API Laravel

El objetivo es desarrollar una plataforma donde gestionar tareas. La aplicación permite a los usuarios gestionar su perfil, configurar sus preferencias de interfaz y organizar sus flujos de trabajo mediante tareas y etiquetas categorizadas.

Se evaluará no solo la funcionalidad del código, sino también la adherencia a los estándares HTTP, la seguridad en el manejo de datos sensibles y la integridad de las relaciones en la base de datos.

1. Registro de usuario

- **Ruta:** `POST /api/register`
- **Controlador:** `UserController@register`
- **Descripción:** Implementación de un sistema de registro que asegura la integridad de los datos mediante el uso de `Validator`. El sistema procesa los siguientes campos:
- **Campos del Request:**
    - **`name`**: (Obligatorio) Tipo string, máximo 255 caracteres.
    - **`lastname`**: (Obligatorio) Tipo string, máximo 255 caracteres.
    - **`email`**: (Obligatorio) Formato email válido, máximo 255 caracteres y único en la tabla `users`.
    - **`password`**: (Obligatorio) Mínimo 6 caracteres. Debe guardarse encriptado.
    - **`confirm_password`**: (Obligatorio) Debe coincidir exactamente con el campo `password`.
    - **`biography`**: (Opcional) Tipo string, máximo 255 caracteres.
    - **`website`**: (Opcional) Tipo string, máximo 255 caracteres.
- **Respuestas:**
    - **201:** si el usuario se ha creado con éxito.
    - **400:** si la validación falla.

---

2. Actualización de usuario

- **Ruta:** `PUT /api/user/{id}`
- **Controlador:** `UserController@update`
- **Descripción:** Permite modificar los datos del perfil existente validando que la información sea correcta.
- **Campos del Request:**
  - **`name`**: (Obligatorio) Tipo string, máximo 255 caracteres.
  - **`lastname`**: (Obligatorio) Tipo string, máximo 255 caracteres.
  - **`email`**: (Obligatorio) Formato email válido. Debe ser único.
  - **`biography`**: (Opcional) Tipo string, máximo 255 caracteres.
  - **`website`**: (Opcional) Tipo string, máximo 255 caracteres.
- **Respuestas:**
  - **200:** si el usuario se ha creado con éxito.
  - **400:** si la validación falla.

---

3. Borrado de Usuario

- **Ruta:** `DELETE /api/user/{id}`
- **Controlador:** `UserController@destroy`
- **Descripción:** Elimina de forma permanente un usuario del sistema.
- **Respuestas:**
  - **200:** confirmando la eliminación.
  - **404:** si el usuario no existe.

---

4. Detalle de Usuario

- **Ruta:** `GET /api/user/{user}`
- **Controlador:** `UserController@show`
- **Descripción:** Devuelve la información completa del perfil de un usuario específico, permitiendo la visualización de los datos extendidos del perfil.
- **Respuestas:**
  - **200:** y un JSON con los datos del usuario solicitado (id, name, lastname, email, biography y website).
  - **404:** Si el ID del usuario no existe en la base de datos.

5. Inicio de sesión de Usuario

- **Ruta:** `POST /api/login`
- **Controlador:** `UserController@login`
- **Descripción:** Permite el acceso de usuarios registrados validando las credenciales.
- **Campos:** Requiere `email` y `password`.
- **Respuestas:**
  - **400:** Si la validación falla.
  - **401:** Si las credenciales son incorrectas.
  - **200:** si las credenciales son correctas.La respuesta debe incluir un JSON con los datos del usuario que ha iniciado sesión.

---

6. Preferencias - Relación 1:1

- **Ruta:** **`GET /api/user/{user}/preferences`**
- **Controlador:** **`UserController@getPreferences`**
- **Descripción:** Accede a la configuración personalizada del usuario a través de la relación de elocuente.
- **Requisito Eloquent:** Uso de la relación definida en el modelo.
- **Respuestas:**
  - **200:** JSON con los campos **`dark_mode`**, **`notifications_enabled`** y **`compact_view`**.
  - **404:** Si el usuario no existe.

---

7. Tareas - Relación 1:N

- **Ruta:** **`GET /api/user/{user}/tasks`**
- **Controlador:** **`UserController@getTasks`**
- **Descripción:** Obtiene el listado de todas las tareas vinculadas al usuario.
- **Requisito Eloquent:** Uso de la relación definida en el modelo.
- **Respuestas:**
  - **200:** Array JSON conteniendo objetos con `title` y `description`.
  - **404:** Si el usuario no existe.

8. Etiquetas de una Tarea - Relación N:M

- **Ruta:** `GET /api/task/{task}/labels`
- **Controlador:** `TaskController@getLabels`
- **Descripción:** Accede a la relación muchos a muchos. Devuelve el listado de todas las etiquetas asociadas.
- **Requisito Eloquent:** Uso de la relación definida en el modelo.
- **Respuestas:**
    - **200:** Array JSON con las etiquetas vinculadas (campos `id` y `name`).
    - **404:** Si la tarea no existe.