	]
}
```


GET /encuesta/verCantidadVotos/<id>: Ver la cantidad de usuarios que han participado en una encuesta a partir de su id

Respuesta:


```
{
	"cantidad": 8
}
```


GET /informe/ver/<id>: Ver un informe público a partir de su id

Respuesta:

<Mirar archivo adjunto EJEMPLO_INFORME.json, se incluye aparte porque era demasiado largo>

Así es como se ve un informe de ejemplo con la configuración actual

GET /informes/verEncuesta/<id>: Ver los informes púbicos de una encuesta a partir de su id

Respuesta:


```
{
	"informes": [
		{
			"id": "44d620ce-3bb4-4070-b3f5-6cc16f7ddc3c"
		}, ........
	]
}
```

Rutas privadas

Todas estas rutas requieren un token en Auth Bearer Token en la petición HTTP, el cual debe hacer referencia a una sesión válida de un usuario. Por lo tanto todas estas rutas pueden conocer que usuario las está usando

GET /usuario/yo: Devuelve todos los datos del usuario actual

Respuesta:


```
{
	"usuario": {
		"nickname": "Usuario-1",
		"nombre": "usuario 1 usuariez",
		"descripcion": null,
		"url_foto": null,
		"id": "019c9054-8f9c-7159-bac7-59d17a0aaea1",
		"permisos": 0,
		"fecha_creacion": "2026-02-24 15:45:47",
		"email": "usuario1@yangmail.com"
	}
}
```


GET /validarSesion: Si esta ruta no devuelve error significa que el token de sesión ofrecido sigue siendo válido

DELETE /usuario: Borrar el usuario actual, provocando un gran borrado en cascada

Petición:


```
{
	"password": "123456789aA."
}
```


Por seguridad, en esta ruta se pide la contraseña

DELETE: /usuario/cerrarSesión: Invalida la sesión y su token (todos los que estén asociados a ese usuario)

Rutas de editor

Estas son las rutas que obligatoriamente requieren que los permisos del usuario sean o admin o normal

PATCH /usuario: Editar los datos del usuario actual

Petición:

```
{
        "nickname": "Usuario-nuevo",
        "nombre": "Mi nuevo nombre",
        "email": "minuevocorreo@yangmail.com",
        "password": "123456789aA..",
        "descripcion": "holaholahola",
        "url_foto": "http://delivery.io/aaa.png",
        "oldPassword": "123456789aA."
}
```

Todos los campos son opcionales, y si no se ponen se mantendrán los datos antiguos, aunque para cambiar el correo o la contraseña hace falta oldPassword, que tiene que ser la contraseña anterior

Rutas de votante

Estas rutas requieren que el usuario actual esté registrado como votante

POST /encuesta/votar/<id>: Votar en una encuesta concreta, estableciendo de golpe todas las respuestas

Petición:

```
{
        "respuestas": [
                "asldfksjadf",
                [1,2],
                2,
                4.4,
                ""
        ]
}
```

No solo es necesario resonder a todas las preguntas, sino que se hace en el mismo orden en el que aparecen. Para responder en blanco se puede poner “” ya que cada elemento del array debe hacer referencia a una pregunta. Esta petición de ejemplo muestra todas las posibilidades correctas, estaría respondiendo en orden a: una pregunta de texto, una de multiple opción, una de única opción, una númerica y una vacía en una opcional

GET /encuesta/votado/<id>: Comprobar si en cierta encuesta el usuario actual ha votado o no

Respuesta:

```
{
        "votado": true
}
```

Rutas de publicante

Estas rutas requieren que el usuario actual esté registrado como publicante (creador de encuestas)

POST /encuesta: Crear una nueva encuesta (solo los datos, las preguntas se crean después)

Petición:


```
{
        "anonimo": true,
        "publico": true,
        "votacion": false,
        "nombre": "preguntas12 oficial",
        "descripcion": "unas cuantas preguntas",
        "certificacion": "1234a",
        "url_foto": "http://yangmei.io/a.png"
}
```


Respuesta:


```
{
        "encuesta": {
                "nombre": "preguntas12 oficial",
                "descripcion": "unas cuantas preguntas",
                "url_foto": "http://yangmei.io/a.png",
                "id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
                "fecha_creacion": "2026-02-24 15:45:47",
                "certificacion": "1234a",
                "publico": true,
                "votacion": false,
                "anonimo": true,
                "fecha_inicio": "N/A",
                "fecha_fin": "N/A",
                "estado": 0,
                "id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1"
        }
}
```


GET /encuesta/verMia/<id>: Ver todos los datos de una encuesta creada por el usuario actual, útil para encuestas que aún son privadas

Respuesta:


```
{
        "encuesta": {
                "nombre": "preguntas12 oficial",
                "descripcion": "unas cuantas preguntas",
                "url_foto": "http://yangmei.io/a.png",
                "id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
                "fecha_creacion": "2026-02-24 15:45:47",
                "certificacion": "1234a",
                "publico": false,
                "votacion": false,
                "anonimo": true,
                "fecha_inicio": "N/A",
                "fecha_fin": "N/A",
                "estado": 0,
                "id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1"
        }
}
```

GET /encuestas/verMias: Devuelve las encuestas del usuario actual

Respuesta:


```
{
	"encuestas": [
		{
			"nombre": "preguntas12 oficial",
			"descripcion": "unas cuantas preguntas",
			"url_foto": "http://yangmei.io/a.png",
			"id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
			"fecha_creacion": "2026-02-24 15:45:47",
			"certificacion": "1234a",
			"votacion": false,
			"anonimo": true,
			"publico": false,
			"fecha_inicio": "N/A",
			"fecha_fin": "N/A",
			"estado": 0,
			"id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1"
		}
	]
}
```


DELETE /encuesta/borrar/<id>: Borra una encuesta a partir de su id si se es dueño, esta ruta es bastante sensible y está pendiente aumentar la seguridad

PATCH /encuesta/editar/<id>: Edita ciertos datos de una encuesta solo si no ha empezado aún

Petición:


```
{
	"anonimo": true,
	"publico": true,
	"votacion": false,
	"nombre": "preguntas33",
	"descripcion": "asdeyfasuiofdy",
	"url_foto": "http://yangmei.io/a.png"
}
```


Solo se editarán los campos que estén presentes

POST /encuesta/iniciar/<id>: Inicia una encuesta solo si no había empezado ya, si tiene al menos una pregunta, era pública y además se es dueño

POST /encuesta/finalizar/<id>: Termina una encuesta solo si estaba activa

PUT /preguntas/establecer/<id>: Establece reemplazando o no las preguntas en una encuesta concreta

Petición:


```
{
	"destructivo": true,
	"preguntas": [
		{
			"titulo": "escribe texto",
			"opcional": true,
			"tipo": 0,
			"subtitulo": "abajo",
			"placeholder": "abc",
			"correcta": "defg",
			"descripcion": "descripcion hola hola"
		},{
			"titulo": "selecciona varias",
			"opcional": true,
			"contenido": ["aaa", "bbb", "ccc"],
			"tipo": 1,
			"subtitulo": "aquello",
			"placeholder": [1,2],
			"correcta": [0,1]
		},{
			"titulo": "selecciona una",
			"opcional": false,
			"contenido": ["aaa", "bbb", "ccc"],
			"tipo": 2,
			"subtitulo": "esto",
			"placeholder": 2,
			"correcta": 0,
			"descripcion": "e e e e"
		}, {
			"titulo": "escoge un numero",
			"opcional": true,
			"tipo": 3,
			"subtitulo": "aquello",
			"placeholder": 20.5,
			"correcta": -10
		},{
			"titulo": "no hace falta que respondas aqui",
			"opcional": true,
			"tipo": 2,
			"contenido": ["si", "no"]
		}
	]
}
```


Destructivo indica si se van a borrar las preguntas anteriormente creadas o no, en caso de estar en false hay que asegurarse de que no hayan preguntas repetidas.

Se admiten hasta 127 preguntas por encuesta, las cuales constan de los campos ya mencionados, y los campos contenido, placeholder y correcta dependientes del tipo. Esta petición de ejemplo cubre todas las posibilidades correctas.

GET /preguntas/verMia/<id>/<pagina?>: Ver las preguntas de una encuesta de la cual se es dueño en cierta página

Respuesta:


```
{
        "preguntas": [
                {
                        "id": "2ed4545a-01af-43c8-9c55-a55b81807ab1",
                        "titulo": "escribe texto",
                        "descripcion": "descripcion hola hola",
                        "contenido": null,
                        "opcional": true,
                        "tipo": 0,
                        "id_encuesta": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
                        "placeholder": "abc",
                        "subtitulo": "abajo",
                        "correcta": "defghi"
                }, ......
        ]
}
```


GET /respuestas/ver/<id>/<pagina?>: Ver las respuestas de determinada encuesta de la cual se es dueño solo si ya ha terminado, si la encuesta era anónima esto se puede seguir haciendo, pero no se verán los UUIDs de los usuarios

Respuesta:


```
{
        "respuestas": {
                "5db6baa5-603e-4b37-968d-040dedce9d1a": [
                        {
                                "id": "da4e602d-fa83-467e-8f8d-06669b4387f7",
                                "id_user": null,
                                "contenido": "2",
                                "id_pregunta": "5db6baa5-603e-4b37-968d-040dedce9d1a"
                        }, .......
                ], ........
        }
}
```


Por cada pregunta de la encuesta devuelve todas las respuestas, paginado

GET /respuesta/ver/<id>: Ver una respuesta concreta a partir de su id, con una lógica similar a la del endpoint anterior

Respuesta:


```
{
        "respuesta": {
                "id": "da4e602d-fa83-467e-8f8d-06669b4387f7",
                "id_user": null,
                "contenido": "2",
                "id_pregunta": "5db6baa5-603e-4b37-968d-040dedce9d1a"
        }
}
```


POST /informe/crear/<id>: Crear un informe a partir del id de una encuesta, solo si ya ha terminado y se es dueño

Petición:


```
{
        "nombre": "este informe 1",
        "publico": false,
        "opciones": "" (sin usar ahora mismo)
}
```


Respuesta:


```
{
        "informe_id": "5eb68957-500d-4935-b667-93917afea5bc"
}
```


DELETE /informe/borrar/<id>: Borrar determinado informe si se es dueño

PATCH /informe/publicar/<id>: Establece el campo publico a true en una encuesta de la cual se es dueño

GET /informe/verMio/<id>: Ver un informe del cual se es dueño, útil para informes privados

Respuesta:

<Mirar archivo adjunto EJEMPLO_INFORME.json, se incluye aparte porque era demasiado largo>

Así es como se ve un informe de ejemplo con la configuración actual

GET /informes/verMiEncuesta/<id>: Ver todos los informes de una encuesta de la cual se es dueño

Respuesta:


```
{
        "informes": [
                {
                        "id": "44d620ce-3bb4-4070-b3f5-6cc16f7ddc3c"
                }, ........
        ]
}
```

Rutas de admin

Estas rutas requieren estrictamente que el usuario del token de sesión tenga permisos de administrador, los cuales solo se pueden obtener alterando la base de datos o mediante otro administrador

GET /admin: Si esta petición no falla, es que es posible usar los endpoints de admin

PATCH /admin/usuario/alterPerms: Altera los permisos de un usuario, incluso pudiendo bloquearlo o volverlo administrador

Petición:

```
{
	"id": "019c9054-8f9c-7159-bac7-59d17a0aaea1",
	"permisos": 2
}
```

PATCH /admin/usuario/<id>/editar: Edita los datos de un usuario

Petición:

```
{
	"nickname": "Usuario-mod",
	"nombre": "El modificado",
	"email": "modificado@yangmail.com",
	"password": "123456789aA..",
	"descripcion": "esto lo ha puesto un admin realmente",
	"url_foto": "http://delivery.io/aaa.png"
}
```

DELETE /admin/usuario/<id>/borrar: Borra entero un usuario

GET /admin/usuario/<id>/ver: Ver todos los datos de cualquier usuario

Respuesta:

```
{
	"usuario": {
		"id": "019c9054-8f9c-7159-bac7-59d17a0aaea1",
		"nickname": "Usuario-mod",
		"nombre": "El modificado",
		"email": "modificado@yangmail.com",
		"descripcion": "esto lo ha puesto un admin realmente",
		"url_foto": "http://delivery.io/aaa.png",
		"permisos": 2,
		"publicante": true,
		"fecha_creacion": "2026-02-24 15:45:47",
		"created_at": "2026-02-24T15:46:26.000000Z",
		"updated_at": "2026-02-24T17:43:31.000000Z"
	}
}
```

GET /admin/encuesta/<id>/ver: Ver todos los datos de cualquier encuesta (solo de la encuesta en sí)

Respuesta:


```
{
	"encuesta": {
		"id": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
		"nombre": "preguntas12 oficial",
		"descripcion": "unas cuantas preguntas",
		"url_foto": "http://yangmei.io/a.png",
		"certificacion": "1234a",
		"votacion": false,
		"anonimo": true,
		"fecha_creacion": "2026-02-24 15:45:47",
		"fecha_inicio": "2026-02-24 16:39:29",
		"fecha_fin": "2026-02-24 17:17:06",
		"estado": 2,
		"id_user": "019c9054-8f9c-7159-bac7-59d17a0aaea1",
		"created_at": "2026-02-24T16:04:58.000000Z",
		"updated_at": "2026-02-24T17:17:06.000000Z"
	}
}
```


PATCH /admin/encuesta/<id>/editar: Edita los datos de cierta encuesta que no haya empezado aún

Petición:


```
{
	"anonimo": true,
	"publico": true,
	"votacion": false,
	"nombre": "encuesta editada",
	"descripcion": "hola nuevo",
	"url_foto": "http://yangmei.io/a.png"
}
```


DELETE /admin/encuesta/<id>/borrar: Borra una encuesta

GET /admin/preguntas/<id>/ver/<pagina?>: Ver las preguntas de cierta encuesta en cierta página

Respuesta:


```
{
        "preguntas": [
                {
                        "id": "2ed4545a-01af-43c8-9c55-a55b81807ab1",
                        "titulo": "escribe texto",
                        "descripcion": "descripcion hola hola",
                        "contenido": null,
                        "opcional": true,
                        "tipo": 0,
                        "id_encuesta": "3ec7fc79-8b7b-4333-b5f0-733651ac7fab",
                        "placeholder": "abc",
                        "subtitulo": "abajo",
                        "correcta": "defg"
                }, .....
        ]
}
```


GET /admin/respuestas/<id>/ver/<pagina?>: Ver las respuestas de cierta encuesta en cierta página

Respuesta:


```
{
        "respuestas": {
                "5db6baa5-603e-4b37-968d-040dedce9d1a": [
                        {
                                "id": "da4e602d-fa83-467e-8f8d-06669b4387f7",
                                "id_user": null,
                                "contenido": "2",
                                "id_pregunta": "5db6baa5-603e-4b37-968d-040dedce9d1a"
                        }, ........
                ], ......
        }
}
```


GET /admin/informes/<id>/ver: Ver cualquier informe

Respuesta:

<Mirar archivo adjunto EJEMPLO_INFORME.json, se incluye aparte porque era demasiado largo>

Así es como se ve un informe de ejemplo con la configuración actual

DELETE /admin/informes/<id>/borrar: Borrar un informe y sus datos

Extras

Posibles mejoras

Si bien el proyecto podría considerarse relativamente completo, aquí se mencionan algunas extensiones que habría estado bien añadir

Encuestas privadas: Posibilidad de crear encuestas restringidas solo a ciertos usuarios

Encuestas programadas: Poder establecer fechas a las encuestas para que inicien/finalicen automáticamente

Mejores informes: Actualmente el código está preparado para poder usar configuraciones a la hora de crear los informes, estaría interesante expandir sus funcionalidades y hacerlos más flexibles

Certificaciones útiles: Actualmente el campo de certificado en las encuestas no tiene más utilidad que dar más confianza al usuario, se podría implementar una manera de usar un estándar real de certificados

Limites: El backend tiene pocos límites, el mismo usuario podría hacer tantas encuestas como quisiera y hacer tantas peticiones como le apetezca, en un entorno de producción esto no suele ser así, se suelen establecer límites sobretodo por cuestiones de seguridad

Seguridad al borrar encuestas: Borrar una encuesta es una acción bastante destructiva ya que provocará el borrado en cascada de múltiples entidades, estaría bien implementar alguna manera de que no sea una acción tan fácil de hacer para aumentar la seguridad

Recuperación contraseña y correo: Si un usuario olvida su contraseña no tiene manera de recuperarla, estaría bien implementar un sistema de autenticación con correo, que realmente mande correos. Lamentablemente esto vale dinero ya que hace falta un servidor de correos y un dominio

Documentación extra

Se han usado dos tecnologías principales para el debugging y el desarrollo, una son los Seeders y Factories de Laravel, que fueron útiles para comprobar el correcto funcionamiento del paginado, stress-testing, etc.

Por otro lado se usó Insomnia para interactuar con los endpoints, ya que este proyecto es solo de backend y no incluye interfaz. Hay dos colecciones de endpoints, una con todas las rutas y otra con un test lineal en el que: se crea un usuario publicante, se editan sus datos, se ven, crea una encuesta, establece sus preguntas, edita y ve los datos, inicia la encuesta, se hace logout, se crea otro usuario pero esta vez votante, se leen los datos de la encuesta, se vota en ella, se hace logout, se hace login con el usuario anterior, se finaliza la encuesta, se crea un informe sobre esta, se publica, se hace logout, se hace login con el usuario votante y se lee el informe.

Este test no cubre todos los endpoints, pero si el flujo principal de la aplicación