BichoPedia 2

1. Introducción

BichoPedia es un proyecto que viene desde la afición a los animales por parte del autor, en el que se puede ver la forma de “capturar bichos” como un hobby más entretenido e informativo.

2. Plan de Empresa

- JUSTIFICACIÓN:

Este negocio viene de la inspiración por el autor de capturar bichos y registrarlos en alguna aplicación, existen algunas apps en el mercado que ya cubre esta necesidad pero este viene con un enfoque más local, interactivo intuitivo y simple.

- NOMBRE y LOGO

El nombre y el logo aún no son definitivos, es más aún no hay ni logo, pienso que el logo debería realizarse al final del desarrollo de la misma app para que puedan surgir ideas del logo.
El nombre “BichoPedia 2” viene de que ya hubo un proyecto llamado bichopedia, pero este lo realicé con menos conocimientos, el término bichopedia viene de las palabras “bicho” que es la entidad por la que se maneja la aplicación y pedía de enciclopedia ya que también funciona como una enciclopedia.

- PRODUCTO

Se va a ofrecer al público una aplicación móvil en la que podrías consultar las características y los datos de una especie así como el encuentro que haya registrado alguien de una especie, a su misma vez podrás ir registrando tus encuentros y subir de nivel para ser un usuario más reconocido.

- MODELO DE NEGOCIO

Como modelo de negocio se va a usar el sistema Modelo freemium que permite que las empresas ofrezcan de forma gratuita sus servicios básicos en internet, sin embargo, en caso de necesitar una ampliación de estos, el usuario tendrá que pagar por ello.

Este modelo permite captar la atención de clientes que comienzan usando el servicio sin pagar, pero que terminan precisando un servicio más completo y se convierten en usuarios del paquete completo.

- CONSUMIDORES Y POSIBLES CLIENTES

Los posibles clientes pueden ser o bien aficionados a la herpetología y entomología o jóvenes interesados en la naturaleza, así también se podría usar como material educativo en centros de primaria en las asignaturas de Biología.

- COMPETENCIA

Naturalist, aunque este tiene más un enfoque identificativo, más que informativo.

- DAFO

Debilidades

La debilidad que tiene este proyecto es que apenas nadie se interesa ya por la fauna local y la biodiversidad.

Amenazas

Las amenazas pueden surgir en torno a la competencia ya mencionada en el apartado anterior.

Fortalezas

La fortaleza que tiene es que no necesita de muchos recursos y se retroalimenta de información.

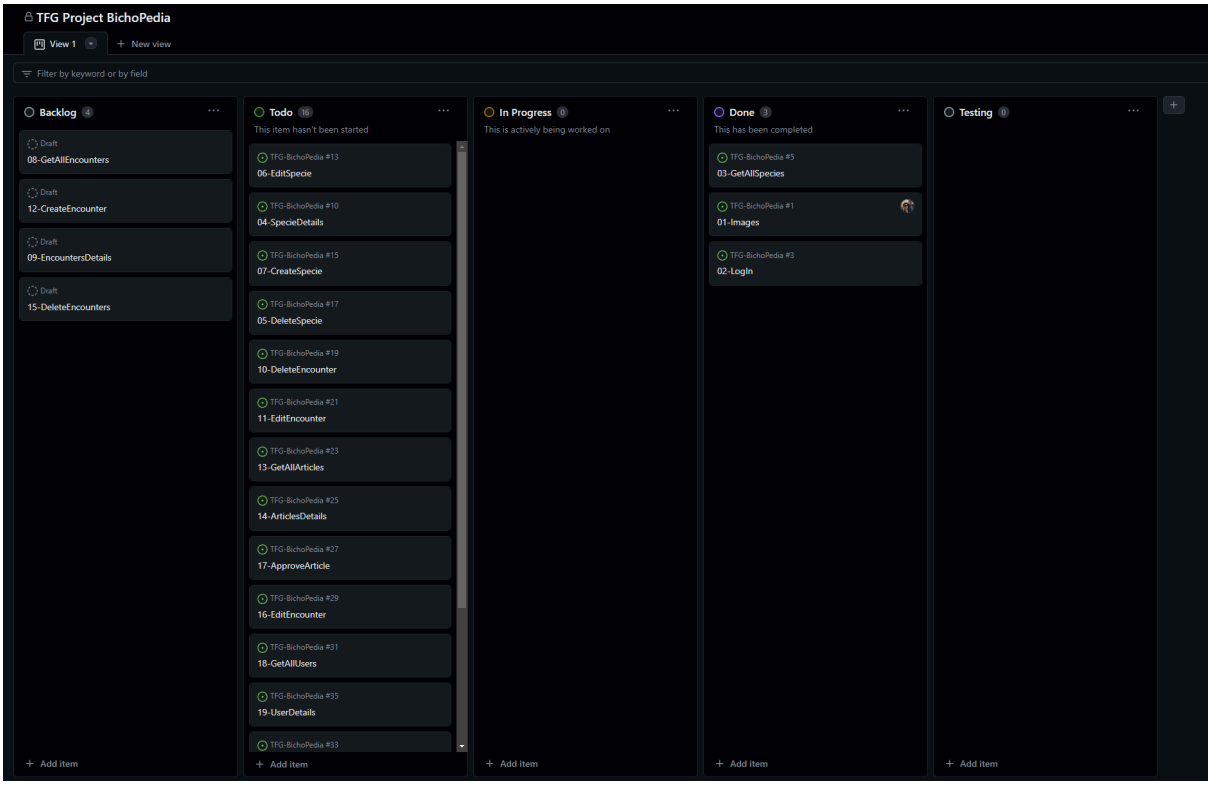
Oportunidades

Muy pocas oportunidades, dependiendo del cliente.

- PUBLICIDAD Y PROMOCIÓN

La publicidad se espera abrir proyectos con las concejalías de medioambiente de los municipios cercanos y asociaciones para fomentar la pasión por la naturaleza y que ellos nos puedan otorgar publicidad, ya que esta aplicación tiene más uso en un pueblo que en una ciudad.

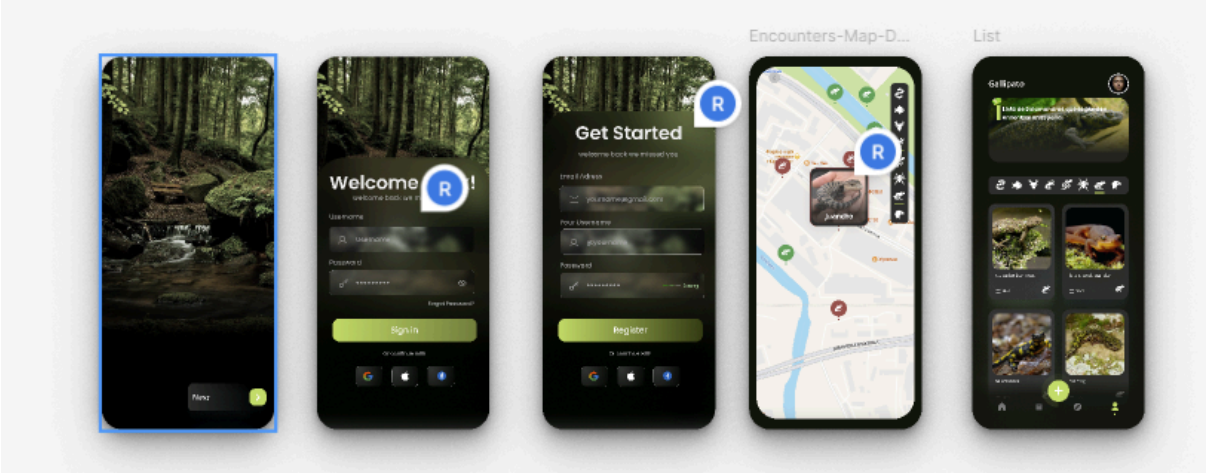
3. Descripción detallada del sistema y listado de historias de usuario, siguiendo el formato de

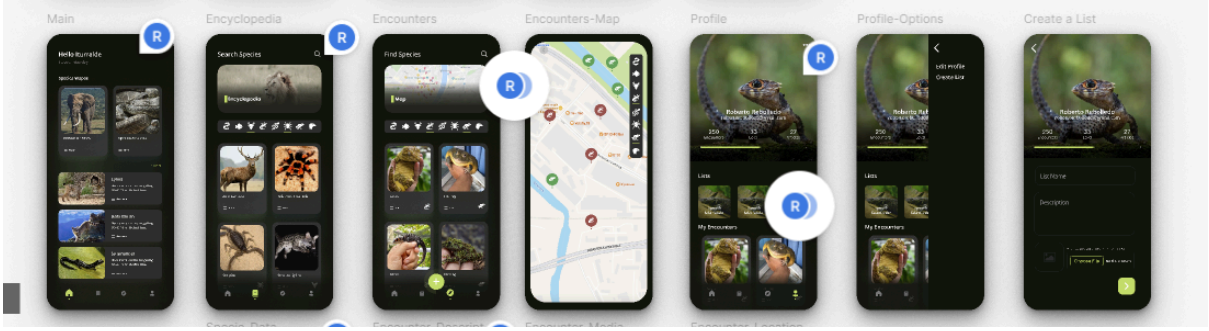


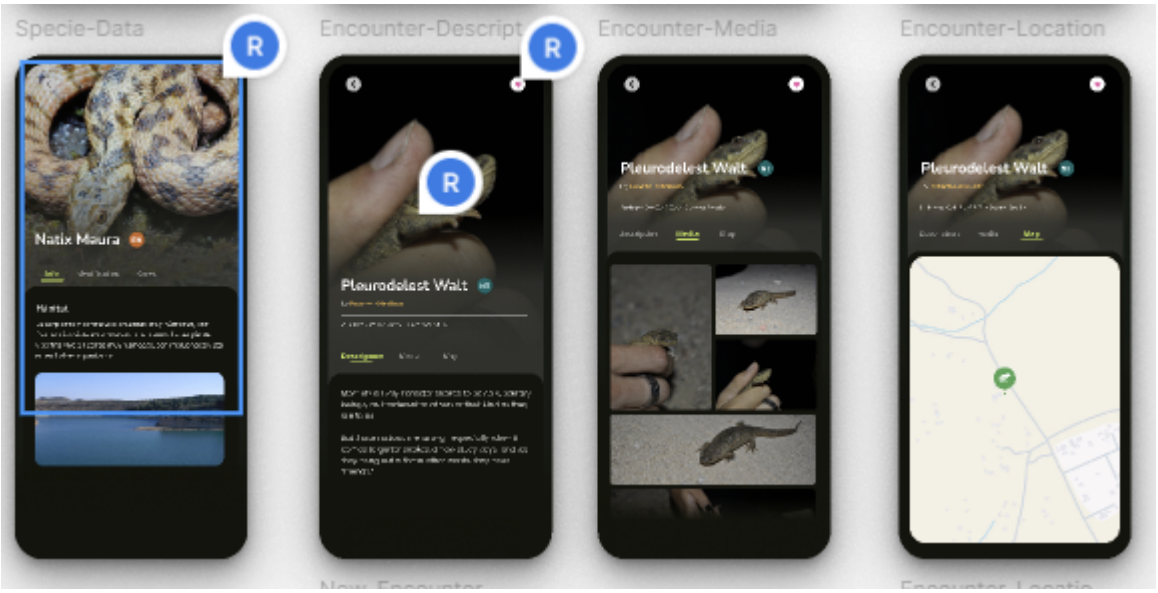
Las HU están más bien enfocadas por endpoints, sus 4 columnas ya nos dicen en la fase en la que están.
Las HU comparten tanto parte front como back por lo que ya hay tareas que se hicieron previamente en el back y no se han desarrollado aún en el front (backLog) y tareas que se han realizado una parte back o front pero no se ha realizado la otra correspondiente (toDo).

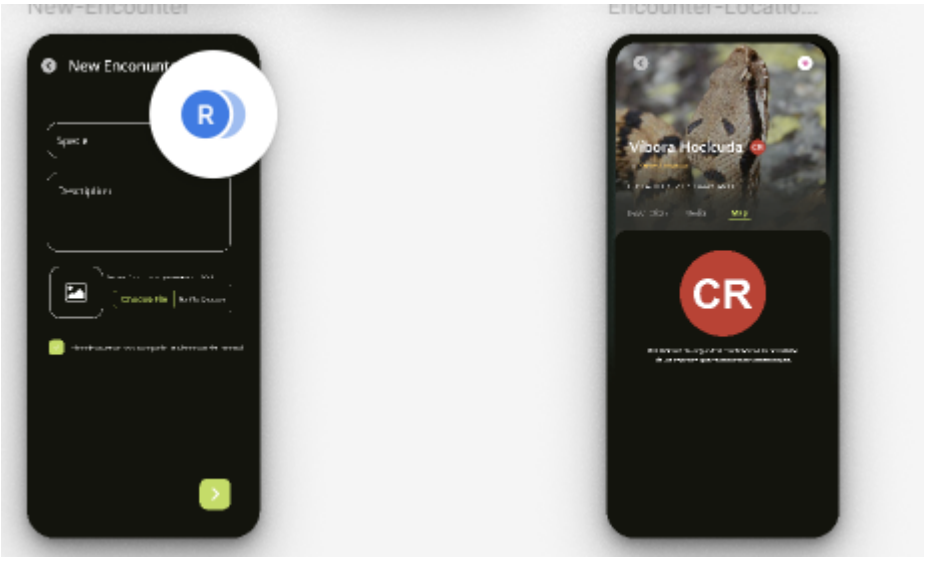
4. Modelado y diseño

4.1 Prototipado en Figma de las diferentes aplicaciones (móvil y web).

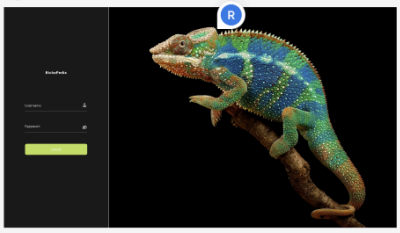


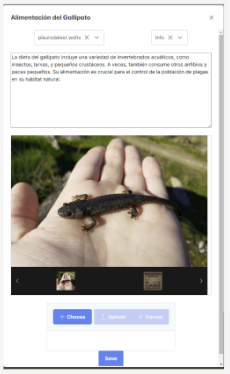


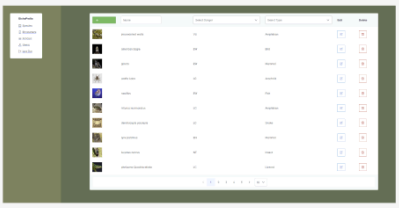

Specie-Data
Encounter-Descript
Encounter-Media
Encounter-Location
Natix Maura
Pleurodelest Walt
Pleurodelest Walt
Pleurodelest Walt


New Encounter
Encounter-Locatio...
New Enconunt
Vibora Hocicuda
CR

Login



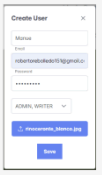

Alimentación del Gallipato
La dieta del gallipato incluye una variedad de invertebrados acuáticos, como insectos, larvas, y pequeños crustáceos. A veces, también consume otros anfibios y peces pequeños. Su alimentación es crucial para el control de la población de plagas en su hábitat natural.
Choose
Upload
Cancel
Save




pleurodeles waltl
Select a Danger
Select a Type
Save


Are you sure to remove this specie?
All articles on encounter related to the specie will be removed.
No
Yes


Edit User Info
Save


Create User
ADMIN, WRITER
Save


Rejected
You have rejected

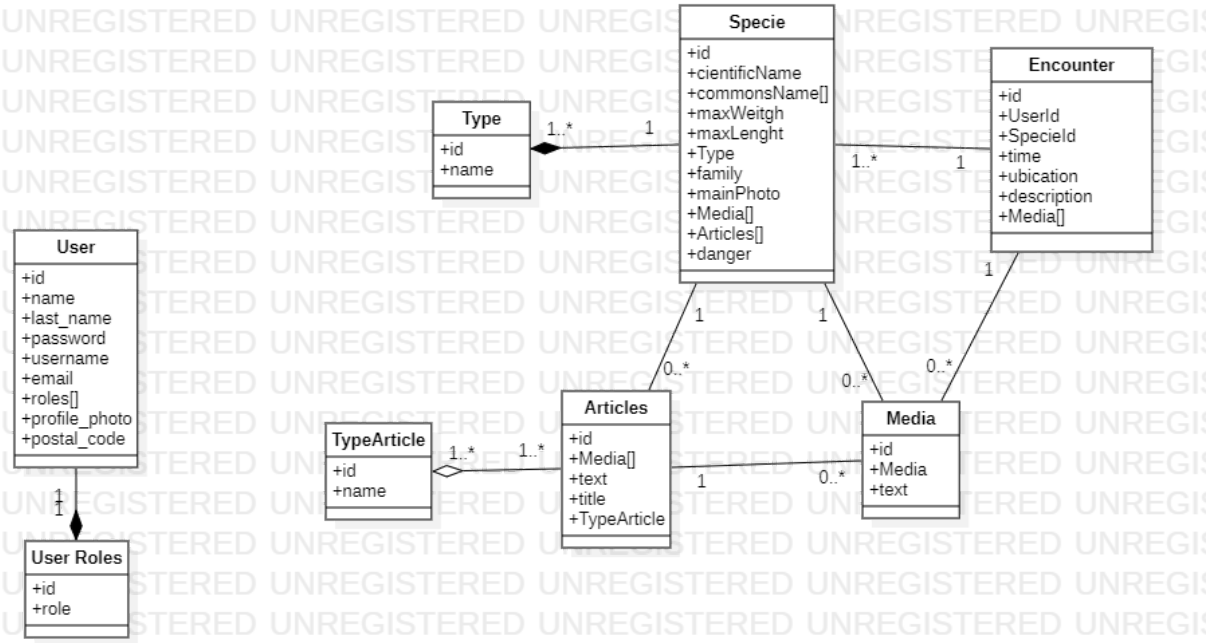

Add New Specie
scientificName
Select a Danger
Select a Type
Save


Success
The specie was removed


Approved

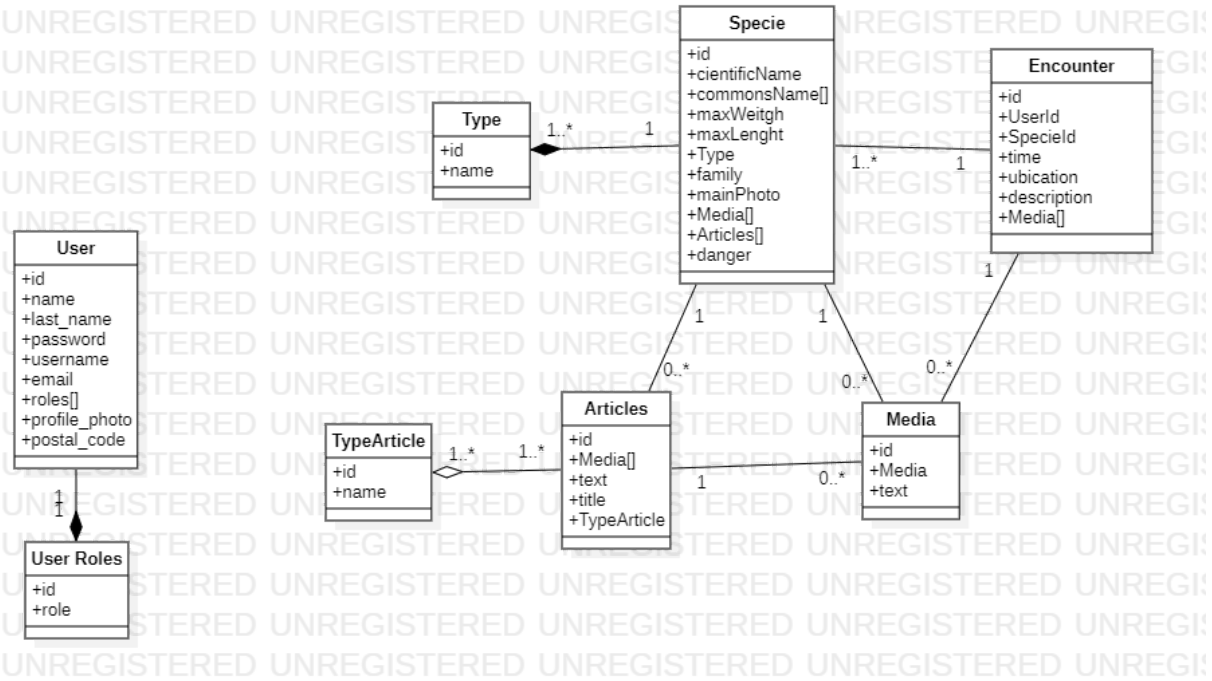

Warning
No Species was found with scientificName=Miguel

4.2 Diagrama de clases del modelo de dominio de la API.



5. Diseño

5.1 Diagrama de clases del diseño indicando: direccionalidad de las asociaciones, tipos y los tipos de datos a utilizar.



5.2 Diseño los contratos (DTOs) de los diferentes endpoints

Endpoints del Controlador `ArticleControllerWriter`

GET `/writer/articles/allArticles`

- **Descripción:** Obtiene todos los artículos.
- **Parámetros de consulta:**
  - `search` (opcional): Búsqueda avanzada por término.
  - `c` (opcional, por defecto 100): Cantidad de resultados por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**
  - **200 OK:** Lista de objetos `GETArticleSimpleDTO`.
  - **500 Error interno:** Error del servidor.

POST `/writer/articles/create`

- **Descripción:** Crea un nuevo artículo.
- **Cuerpo de la solicitud:** Objeto `POSTArticleDTO`.
- **Respuestas:**
    - **200 OK:** Objeto `GETArticleSimpleDTO` del artículo creado.
    - **400 Datos inválidos:** Los datos proporcionados son inválidos.
    - **500 Error interno:** Error del servidor.

PUT `/writer/articles/edit/{id}`

- **Descripción:** Edita un artículo existente por su ID.
- **Parámetros de ruta:**
    - `id`: ID del artículo a editar.
- **Cuerpo de la solicitud:** Objeto `PUTArticleDTO`.
- **Respuestas:**
    - **200 OK:** Objeto `GETArticleSimpleDTO` del artículo editado.
    - **400 Datos inválidos:** Los datos proporcionados son inválidos.
    - **404 No encontrado:** El artículo con el ID especificado no existe.
    - **500 Error interno:** Error del servidor.

GET `/writer/articles/changeApprovedArticle/{id}`

- **Descripción:** Cambia el estado de aprobación de un artículo por su ID.
- **Parámetros de ruta:**
    - `id`: ID del artículo cuyo estado de aprobación se cambiará.
- **Respuestas:**
    - **200 OK:** Objeto `GETArticleSimpleDTO` del artículo con el estado de aprobación cambiado.
    - **404 No encontrado:** El artículo con el ID especificado no existe.
    - **500 Error interno:** Error del servidor.

GET `/writer/articles/details/{id}`

- **Descripción:** Obtiene los detalles de un artículo por su ID.
- **Parámetros de ruta:**
    - `id`: ID del artículo del cual se obtendrán los detalles.
- **Respuestas:**
    - **200 OK:** Objeto `GETArticleDetailsDTO` con los detalles del artículo.
    - **404 No encontrado:** El artículo con el ID especificado no existe.
    - **500 Error interno:** Error del servidor.

DTOs Relacionados

`GETArticleSimpleDTO`

```
public record GETArticleSimpleDTO(
    String id,
    String specieName,
```

```
    String title,
    boolean approved,
    String type
) {}
```

GETArticleDetailsDTO

```
public record GETArticleDetailsDTO(
    String id,
    String title,
    String text,
    boolean approved,
    List<String> archives,
    String specieId,
    String createdBy,
    String type
) {}
```

POSTArticleDTO

```
public record POSTArticleDTO(
    String title,
    String text,
    List<String> medias,
    String userId,
    String specieId,
    String type
) {}
```

PUTArticleDTO

```
public record PUTArticleDTO(
    boolean approved,
    String text,
    List<String> medias,
    String userId,
    String specieId,
    String type
) {}
```

Endpoints del Controlador `EncounterController`

GET `/user/encounters/most-liked/simple`

- **Descripción:** Obtiene las especies más gustadas con paginación.
- **Parámetros de consulta:**
  - `c` (opcional, por defecto 10): Número de elementos por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**
  - **200 OK:** Lista de objetos `GETEncounterSimpleDTO`.
  - **400 Datos inválidos:** Los datos proporcionados son inválidos.
  - **500 Error interno:** Error del servidor.

GET `/user/encounters/allencounters`

- **Descripción:** Obtiene todos los encuentros por criterio.
- **Parámetros de consulta:**
  - `search` (opcional): Criterio de búsqueda para los encuentros.
  - `c` (opcional, por defecto 10): Número de encuentros por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**
  - **200 OK:** Lista de objetos `GETEncounterDTO`.
  - **400 Datos inválidos:** Los datos proporcionados son inválidos.
  - **500 Error interno:** Error del servidor.

GET `/user/encounters/myencounters/{id}`

- **Descripción:** Obtiene los encuentros de un usuario por ID.
- **Parámetros de consulta:**
  - `c` (opcional, por defecto 10): Número de encuentros por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**
  - **200 OK:** Lista de objetos `GETEncounterDTO`.
  - **400 Datos inválidos:** Los datos proporcionados son inválidos.
  - **404 No encontrado:** El usuario con el ID especificado no existe.
  - **500 Error interno:** Error del servidor.

GET `/user/encounters/allmarkers`

- **Descripción:** Obtiene todos los marcadores de encuentros.
- **Respuestas:**
  - **200 OK:** Lista de objetos `GETMarker`.
  - **500 Error interno:** Error del servidor.

GET `/user/encounters/encounterdetails/{id}`

- **Descripción:** Obtiene los detalles de un encuentro por ID.

- **Parámetros de ruta:**
  - `id`: ID del encuentro del cual se obtendrán los detalles.
- **Respuestas:**
  - **200 OK:** Objeto `GETEncounterDetailDTO` con los detalles del encuentro.
  - **400 Datos inválidos:** El ID del encuentro no tiene un formato válido.
  - **404 No encontrado:** El encuentro con el ID especificado no existe.
  - **500 Error interno:** Error del servidor.

POST `/user/encounters/find/`

- **Descripción:** Guarda un nuevo encuentro.
- **Cuerpo de la solicitud:** Objeto `POSTEncounterDTO`.
- **Respuestas:**
  - **201 Creado:** Objeto `POSTEncounterDTO` del encuentro guardado.
  - **400 Datos inválidos:** Los datos proporcionados para el encuentro son inválidos.
  - **401 No autorizado:** El usuario no está autorizado para realizar esta acción.
  - **500 Error interno:** Error del servidor.

DELETE `/user/encounters/{id}`

- **Descripción:** Elimina un encuentro por ID.
- **Parámetros de ruta:**
  - `id`: ID del encuentro que se desea eliminar.
- **Respuestas:**
  - **202 Aceptado:** El encuentro se eliminó correctamente.
  - **401 No autorizado:** El usuario no está autorizado para realizar esta acción.
  - **404 No encontrado:** El encuentro con el ID especificado no existe.
  - **500 Error interno:** Error del servidor.

PUT `/user/encounters/`

- **Descripción:** Edita un encuentro existente.
- **Cuerpo de la solicitud:** Objeto `PUTEncounterDTO`.
- **Respuestas:**
  - **201 Creado:** Objeto `GETEncounterDetailDTO` del encuentro editado.
  - **400 Datos inválidos:** Los datos proporcionados para el encuentro son inválidos.
  - **401 No autorizado:** El usuario no está autorizado para realizar esta acción.
  - **404 No encontrado:** El encuentro que se intenta editar no existe.
  - **500 Error interno:** Error del servidor.

Endpoints del Controlador `EncounterControllerEditor`

GET `/writer/encounters/allencounters`

- **Descripción:** Obtiene todos los encuentros.
- **Parámetros de consulta:**
  - `c` (opcional, por defecto 10): Número de encuentros por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**
  - **200 OK:** Lista de objetos `GETEncounterDetailDTO`.
  - **500 Error interno:** Error del servidor.

PUT `/writer/encounters/`

- **Descripción:** Edita un encuentro existente.
- **Cuerpo de la solicitud:** Objeto `PUTEncounterDTO`.
- **Respuestas:**
  - **201 Creado:** Objeto `GETEncounterDetailDTO` del encuentro editado.
  - **400 Datos inválidos:** Los datos proporcionados para el encuentro son inválidos.
  - **500 Error interno:** Error del servidor.

DELETE `/writer/encounters/{id}`

- **Descripción:** Elimina un encuentro por ID.
- **Parámetros de ruta:**
  - `id`: ID del encuentro que se desea eliminar.
- **Respuestas:**
  - **202 Aceptado:** El encuentro se eliminó correctamente.
  - **404 No encontrado:** El encuentro con el ID especificado no existe.
  - **500 Error interno:** Error del servidor.

DTOs Relacionados

GETEncounterDetailDTO

```
public record GETEncounterDetailDTO(
    String id,
    String scientificName,
    String mainPhoto,
    String danger,
    String type,
    String date,
    String username,
    String description,
    List<String> media,
    String lat,
    String lon
) {}
```

GETEncounterDTO

```
public record GETEncounterDTO(
    UUID id,
    String url,
    String scientificName,
    String type
) {}
```

GETEncounterSimpleDTO

```
public record GETEncounterSimpleDTO(
    UUID id,
    String scientificName,
    String description,
    String photo
) {}
```

GETMarker

```
public record GETMarker(
    String id,
    String latitud,
    String longuitud
) {}
```

POSTEncounterDTO

```
public record POSTEncounterDTO(
    String specieId,
    String description,
    String location,
    List<String> photos
) {}
```

PUTEncounterDTO

```
public record PUTEncounterDTO(
    String encounterId,
    String specieId,
    String description,
    String location,
    LocalDate date,
    List<String> photos
) {}
```

Endpoints del Controlador `SpecieController`

GET `/user/species/danger-extinction/simple`

- **Descripción:** Obtiene una lista de especies en peligro de extinción con paginación.
- **Parámetros de consulta:**
  - `c` (opcional, por defecto 10): Número de elementos por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**
  - **200 OK:** Lista de objetos `SpecieSimpleDTO`.
  - **400 Datos inválidos:** Los datos de entrada son inválidos.
  - **404 No encontrado:** Especies no encontradas.
  - **500 Error interno:** Error del servidor.

GET `/user/species/allspecies`

- **Descripción:** Obtiene una lista de todas las especies o por criterios de búsqueda.
- **Parámetros de consulta:**
  - `search` (opcional): Búsqueda avanzada por nombre científico o tipo de especie.
  - `c` (opcional, por defecto 10): Número de elementos por página.
  - `p` (opcional, por defecto 0): Número de página.
- **Respuestas:**

    - **200 OK:** Lista de objetos `SpecieDTO` o `SpecieNotFoundException`.
    - **400 Datos inválidos:** Los datos de entrada son inválidos.
    - **404 No encontrado:** Especies no encontradas.
    - **500 Error interno:** Error del servidor.

GET `/user/species/speciebyid/{id}`

- **Descripción:** Obtiene los detalles de una especie por su ID.
- **Parámetros de ruta:**
    - `id`: ID de la especie.
- **Respuestas:**
    - **200 OK:** Detalles de la especie en un objeto `SpecieDetailsDTO`.
    - **400 Datos inválidos:** Los datos de entrada son inválidos.
    - **404 No encontrado:** Especie no encontrada.
    - **500 Error interno:** Error del servidor.

GET `/user/species/names`

- **Descripción:** Obtiene todos los nombres de las especies.
- **Respuestas:**
    - **200 OK:** Lista de objetos `SpeciesNameDTO`.
    - **500 Error interno:** Error del servidor.

DTOs Relacionados

`SpecieArticlesDTO`

```
@Builder
public record SpecieArticlesDTO(
        String id,
        String scientificName,
        List<GETArticleSimpleDTO> articles
) {}
```

`SpecieDetailsDTO`

```
@Builder
public record SpecieDetailsDTO(
        String scientificName,
        String danger,
        String mainPhoto,
        List<GETArticleDTO> info,
```

```
        List<GETArticleDTO> identification,
        List<GETArticleDTO> cares
) {}
```

SpecieDTO

```
@Builder
public record SpecieDTO(
        UUID id,
        String url,
        String scientificName,
        String danger,
        String type
) {}
```

SpeciePostDTO

```
@Builder
public record SpeciePostDTO(
        String scientificName,
        String danger,
        String mainPhoto,
        String type
) {}
```

SpeciePutDTO

```
@Builder
public record SpeciePutDTO(
        String id,
        String scientificName,
        String danger,
        String mainPhoto,
        String type
) {}
```

SpecieSimpleDTO

```
public record SpecieSimpleDTO(
        UUID id,
        String url,
        String scientificName
) {}
```

SpeciesNameDTO

```
@Builder
public record SpeciesNameDTO (
        String id,
        String name
) {}
```

Endpoints del Controlador `SpecieControllerWriter`

PUT `/writer/species/`

- **Descripción:** Actualiza los detalles de una especie existente.
- **Operación:** `editSpecie`
- **Respuestas:**
    - **201 OK:** Especie editada exitosamente en formato `SpecieDTO`.
    - **400 Datos inválidos:** Los datos proporcionados para la especie son inválidos.
    - **500 Error interno:** Error del servidor.

Ejemplo de cuerpo de solicitud:

```
{
    "id": "00000000-0000-0000-0000-000000000000",
    "scientificName": "Scientific Name",
    "danger": "High",
    "mainPhoto": "http://example.com/specie/1",
    "type": "Type"
}
```

-

POST `/writer/species/`

- **Descripción:** Registra una nueva especie.
- **Operación:** `registerSpecie`
- **Respuestas:**
    - **201 OK:** Especie registrada exitosamente en formato `SpecieDTO`.
    - **400 Datos inválidos:** Los datos proporcionados para la especie son inválidos.
    - **500 Error interno:** Error del servidor.

Ejemplo de cuerpo de solicitud:

```
{
    "scientificName": "Scientific Name",
    "danger": "High",
    "mainPhoto": "http://example.com/specie/1",
    "type": "Type"
}
```

-

DELETE `/writer/species/{id}`

- **Descripción:** Elimina una especie por su ID.
- **Operación:** `deleteSpecie`
- **Respuestas:**
  - **202 Accepted:** Especie eliminada exitosamente.
  - **404 Not Found:** La especie no fue encontrada.
- **Ejemplo de URL de solicitud:**
  `/writer/species/00000000-0000-0000-0000-000000000000`

Endpoints del Controlador `AuthController`

POST `/auth/register`

- **Descripción:** Registra un nuevo usuario.
- **Operación:** `createPersonWithUserRole`
- **Respuestas:**
  - **201 Created:** Usuario registrado exitosamente en formato `JwtUserResponse`.
  - **400 Bad Request:** Los datos introducidos no son válidos.

Ejemplo de cuerpo de solicitud:

```
{
    "username": "rducker0",
    "password": "password"
}
```

-

POST `/auth/login`

- **Descripción:** Inicia sesión para un usuario existente.
- **Operación:** `login`
- **Respuestas:**
  - **201 Created:** Inicio de sesión exitoso en formato `JwtUserResponse`.
  - **400 Bad Request:** Los datos introducidos no son válidos.

Ejemplo de cuerpo de solicitud:

```
{
    "username": "rducker0",
    "password": "password"
}
```

-

POST `/userLogout`

- **Descripción:** Cierra la sesión de un usuario invalidando su token.
- **Operación:** `logout`
- **Respuestas:**
    - **200 OK:** Usuario desconectado exitosamente.
    - **400 Bad Request:** Token inválido o token faltante.

UserAdminController

Descripción: Este endpoint permite obtener todos los usuarios o buscar usuarios por criterios específicos.

- **URL:** `/admin/user/allusers`
- **Método HTTP:** GET

Parámetros de consulta

- `search` (opcional): Criterio de búsqueda para usuarios.
- `c` (opcional, valor por defecto: 100): Número de usuarios por página.
- `p` (opcional, valor por defecto: 0): Número de página.

Respuestas

- **Código de respuesta 200 OK**
  `[`

```
  {
    "id": 1234567890,
    "username": "krobert152",
    "email": "robertorebolledo152@gail.com",
    "password": "Lagarto_Wapo32",
    "roles": ["ADMIN", "USER", "MANOLO"],
    "profilePhoto": "fotowapa"
  }
]
```

    -

- **Código de respuesta 400 Bad Request**

```
{
  "error": "Bad Request",
  "message": "The user data provided is invalid. Please check the request body for errors."
}
```

- **Código de respuesta 500 Internal Server Error**
  {

```
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try again later."
}
```

2. Crear un nuevo usuario

Descripción: Este endpoint permite crear un nuevo usuario.

- **URL:** `/admin/user/`
- **Método HTTP:** POST

Cuerpo de la solicitud

Ejemplo de cuerpo de solicitud:
json
Copiar código

```
{
  "username": "krobert152",
  "email": "robertorebolledo152@gail.com",
  "password": "Lagarto_Wapo32",
  "roles": ["ADMIN", "USER", "MANOLO"],
  "profilePhoto": "fotowapa"
}
```

Respuestas

- **Código de respuesta 201 Created**
  {

```
  "username": "krobert152",
  "email": "robertorebolledo152@gail.com",
  "password": "Lagarto_Wapo32",
  "roles": ["ADMIN", "USER", "MANOLO"],
  "profilePhoto": "fotowapa"
}
```

- **Código de respuesta 400 Bad Request**

```
{
  "error": "Bad Request",
  "message": "The user data provided is invalid. Please check the request body for errors."
}
```

- **Código de respuesta 500 Internal Server Error**

```
{
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try again later."
}
```

3. Obtener detalles de usuario por ID

Descripción: Este endpoint permite obtener los detalles de un usuario específico por su ID.

- **URL:** `/admin/user/{id}`
- **Método HTTP:** GET

Parámetros de ruta

- `{id}`: ID del usuario a recuperar.

Respuestas

- **Código de respuesta 200 OK**

```
{
  "id": "1234567890",
  "username": "john_doe",
```

```
  "email": "john@example.com",
  "profilePhoto": "http://example.com/photos/john.jpg",
  "roles": ["USER", "ADMIN"],
  "encounters": [...],
  "articles": [...],
  "savedLists": [...],
  "level": 5,
  "exp": 100,
  "percentExp": 50,
  "accountNonExpired": true,
  "accountNonLocked": true,
  "credentialsNonExpired": true,
  "enabled": true,
  "createdAt": "2024-06-11",
  "old": "false",
  "passwordExpiredAt": "2025-06-11"
}
```


- **Código de respuesta 404 Not Found**


```
{
  "error": "Not Found",
  "message": "The user with the specified ID was not found."
}
```


- **Código de respuesta 500 Internal Server Error**


```
{
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try again later."
}
```


4. Actualizar información básica del usuario por ID

Descripción: Este endpoint permite actualizar la información básica de un usuario por su ID.

- **URL:** `/admin/user/update/{id}`

- **Método HTTP:** PUT

Parámetros de ruta

- `{id}`: ID del usuario a actualizar.

Cuerpo de la solicitud

```
{
  "username": "john_doe",
  "email": "john@example.com",
  "roles": ["USER"],
  "level": 5,
  "profilePhoto": "http://example.com/photos/john.jpg"
}
```

-

Respuestas

- **Código de respuesta 201 Created**

```
{
  "id": "1",
  "username": "john_doe",
  "email": "john@example.com",
  "roles": ["USER"],
  "level": 5,
  "profilePhoto": "http://example.com/photos/john.jpg"
}
```

    -
- **Código de respuesta 400 Bad Request**

Copiar código

```
{
  "error": "Bad Request",
  "message": "The provided user data is invalid."
}
```

    -
- **Código de respuesta 404 Not Found**

```
{
  "error": "Not Found",
  "message": "The user with the specified ID was not found."
}
```

- **Código de respuesta 500 Internal Server Error**

```
{
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try
again later."
}
```

5. Actualizar permisos de usuario por ID

Descripción: Este endpoint permite actualizar los permisos de un usuario por su ID.

- **URL:** `/admin/user/update/permissions/{id}`
- **Método HTTP:** PUT

Parámetros de ruta

- `{id}`: ID del usuario cuyos permisos se van a actualizar.

Cuerpo de la solicitud

```
{
  "username": "john_doe",
  "email": "john@example.com",
  "accountNonExpired": true,
  "accountNonLocked": true,
  "credentialsNonExpired": true,
  "enabled": true
}
```

Respuestas

- **Código de respuesta 201 Created**

```
{
  "username": "john_doe",
  "email": "john@example.com",
  "accountNonExpired": true,
  "accountNonLocked": true,
  "credentialsNonExpired": true,
  "enabled": true
}
```


- **Código de respuesta 400 Bad Request**


```
{
  "error": "Bad Request",
  "message": "The provided permissions data is invalid."
}
```


- **Código de respuesta 404 Not Found**


```
{
  "error": "Not Found",
  "message": "The user with the specified ID was not found."
}
```


- **Código de respuesta 500 Internal Server Error**


```
{
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try again later."
}
```


6. Actualizar roles de usuario por ID

Descripción: Este endpoint permite actualizar los roles de un usuario por su ID.

- **URL:** `/admin/user/update/roles/{id}`
- **Método HTTP:** PUT

Parámetros de ruta

- `{id}`: ID del usuario cuyos roles se van a actualizar.

Cuerpo de la solicitud

```
{
  "roles": ["USER", "ADMIN"]
}
```

- **Código de respuesta 400 Bad Request**

```
{
  "error": "Bad Request",
  "message": "The provided roles data is invalid."
}
```

- **Código de respuesta 404 Not Found**

```
{
  "error": "Not Found",
  "message": "The user with the specified ID was not found."
}
```

- **Código de respuesta 500 Internal Server Error**

```
{
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try
again later."
}
```

7. Eliminar usuario por ID

Descripción: Este endpoint permite eliminar un usuario por su ID.

- **URL:** `/admin/user/delete/{id}`
- **Método HTTP:** DELETE

Parámetros de ruta

- `{id}`: ID del usuario a eliminar.

Respuestas

- **Código de respuesta 202 Accepted**
    - La eliminación se realiza correctamente.
- **Código de respuesta 404 Not Found**

```
{
  "error": "Not Found",
  "message": "The user with the specified ID was not found."
}
```

    -

8. Obtener detalles del usuario por ID

Descripción: Este endpoint permite obtener los detalles completos de un usuario por su ID.

- **URL:** `/admin/user/{id}`
- **Método HTTP:** GET

Parámetros de ruta

- `{id}`: ID del usuario del cual se desean obtener los detalles.

Respuestas

- **Código de respuesta 200 OK**

```
{
  "id": "1234567890",
  "username": "john_doe",
  "email": "john@example.com",
  "profilePhoto": "http://example.com/photos/john.jpg",
  "roles": ["USER", "ADMIN"],
  "encounters": [
      {
          "id": "encounter123",
          "name": "SpeciesName_2024-06-11"
      },
      {
          "id": "encounter456",
```

```
            "name": "AnotherSpeciesName_2024-06-12"
        }
    ],
    "articles": [
        {
            "id": "article123",
            "name": "Article 1"
        },
        {
            "id": "article456",
            "name": "Article 2"
        }
    ],
    "savedLists": [
        {
            "id": "list123",
            "name": "Saved List 1"
        },
        {
            "id": "list456",
            "name": "Saved List 2"
        }
    ],
    "level": 5,
    "exp": 100,
    "percentExp": 50,
    "accountNonExpired": true,
    "accountNonLocked": true,
    "credentialsNonExpired": true,
    "enabled": true,
    "createdAt": "2024-06-11",
    "old": "false",
    "passwordExpiredAt": "2025-06-11"
}
```


- **Código de respuesta 404 Not Found**


```
{
    "error": "Not Found",
    "message": "The user with the specified ID was not found."
```

```
}
```

- **Código de respuesta 500 Internal Server Error**

Ejemplo de cuerpo de respuesta:
json
Copiar código

```
{
  "error": "Internal Server Error",
  "message": "An unexpected error occurred on the server. Please try again later."
}
```