5. 📚 Documentación de la API

Esta documentación sigue el estilo **RPC (Remote Procedure Call)** donde los endpoints tienen nombres de acciones o funciones, no de recursos. Cada endpoint está documentado con la siguiente estructura de 8 puntos obligatorios:

5.1. Autenticación

Endpoint: /api/register

1. **Endpoint:** /api/register
2. **Método HTTP:** POST
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:**

```
{
  "username": "juan_perez",
  "email": "juan@example.com",
  "password": "miPassword123"
}
```

7. **Estructura de datos de salida (Éxito):**

```
{
  "id": "507f1f77bcf86cd799439011",
  "username": "juan_perez",
  "email": "juan@example.com",
  "message": "Usuario registrado exitosamente."
}
```

8. **Estructura de datos de salida (Error):**

```
{
  "type": "error",
  "description": "Todos los campos son obligatorios."
}
```

o

```
{
  "type": "error",
```

```
  "description": "Este email o nombre de usuario ya está registrado."
}
```

Endpoint: /api/login

1. **Endpoint:** /api/login
2. **Método HTTP:** POST
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:**

```
{
  "email": "juan@example.com",
  "password": "miPassword123"
}
```

7. **Estructura de datos de salida (Éxito):**

```
{
  "id": "507f1f77bcf86cd799439011",
  "username": "juan_perez",
  "email": "juan@example.com",
  "message": "Inicio de sesión exitoso."
}
```

8. **Estructura de datos de salida (Error):**

```
{
  "type": "error",
  "description": "Email y contraseña son obligatorios."
}
```

o

```
{
  "type": "error",
  "description": "Credenciales inválidas."
}
```

5.2. Perfil de Usuario

Endpoint: /api/profile

1. **Endpoint:** /api/profile
2. **Método HTTP:** GET
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:** - (Los datos se envían mediante query parameter userId)
    - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011
7. **Estructura de datos de salida (Éxito):**


```
{
  "diet_preference": "vegetarian",
  "maxCalories": 2000,
  "maxCarbs": 250,
  "maxProtein": 120,
  "maxSugar": 40
}
```


8. **Estructura de datos de salida (Error):**


```
{
  "type": "error",
  "description": "Falta el ID del usuario."
}
```


o


```
{
  "type": "error",
  "description": "Perfil no encontrado."
}
```


Endpoint: /api/profile (Actualizar)

1. **Endpoint:** /api/profile
2. **Método HTTP:** PUT
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:**
    - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011

- Body:


```
{
  "diet_preference": "vegan",
  "maxCalories": 1800,
  "maxCarbs": 200,
  "maxProtein": 100,
  "maxSugar": 35
}
```


> Nota: Todos los campos son opcionales. Se pueden actualizar uno o varios campos.

7. **Estructura de datos de salida (Éxito):**


```
{
  "diet_preference": "vegan",
  "maxCalories": 1800,
  "maxCarbs": 200,
  "maxProtein": 100,
  "maxSugar": 35
}
```


8. **Estructura de datos de salida (Error):**


```
{
  "type": "error",
  "description": "No se proporcionaron campos válidos para actualizar o el perfil no se pudo actualizar."
}
```


5.3. Inventario

Endpoint: /api/inventario (Listar)

1. **Endpoint:** /api/inventario
2. **Método HTTP:** GET
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:** - (Los datos se envían mediante query parameter userId)
   - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011

7. **Estructura de datos de salida (Éxito):**


```
[
 {
  "_id": "507f1f77bcf86cd799439012",
  "user": "507f1f77bcf86cd799439011",
  "article_name": "pan",
  "quantity": 500,
  "unit": "gramos",
  "createdAt": "2024-01-15T10:30:00.000Z",
  "updatedAt": "2024-01-15T10:30:00.000Z"
 },
 {
  "_id": "507f1f77bcf86cd799439013",
  "user": "507f1f77bcf86cd799439011",
  "article_name": "leche",
  "quantity": 2,
  "unit": "litros",
  "createdAt": "2024-01-15T11:00:00.000Z",
  "updatedAt": "2024-01-15T11:00:00.000Z"
 }
]
```


8. **Estructura de datos de salida (Error):**


```
{
 "type": "error",
 "description": "Error interno al listar inventario."
}
```


Endpoint: /api/inventario (Crear)

1. **Endpoint:** /api/inventario
2. **Método HTTP:** POST
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:**
    - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011
    - Body:


```
{
 "article_name": "huevos",
```

```
  "quantity": 12,
  "unit": "unidades"
}
```

> Nota: El campo unit debe ser uno de: "gramos", "kilogramos", "unidades", "litros", "mililitros". Si el alimento ya existe, se sumará la cantidad nueva a la existente.

7. **Estructura de datos de salida (Éxito):**

```
{
  "_id": "507f1f77bcf86cd799439014",
  "user": "507f1f77bcf86cd799439011",
  "article_name": "huevos",
  "quantity": 12,
  "unit": "unidades",
  "createdAt": "2024-01-15T12:00:00.000Z",
  "updatedAt": "2024-01-15T12:00:00.000Z"
}
```

8. **Estructura de datos de salida (Error):**

```
{
  "type": "error",
  "description": "Todos los campos son obligatorios."
}
```

o

```
{
  "type": "error",
  "description": "Error interno al agregar alimento."
}
```

Endpoint: /api/inventario/check

1. **Endpoint:** /api/inventario/check
2. **Método HTTP:** GET
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:** - (Los datos se envían mediante query parameters)
    - Query Parameters requeridos:

        - ?userId=507f1f77bcf86cd799439011
        - &name=pan
7. **Estructura de datos de salida (Éxito - Alimento existe):**


```
{
  "exists": true,
  "id": "507f1f77bcf86cd799439012",
  "quantity": 500,
  "unit": "gramos"
}
```


Estructura de datos de salida (Éxito - Alimento no existe):


```
{
  "exists": false
}
```


8. **Estructura de datos de salida (Error):**


```
{
  "type": "error",
  "description": "Falta el nombre del alimento para chequear."
}
```


o


```
{
  "type": "error",
  "description": "Error interno al chequear duplicado."
}
```


Endpoint: /api/inventario/:alimentoId/sumar

1. **Endpoint:** /api/inventario/:alimentoId/sumar
2. **Método HTTP:** PATCH
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:**
    - URL Parameter: :alimentoId (ej: 507f1f77bcf86cd799439012)
    - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011

- Body:

```
{
  "quantity": 100
}
```

7. **Estructura de datos de salida (Éxito):**

```
{
  "message": "Cantidad sumada con éxito."
}
```

8. **Estructura de datos de salida (Error):**

```
{
  "type": "error",
  "description": "Falta la cantidad a sumar."
}
```

o

```
{
  "type": "error",
  "description": "Alimento no encontrado o no pertenece al usuario."
}
```

Endpoint: /api/inventario/:alimentoId (Actualizar)

1. **Endpoint:** /api/inventario/:alimentoId
2. **Método HTTP:** PUT
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:**
   - URL Parameter: :alimentoId (ej: 507f1f77bcf86cd799439012)
   - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011
   - Body (todos los campos son opcionales):

```
{
  "article_name": "pan integral",
  "quantity": 600,
  "unit": "gramos"
}
```

Nota: Se pueden actualizar uno, varios o todos los campos. Al menos un campo debe estar presente.

7. **Estructura de datos de salida (Éxito):**

```
{
  "message": "Alimento actualizado con éxito."
}
```

8. **Estructura de datos de salida (Error):**

```
{
  "type": "error",
  "description": "No se proporcionaron campos para actualizar."
}
```

o

```
{
  "type": "error",
  "description": "Alimento no encontrado, no autorizado o sin cambios."
}
```

Endpoint: /api/inventario/:alimentoId (Eliminar)

1. **Endpoint:** /api/inventario/:alimentoId
2. **Método HTTP:** DELETE
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:** - (Los datos se envían mediante URL y query parameters)
    - URL Parameter: :alimentoId (ej: 507f1f77bcf86cd799439012)
    - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011
7. **Estructura de datos de salida (Éxito):**

```
{
  "message": "Alimento eliminado con éxito."
}
```

8. **Estructura de datos de salida (Error):**


```
{
  "type": "error",
  "description": "Alimento no encontrado o no autorizado."
}
```


o


```
{
  "type": "error",
  "description": "Error interno al eliminar alimento."
}
```


5.4. Recetas

Endpoint: /api/recetas/inventario

1. **Endpoint:** /api/recetas/inventario
2. **Método HTTP:** GET
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:** - (Los datos se envían mediante query parameters)
    - Query Parameter requerido: ?userId=507f1f77bcf86cd799439011
    - Query Parameters opcionales:
        - &diet=vegetarian (opciones: vegetarian, vegan, gluten free, none)
        - &maxCalories=2000
        - &maxCarbs=250
        - &maxProtein=120
        - &maxSugar=40
7. **Estructura de datos de salida (Éxito):**


```
[
  {
    "id": 654959,
    "title": "Tortilla de Patatas",
    "image":
"[https://spoonacular.com/recipeImages/654959-312x231.jpg](https://spoonacular.com/recipeI
mages/654959-312x231.jpg)",
    "missedIngredients": [
      {
        "id": 11215,
```

```
        "amount": 2.0,
        "unit": "cloves",
        "unitLong": "cloves",
        "unitShort": "cloves",
        "aisle": "Produce",
        "name": "garlic",
        "original": "2 dientes de ajo",
        "originalName": "garlic",
        "meta": [],
        "extendedName": null,
        "image":
"[https://spoonacular.com/cdn/ingredients_100x100/garlic.png](https://spoonacular.com/cdn/ingredients_100x100/garlic.png)"
      }
    ],
    "usedIngredients": [
      {
        "id": 11297,
        "amount": 4.0,
        "unit": "large",
        "unitLong": "larges",
        "unitShort": "large",
        "aisle": "Produce",
        "name": "eggs",
        "original": "4 huevos grandes",
        "originalName": "eggs",
        "meta": [],
        "extendedName": null,
        "image":
"[https://spoonacular.com/cdn/ingredients_100x100/egg.png](https://spoonacular.com/cdn/ingredients_100x100/egg.png)"
      }
    ]
  }
]
```


Nota: Si no hay ingredientes en el inventario, se devuelve un array vacío [].

8. **Estructura de datos de salida (Error):**


```
{
  "type": "error",
  "description": "Error interno al buscar recetas."
```

}

Endpoint: /api/recetas/detalles/:recipeId

1. **Endpoint:** /api/recetas/detalles/:recipeId
2. **Método HTTP:** GET
3. **Formato de serialización:** JSON
4. **Cabecera de entrada:** -
5. **Cabecera de salida:** -
6. **Estructura de datos de entrada:** - (Los datos se envían mediante URL parameter)
    - URL Parameter: :recipeId (ej: 654959)
7. **Estructura de datos de salida (Éxito):**


```
{
  "id": 654959,
  "title": "Tortilla de Patatas",
  "summary": "Esta es una deliciosa receta de tortilla española...",
  "instructions": "1. Pelar y cortar las patatas...",
  "image":
"[https://spoonacular.com/recipeImages/654959-556x370.jpg](https://spoonacular.com/recipe
Images/654959-556x370.jpg)",
  "extendedIngredients": [
    {
      "id": 11297,
      "amount": 4.0,
      "unit": "large",
      "unitLong": "larges",
      "unitShort": "large",
      "aisle": "Produce",
      "name": "eggs",
      "original": "4 huevos grandes",
      "originalName": "eggs",
      "meta": [],
      "extendedName": null,
      "image":
"[https://spoonacular.com/cdn/ingredients_100x100/egg.png](https://spoonacular.com/cdn/in
gredients_100x100/egg.png)"
    }
  ],
  "readyInMinutes": 45,
  "servings": 4,
  "sourceUrl": "https://..."
}
```

Nota: Todos los textos (título, resumen, instrucciones, ingredientes) están traducidos al español.

8. **Estructura de datos de salida (Error):**

```
{
  "type": "error",
  "description": "Error interno del servidor al buscar detalles de receta."
}
```

o

```
{
  "type": "error",
  "description": "Error de la API externa al obtener detalles."
}
```

Notas Importantes sobre la Documentación

- **Filosofía RPC:** Todos los endpoints siguen el estilo RPC con nombres de acciones (ej: /api/login, /api/register, /api/recetas/inventario).
- **Autenticación:** Los endpoints protegidos requieren el query parameter userId que debe ser un ID válido de usuario en la base de datos.
- **Formato de Error:** Todos los errores deberían seguir la estructura: { "type": "error", "description": "..." }. Actualmente algunos endpoints pueden devolver { "error": "..." }, pero unificaremos al formato estándar.
- **Formato de Serialización:** Todos los endpoints utilizan JSON tanto para entrada como para salida.
- **Cabeceras:** Actualmente no se utilizan cabeceras especiales (ni de entrada ni de salida). La autenticación se maneja mediante query parameters.