EcoLATAM API v1 — Especificación para Frontend (MVP Node/Express)

Base Path: `/api/v1`

Formato de respuestas (envelope estándar)

- **Éxito**: `{ error: false, status: <HTTP_CODE>, body: <payload> }`
- **Error**: `{ error: true, status: <HTTP_CODE>, body: <mensaje> }`

Paginación (en todos los `GET /` de listado): `?limit=<int>` (default 10), `?offset=<int>` (default 0)

Rate limiting: `POST /auth/login` — 5 intentos por minuto.

Autenticación y Tokens

- **Login** → `POST /auth/login`
- **Body**: `{ username: string, password: string }`
- **200** → `{ token: string, data: <datosUsuario>, id: number }`
- Límite: 5 req/min.
- **Portar token**: `Authorization: Bearer <JWT>`
- **Validación interna**:
- El token se verifica con `HS256` (por defecto), `audience` / `issuer` si están configurados.
- El **payload** contiene al menos `id` (ID del usuario). El middleware compara `req.user.id` con el `id` esperado en el **body** según el módulo para autorizar.
- **Patrones de middleware** (según módulo):
- `checkAuth` (usuario dueño: compara `req.user.id` con `body.id`)
- `checkOwner` (dueño del recurso — p. ej. dueño del negocio)
- `checkAdmin` (requiere que el `id` pertenezca a tabla `admin`)
- `checkSAdmin` (súper admin)
- `checkKYCUser` / `checkKYCBusiness` (KYC aprobado)
- `checkVolunteer`, `checkInspector`, `checkTGuide`, `checkCSAgent` (rol específico)

Nota importante: Algunos módulos esperan el **ID del actor** en `body.id` para autorizar (y, en otros, `added_by`). **No confundir con el ID del recurso**. Cuando un endpoint requiere `checkAdmin()` y el controlador usa `id` del `body` como campo del recurso, envía además `added_by` (actor). Ver secciones por módulo.

Convenciones de estado HTTP

- `200 OK` → lecturas / eliminaciones exitosas.
- `201 Created` → creaciones y **también** actualizaciones/aprobaciones (según implementación actual).
- `400 Bad Request` → validación/paginación inválida.
- `401 Unauthorized` → token faltante/rol insuficiente/ID actor no coincide.
- `404 Not Found` → rutas inexistentes.
- `500 Internal Server Error` → errores no controlados.

Módulos y Endpoints

A continuación, para cada módulo: **ruta base**, **endpoints**, **seguridad**, **body esperado** y **respuesta**. Todos los listados aceptan `?limit` y `?offset`.

> **Atajo de lectura**: los módulos de catálogo (`pais`, `provincia`, `canton`, `distrito`, `idtype`, `bcategory`) comparten el patrón CRUD con `checkAdmin` en creación/edición/borrado. Los módulos de *solicitudes/aprobaciones* (`ukyc`, `bkyc`, `inspector`, `vip`, `volunteer`, `tour_guide`, `regel`, `product`, `service`) usan **POST** para "solicitar", **PUT** para "aprobar" (admin) y **DELETE** para "rechazar" (admin).

Auth

`POST /auth/login`

- **Body**: `{ username, password }`
- **200** → `{ token, data, id }`

Users (`/users`) — tabla: `users`

`GET /users`

- **200** → `[{...}]`

`GET /users/:id`

- **200** → `{...}`

`POST /users`

- **Body requerido** (creación de perfil):

```
{
  "id": number?,
  "name": "string",
  "lastname": "string",
  "birthdate": "YYYY-MM-DD",
  "location": "string",
  "id_pais": number,
  "id_provincia": number,
  "id_canton": number,
  "id_distrito": number,
  "zip": "string",
  "cellphone": "string",
  "phone": "string",
  "username": "string?",    // opcional: para crear credenciales
  "password": "string?",    // opcional: idem
  "email": "string?"        // opcional: idem
}
```

- **201** → `"Item guardado con éxito"`

  Si `username/password/email` vienen en el body, se crea/actualiza el registro en `auth` (hash bcrypt) asociado a este `user.id`.

`PUT /users`

- **Seguridad**: `checkAuth()` → `Authorization: Bearer` **y** `body.id` debe ser el **ID del usuario logueado**
- **Body** (parcial):

```
{
  "id": number,              // actor y dueño
  "name": "string?",
  "lastname": "string?",
  "birthdate": "YYYY-MM-DD?",
  "location": "string?",
  "id_pais": number?,
  "id_provincia": number?,
  "id_canton": number?,
  "id_distrito": number?,
  "zip": "string?",
  "cellphone": "string?",
  "phone": "string?"
}
```

- **201** → `"Item actualizado con éxito"`

`DELETE /users`

- **Seguridad**: `checkAuth()` (mismo criterio que PUT)
- **Body**: `{ id: number }`
- **200** → `"Item eliminado satisfactoriamente"`

Business (`/business`) — tabla: `business`

`GET /business` / `GET /business/:id`

- **200**

`POST /business`

- **Seguridad**: `checkKYCUser()` (el usuario del token **con KYC aprobado** y cuyo `id` coincide con `body.id_user`)
- **Body**:

```
{
  "id": number?,
  "id_user": number,       // dueño
  "name": "string",
  "location": "string",
  "id_pais": number,
  "id_provincia": number,
  "id_canton": number,
  "id_distrito": number,
  "zip": "string",
  "id_bcategory": number,
  "phone": "string"
}
```

- **201** → `"Item guardado con éxito"`

`PUT /business`

- **Seguridad**: `checkOwner()` (token del dueño del negocio)
- **Body**: mismo shape que creación + `id` del negocio
- **201** → `"Item actualizado con éxito"`

`DELETE /business`

- **Seguridad**: `checkOwner()`
- **Body**: `{ id: number }`
- **200** → `"Item eliminado satisfactoriamente"`

Catálogos (requiere Admin en mutaciones)

> Todas las lecturas (`GET`) son públicas; **POST/PUT/DELETE** exigen `checkAdmin()` con **ID de actor** en el body (o `added_by` según el módulo).

Países (`/pais`) — tabla: `pais`

- **POST** `{ id?, name, added_by }` → **201**
- **PUT** `{ id, name }` → **201**
- **DELETE** `{ id }` → **200**

Provincias (`/provincia`) — tabla: `provincia`

- **POST** `{ id?, id_pais, name, added_by }` → **201**
- **PUT** `{ id, name }` → **201**
- **DELETE** `{ id }` → **200**

Cantones (`/canton`) — tabla: `canton`

- **POST** `{ id?, id_provincia, name, added_by }` → **201**
- **PUT** `{ id, name }` → **201**
- **DELETE** `{ id }` → **200**

Distritos (`/distrito`) — tabla: `distrito`

- **POST** `{ id?, id_canton, name, added_by }` → **201**
- **PUT** `{ id, name }` → **201**
- **DELETE** `{ id }` → **200**

Tipos de ID (`/idtype`) — tabla: `idtype`

- **POST** `{ id?, name, description, added_by }` → **201**
- **PUT** `{ id, name?, description? }` → **201**
- **DELETE** `{ id }` → **200**

Categorías de negocio (`/bcategory`) — tabla: `bcategory`

- **POST** `{ id?, name, description, added_by }` → **201**
- **PUT** `{ id, name?, description? }` → **201**
- **DELETE** `{ id }` → **200**

KYC de Usuario (`/ukyc`) — tabla: `ukyc`

- **GET /** — **Solo Admin** (`checkAdmin()`)

- **GET /****:id** — público (detalle por ID)
- **POST /** — `checkAuth()` (actor = `body.id` y `id_user`)

```
{
  "id": number?,
  "id_user": number,
  "id_idtype": number,
  "identity": "string",    // número/doc
  "pictures": "string"     // evidencia
}
```

201 → creado (solicitud enviada)

- **PUT /** — `checkAdmin()` aprobar

```
{ "id": number, "approved_by": number }
```

201 → aprobado (`approve: true`)

- **DELETE /** — `checkAdmin()` rechazar

```
{ "id": number }
```

200 → eliminado

KYC de Negocio (`/bkyc`) — tabla: `bkyc`

- Igual a `ukyc`, pero con `id_business` en vez de `id_user`.
- **POST** `{ id?, id_business, id_idtype, identity, pictures }`
- **PUT** `{ id, approved_by }` (admin)
- **DELETE** `{ id }` (admin)

Roles & Staff

Admins (`/admin`) — tabla: `admin`

- **GET /**, **GET /****:id** — públicos
- **POST /** — `security()` = `checkSAdmin()`

```
{ "id": number?, "id_user": number, "added_by": number }
```

201

- **DELETE /** — *sin middleware explícito* (⚠️ revisar), body `{ id: number }`, **200**

Superadmins (`/superadmin`) — tabla: `superadmin`

- **GET /**, **GET /****:id** — públicos
- **POST /** — `checkAdmin()`

```
{ "id": number?, "id_user": number, "added_by": number }
```

201

- **DELETE /** — `checkAdmin()` → `{ id }` → **200**

CS Agents (`/csagent`) — tabla: `csagent`

- **GET /**, **GET /****:id** — públicos
- **POST /** — `checkAdmin()` `{ id?, id_user, added_by }` → **201**
- **DELETE /** — `checkAdmin()` `{ id }` → **200**

Inspectores (`/inspector`) — tabla: `inspector`

- **POST /** — `checkAuth()` `{ id?, id_user }` (solicitud)
- **PUT /** — `checkAdmin()` `{ id, approved_by }` (aprobación) → **201**
- **DELETE /** — `checkAdmin()` `{ id }` (rechazo) → **200**

VIP (`/vip`) — tabla: `vip`

- Patrón igual a `inspector` (`checkAuth` → solicitar; `checkAdmin` → aprobar/rechazar)

Voluntarios (`/volunteer`) — tabla: `volunteer`

- Patrón igual a `inspector`.

Guías de Tour (`/tour_guide`) — tabla: `tour_guide`

- Patrón igual a `inspector`.

Productos y Servicios

Productos (`/product`) — tabla: `product`

- **GET /**, **GET /****:id** — públicos
- **POST /** — `checkOwner()` (solicitar publicación)

```
{ "id": number?, "id_business": number, "name": "string", "description":
"string", "price": number, "disponibility": "string" }
```

201

- **PUT /** — `checkAdmin()` (aprobar)

```
{ "id": number, "approved_by": number }
```

201

- **DELETE /** — `checkAdmin()` (rechazar)

```
{ "id": number }
```

200

Servicios (`/service`) — tabla: `service`

- Mismo patrón y body que **Productos**.

Reseñas / Reviews (públicas al crear; edit/eliminar con token)

Create abierto; **PUT/DELETE** requieren `checkAuth()` (el autor del review).

Review de Negocio (`/business_review`) — tabla: `business_review`

- **POST** `{ id?, id_business, id_user, score: number, comment: string }` → **201**
- **PUT** `{ id, score?, comment? }` → **201**
- **DELETE** `{ id }` → **200**

Review de Producto (`/product_review`) — tabla: `product_review`

- Igual a negocio, con `id_product`.

Review de Servicio (`/service_review`) — tabla: `service_review`

- Igual a negocio, con `id_service`.

Review de CS Agent (`/csagent_review`) — tabla: `csagent_review`

- Igual a negocio, con `id_csagent`.

Review de Guías (`/tour_guide_review`) — tabla: `tour_guide_review`

- Igual a negocio, con `id_tour_guide`.

Review Voluntario→Negocio (`/volunteer_review_business`)

- **POST** `{ id?, id_business, id_volunteer, score, comment }`
- **PUT/DELETE** igual patrón

Review Negocio→Voluntario (`/business_review_volunteer`)

- **POST** `{ id?, id_business, id_volunteer, score, comment }`
- **PUT/DELETE** igual patrón

Legal / Documentos de negocio (`/regel`) — tabla: `regel`

- **GET /**, **GET /****:id** — públicos
- **POST /** — `checkOwner()` (dueño del negocio)

```
{ "id": number?, "id_user": number, "id_business": number, "document":
"string" }
```

- **PUT /** — `checkAdmin()` `{ id, approved_by }` → **201**
- **DELETE /** — `checkAdmin()` `{ id }` → **200**

Clientes / Contactos (`/clients`) — tabla: `clients`

Sin auth (usa validación de Joi)

- **POST** `{ name, email }` → **201**
- **PUT** `{ id, name?, email? }` → **201**
- **DELETE** `{ id }` → **200**
- **GET /**, **GET /****:id** → **200**

Ejemplos de uso

1) Login y uso de token

```
# Login
curl -X POST
  -H "Content-Type: application/json"
  -d '{"username":"demo","password":"secret"}'
  https://<host>/api/v1/auth/login

# Usar token en requests
curl -H "Authorization: Bearer <JWT>"
  https://<host>/api/v1/business?limit=20&offset=0
```

2) Crear usuario + credenciales

```
curl -X POST
  -H "Content-Type: application/json"
  -d '{
    "name":"Ana",
    "lastname":"Solano",
    "birthdate":"1995-06-10",
    "location":"CR",
    "id_pais":1, "id_provincia":1, "id_canton":1, "id_distrito":1,
    "zip":"11001",
    "cellphone":"+506 8888-0000",
    "phone":"+506 2222-0000",
    "username":"ana",
    "password":"S3guro!",
    "email":"ana@example.com"
  }'
  https://<host>/api/v1/users
```

3) Crear negocio (KYC requerido)

```
curl -X POST
  -H "Authorization: Bearer <JWT>"
  -H "Content-Type: application/json"
  -d '{
    "id_user": 42,
    "name": "Finca Verde",
    "location": "Quepos",
    "id_pais":1, "id_provincia":1, "id_canton":1, "id_distrito":1,
    "zip":"60601",
    "id_bcategory": 3,
    "phone": "+506 2777-0000"
  }'
  https://<host>/api/v1/business
```

4) Solicitar KYC de usuario

```
curl -X POST
  -H "Authorization: Bearer <JWT>"
  -H "Content-Type: application/json"
  -d '{
    "id": 42,                  // actor (debe coincidir con token)
    "id_user": 42,             // mismo usuario
    "id_idtype": 1,
```

```
    "identity": "1-1234-5678",
    "pictures": "ipfs://..."
  }'
  https://<host>/api/v1/ukyc
```

Observaciones y notas de integración

- **ID del actor**: En endpoints protegidos por `checkAdmin()` o similares, el middleware espera un **ID en el body** para cotejar contra el JWT. En módulos de *catálogo* la práctica recomendada es enviar `added_by` como actor y mantener `id` para el recurso; sin embargo, hay módulos que usan `id` también para el check. **Recomendación**: Enviar **ambos** cuando proceda (p. ej., `{ id: <recurso>, added_by: <actor> }`).
- **Fechas**: El backend formatea fechas en **UTC** (`YYYY-MM-DD`).
- **Validación**: Donde hay `validator.js` (Joi), las reglas se aplican en `POST/PUT`. Si falta validator en el módulo, el controlador acepta los campos indicados en esta guía.
- **Aprobaciones**: Todos los flujos `request/approve/decline` fijan `approved_by` y `approved_at` en aprobación; `DELETE` elimina el registro (rechazo).
- **Reviews**: La creación no exige token, pero **editar/borrar** sí (el `id` del autor debe coincidir con el token).
