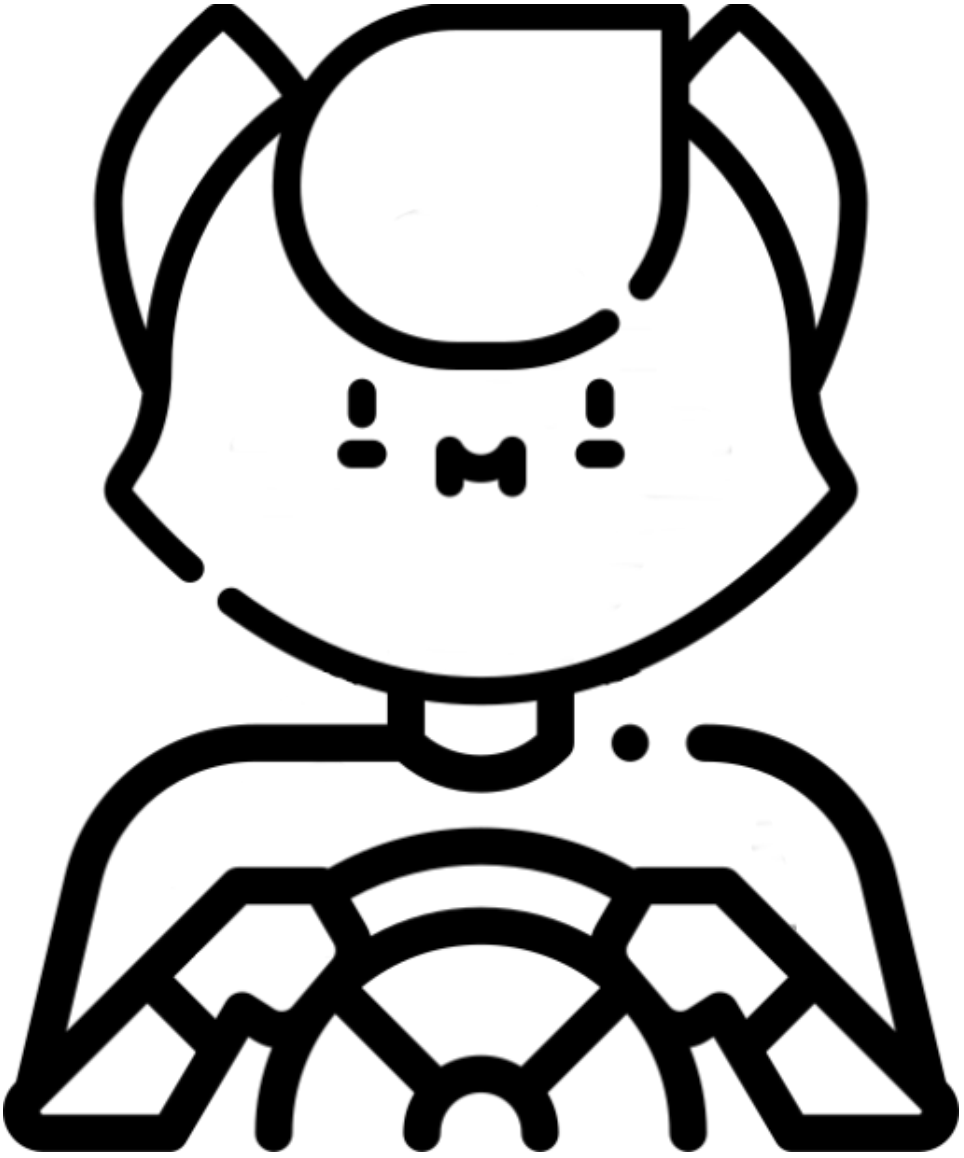


KOMUTANTES

Gestión de la información en la web

Jacinto Calderón
Jcalderon169

Contenido

Introducción

Komutantes es un servicio web que ofrece un medio de organización a personas que comparten un trayecto en coche frecuentemente.
Permite crear un calendario planificado para que los usuarios sepan qué día les toca conducir y llevar un control sobre el total de días.

Proyecto

El proyecto desarrollado para la asignatura de 'Gestión de la información en la web' consiste en una API que podrá ser consumida por cualquier cliente web, como una aplicación móvil o una página web.
La seguridad del servicio se basa en tokens de usuario, para que únicamente puedan ejecutar las distintas funcionalidades los usuarios previamente logados.
En el index del proyecto se encuentra una portada simple con enlaces a la memoria, al proyecto en github y a una colección de pruebas en Postman.
Esta página es adaptativa para utilizar cómodamente desde cualquier tipo de dispositivos.

Arquitectura

Como arquitectura principal he utilizado una estructura API REST.

La API está desarrollada con el framework de PHP Laravel.
Para el sistema de autenticación he usado el plugin Passport.
Para la gestión de fechas he usado la librería Carbon.
Como ORM uso el propio de Laravel, Eloquent.
La base de datos es MySQL. La estructura la crea Laravel automáticamente a partir de los modelos.
La portada usa el template Coming Sssoon Page, HTML5/Bootstrap.

Instalación

Descarga

- Descargar el proyecto del repositorio y descomprimir la carpeta.

Composer

- Instalar composer desde su web oficial.
- Desde la raíz del proyecto ejecutar desde consola: *composer install*

Base de datos

- Renombrar el documento *.env.example* a *.env* que se encuentra en la raíz del proyecto.
- Crear base de datos y rellenar el archivo *.env* con las credenciales.
- Ejecutar las migraciones para crear la estructura y relaciones de la base de datos: *php artisan migrate.*

Generar keys

- Esto genera las keys encriptadas para generar los tokens de acceso:

  php artisan passport:install

Arrancar el servidor

Ejecutar el siguiente comando para arrancar el servidor:

php "{ruta_a_proyecto}\komutantes\artisan" serve --host=localhost --port=8000

Funciones

Lista de endpoints con las funciones de la aplicación

GET|HEAD	/	Closure
GET|HEAD	api/asignarUsuario	App\Http\Controllers\DiaController@asignarUsuario
GET|HEAD	api/bloquearDia	App\Http\Controllers\DiaController@bloquearDia
GET|HEAD	api/desbloquearDia	App\Http\Controllers\DiaController@desbloquearDia
GET|HEAD	api/diasGrupo	App\Http\Controllers\DiaController@diasGrupo
POST	api/generarMes	App\Http\Controllers\DiaController@generarMes
POST	api/generarPlanning	App\Http\Controllers\DiaController@generarPlanning
GET|HEAD	api/gruposUsuario	App\Http\Controllers\UserController@gruposUsuario
POST	api/login	App\Http\Controllers\AuthController@login
GET|HEAD	api/logout	App\Http\Controllers\AuthController@logout
POST	api/nuevoGrupo	App\Http\Controllers\GrupoController@nuevoGrupo
GET|HEAD	api/quitarUsuario	App\Http\Controllers\DiaController@quitarUsuario
POST	api/signup	App\Http\Controllers\AuthController@signup
POST	api/unirseAGrupo	App\Http\Controllers\UserController@unirseAGrupo
GET|HEAD	api/usuariosGrupo	App\Http\Controllers\GrupoController@usuariosGrupo

Las funciones están clasificadas en los distintos controladores según el modelo al que afectan.

La función principal *generarPlanning* se ha simplificado al máximo para el proyecto.

Autenticación

Signup

Crea un nuevo usuario en el sistema.

POST: https://komutantes.herokuapp.com/api/signup

BODY: {"name":"user1",

"email":"123@x.com",

"password": "1234",

"password_confirmation": "1234"}

Devuelve el mensaje "Usuario creado". Si el email ya está registrado devuelve un error.

Login

Login de un usuario.

POST: https://komutantes.herokuapp.com/api/login

BODY: {"email":"123@x.com",

"password": "1234",

"remember_me": true}

Devuelve el token necesario para el resto de request (access_token).

Logout

Cierre de sesión de un usuario. El token ya no será válido después de ejecutarlo.

GET: https://komutantes.herokuapp.com/api/logout

AUTHORIZATION: Bearer [access_token]

Devuelve el mensaje "Ha cerrado la sesión".

Usuarios

unirseAGrupo

Asigna un usuario a un grupo.

POST: https://komutantes.herokuapp.com/api/unirseAGrupo

AUTHORIZATION: Bearer [access_token]

BODY: { "usuarioId":"1",

"codigo":"3u4o9Jp"}

Devuelve el mensaje "Usuario unido al grupo".

gruposUsuario

Devuelve los grupos en los que está un usuario.

GET: https://komutantes.herokuapp.com/api/gruposUsuario?usuarioId=1

AUTHORIZATION: Bearer [access_token]

PARAMETROS: usuarioId

Devuelve la lista de grupos del usuario consultado. Ejemplo:


```
[{"id": 1,
  "nombre": "grupo1",
  "normas": "Normas de conductas del grupo",
  "codigo": "3u4o9Jp",
  "created_at": "2020-02-19 10:44:40",
  "updated_at": "2020-02-19 10:44:40",
  "pivot": {
     "user_id": 1,
     "grupo_id": 1}
   }]
```

Grupos

nuevoGrupo

Crea un grupo nuevo. Devuelve un código necesario para que se unan usuarios.

POST: https://komutantes.herokuapp.com/api/nuevoGrupo

AUTHORIZATION: Bearer [access_token]

BODY: {"nombre":"grupo1",

"normas":"Normas de conductas del grupo"}

Devuelve el código de asignación del nuevo grupo. Alfanumérico de 7 caracteres.

(Ejemplo: 3u4o9Jp)

usuariosGrupo

Devuelve los usuarios que pertenecen a un grupo.

GET: https://komutantes.herokuapp.com/api/usuariosGrupo?grupoId=1

AUTHORIZATION: Bearer [access_token]

PARAMETROS: grupoId

Devuelve la lista de los usuarios del grupo consultado. Ejemplo:


```
[{"id": 1,
  "name": "user1",
  "email": "123@x.com",
  "email_verified_at": null,
  "created_at": "2020-02-18 13:48:21",
  "updated_at": "2020-02-18 13:48:21",
  "pivot": {
     "grupo_id": 1,
     "user_id": 1}
   }]
```

Días

generarMes

Genera los días laborables del mes indicado y los asocia al grupo indicado.

POST: https://komutantes.herokuapp.com/api/generarMes

AUTHORIZATION: Bearer [access_token]

BODY: { "mes":"2",

"grupoId":"1"}

Devuelve todos los días generados. Una vez generados los días del mes se pueden asignar los usuarios manualmente o generar el planning con asignación correlativa.

generarPlanning

Toma los días de un mes y los asocia correlativamente a un grupo de usuarios. Devuelve la lista de los días con los usuarios ya asignados.

POST: https://komutantes.herokuapp.com/api/generarPlanning

AUTHORIZATION: Bearer [access_token]

BODY: { "mes":"2",

"grupoId":"1"}

Devuelve todos los días con los usuarios del grupo asignados.

bloquearDia

Marca un día concreto como bloqueado. Este método se utiliza en un caso de uso no implementado para la entrega.

GET: https://komutantes.herokuapp.com/api/bloquearDia?grupoId=1&fecha=2020-02-04

AUTHORIZATION: Bearer [access_token]

PARAMETROS: grupoId

fecha (yyyy-mm-dd)

Devuelve el mensaje "Día bloqueado".

desbloquearDia

Marca un día concreto como libre. Este método se utiliza en un caso de uso no implementado para la entrega.

GET: https://komutantes.herokuapp.com/api/desbloquearDia?grupoId=1&fecha=2020-02-04

AUTHORIZATION: Bearer [access_token]

PARAMETROS: grupoId

fecha (yyyy-mm-dd)

Devuelve el mensaje “Día desbloqueado”.

diasGrupo

Devuelve los días de un mes para un grupo determinado

GET: https://komutantes.herokuapp.com/api/diasGrupo?grupoId=1&mes=2

AUTHORIZATION: Bearer [access_token]

PARAMETROS: grupoId

mes

Devuelve la lista de días consultada.

asignarUsuario

Asigna un usuario a un día.

GET: https://komutantes.herokuapp.com/api/quitarUsuario?grupoId=1&usuarioId=2&fecha=2020-02-04

AUTHORIZATION: Bearer [access_token]

PARAMETROS: grupoId

usuarioId

fecha (yyyy-mm-dd)

Devuelve el mensaje “Usuario asignado”.

quitarUsuario

Quita a un usuario previamente asignado de un día.

GET: https://komutantes.herokuapp.com/api/quitarUsuario?grupoId=1&fecha=2020-02-04

AUTHORIZATION: Bearer [access_token]

PARAMETROS: grupoId

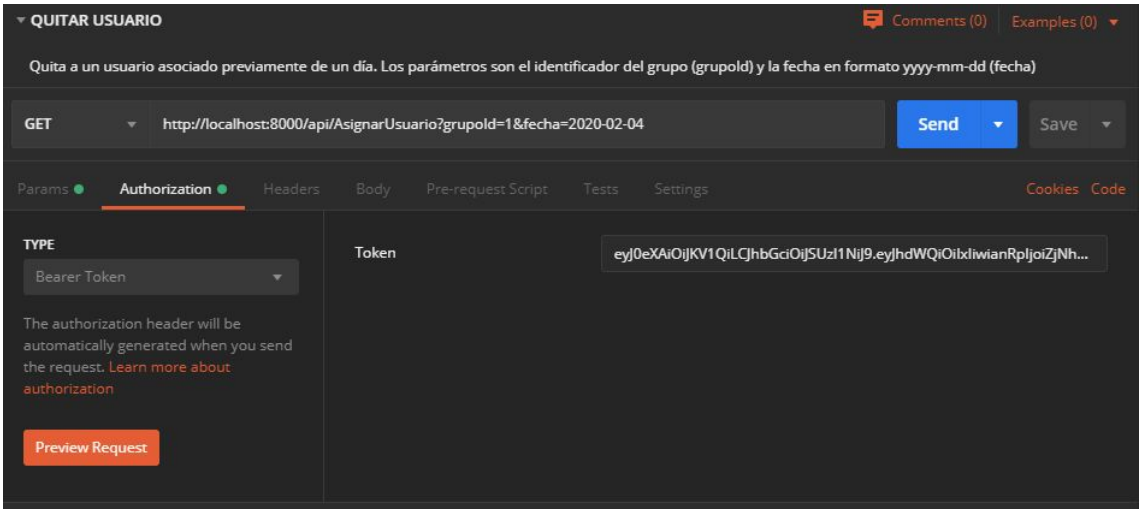
fecha (yyyy-mm-dd)

Devuelve el mensaje “Día sin asignación”.

Pruebas

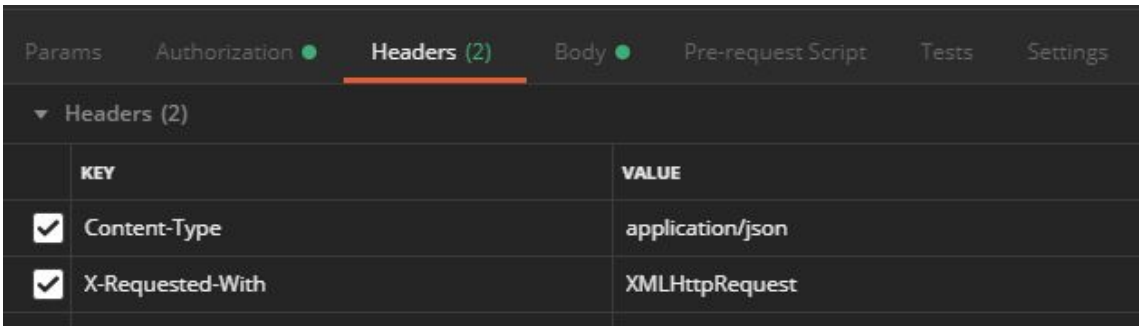
Para probar las funciones de la API se facilita una colección de request para Postman.

En los endpoints que se ejecutan con usuario logado hay que especificar el token de usuario en cada ejecución. En el caso del cliente Postman, el token se puede indicar en la pestaña 'Authorization', con el tipo Bearer Token. Esto se convertirá en una entrada en la cabecera de la petición.



Si el cliente que se utiliza no tiene la opción de añadir el token como tal, se debe especificar en la cabecera con la palabra Bearer antes del token:

Además, todas las requests de tipo POST, sean autenticadas o no, deben llevar los siguientes parámetros en la cabecera:



Los parámetros en las request de tipo POST irán en formato JSON: