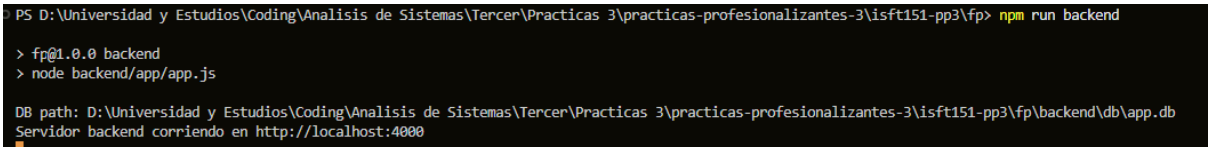
# Tres 3 Dos
# Documentación de la WebAPI

Base URL: http://localhost:4000
Autenticación: Token de sesión (bycript hash method).
Versión: 1.0.0




POST /users/register → Metodo que registra usuario
Request:
{
  "username": "Leo",
  "password": "Leo!", → Esto se hashea, por lo que en la db se guarda el hash
  "organizationId": 1
}
Validaciones:

- username debe tener mínimo 3 caracteres
- password debe tener una mayúscula y un carácter especial
- si el usuario ya existe, devuelve error

Response: 201
{
  "message": "ok",
  "user": {
        "id": 5,
        "username": "Leo",
        "organizationId": 1,
        "active": 1,
        "created_at": "2025-11-03T 18:00:00Z"
        }
}
Response: 400
{ "error": "Username already exists" }

POST /users/login → Inicia sesión y devuelve token de inicio
{
  "username": "Leo",
  "password": "hash de password"
}
Response: 200
{
  "message": "ok",
  "token": "session-<token>",

```
  "role": "usuario", → Al registrar, por defecto son usuarios
  "organizations": [
    { "id": 1, "name": "Tres 3 Dos" }
  ],
  "organizationId": 1
}
```

Response: 400
```
{ "error": "Invalid credentials" }
```

POST /users/logout → Cierra sesión
Request:
```
{ "token": "session-<token>" }
```

Response: 200
```
{ "message": "Logged out" }
```

Get /users → Devuelve lista de usuarios del sistema (solo permiso de administrador)
Response: 200
```
[
  {
    "id": 1,
    "username": "admin",
    "active": 1,
    "created_at": "<timestamp>",
    "role_id": 1,
    "role_name": "administrador_general",
    "organization_id": 1
  },
  {
    "id": 2,
    "username": "esorero",
    "active": 1,
    "created_at": "<timestamp>",
    "role_id": 3,
    "role_name": "tesorero",
    "organization_id": 1
  }
]
```

PUT /users/:id → actualiza datos de usuario
Request:
```
{
  "username": "Leo2",
  "password": "Leoo!"
}
```
Response 200:
```
{
```

```
  "id": 2,
  "username": "Leo2",
  "active": 1,
  "created_at": "<timestamp>"
}
```

DELETE /users/:id → Elimina usuario por ID
response: 200
```
{ "message": "User deleted" }
```

POST /users/:id/assignRole → Asigna rol de organización
Request:
```
{
  "roleId": 2,
  "organizationId": 1
}
```
Response: 200
```
{
  "message": "Rol asignado correctamente",
  "changes": 1
}
```
Response: 400
```
{ "error": "Debe indicar roleId" }
```

Organizaciones → Endpoint /organizations

GET /organizations → Obtiene todas las organizaciones registradas, solo para admin_general
Response:200
```
[
  { "id": 1, "name": "Tres 3 Dos" },
  { "id": 2, "name": "Banda musical de ciro" }
]
```

Roles disponibles en el sistema:
Get /roles → Lista los roles que hay en el sistema

Response 200:
```
[
  { "id": 1, "name": "administrador_general" },
  { "id": 2, "name": "administrador" },
  { "id": 3, "name": "tesorero" }
]
```

Oncomes: /api/incomes
POST /api/incomes → incomeController.create → Registra un ingreso válido.
Request body (ejemplo):

{
"amount": 1500.00,
"description": "Venta entrada evento",
"date": "2025-11-01T10:00:00Z",
"organizationId": 1,
"eventId": 2, // opcional si corresponde
"userId": 5 // usuario que registró el ingreso
}

Response 201:
{
"id": 12,
"amount": 1500.00,
"description": "Venta entrada evento",
"date": "2025-11-01T10:00:00Z",
"organizationId": 1,
"eventId": 2,
"userId": 5
}

GET /api/incomes → incomeController.findAll → Lista todos los ingresos.
Query params sugeridos:

page, limit, startDate, endDate, organizationId, eventId
Response 200:
[
{ "id": 10, "amount": 200.0, "description": "Donación", "date": "2025-10-20T12:00:00Z",
"organizationId":1 },
{ "id": 11, "amount": 50.0, "description": "Venta merch", "date": "2025-10-21T18:00:00Z",
"organizationId":1 }
]

GET /api/incomes/:id → incomeController.findOne → Obtiene un ingreso por ID.
Response 200:
{ "id": 12, "amount": 1500.00, "description": "Venta entrada evento", "date":
"2025-11-01T10:00:00Z", "organizationId":1 }

DELETE /api/incomes/:id → incomeController.delete → Elimina un ingreso.
Response 204: No Content

Payments /api/payments

POST /api/payments → paymentController.create → Registra un pago asociado a una deuda.
Request body (ejemplo):
{
"debtId": 7,
"amount": 500.00,

"date": "2025-11-05T14:00:00Z",
"method": "transferencia",
"notes": "Pago parcial"
}
Response 201:
{
"id": 22,
"debtId": 7,
"amount": 500.00,
"date": "2025-11-05T14:00:00Z",
"method": "transferencia",
"notes": "Pago parcial"
}

GET /api/payments → paymentController.findAll → Lista todos los pagos registrados.
Query params sugeridos: debtId, startDate, endDate, organizationId, page, limit
Response 200:
[
{ "id": 20, "debtId": 5, "amount": 200.0, "date": "2025-10-15T10:00:00Z" },
{ "id": 21, "debtId": 6, "amount": 100.0, "date": "2025-10-20T11:30:00Z" }
]

GET /api/payments/debt/:debtId → paymentController.findByDebtId → Lista pagos de una deuda específica.
Response 200:
[
{ "id": 22, "debtId": 7, "amount": 500.0, "date": "2025-11-05T14:00:00Z" }
]

GET /api/payments/:id → paymentController.findOne → Obtiene un pago por ID.
Response 200:
{ "id": 22, "debtId": 7, "amount": 500.0, "date": "2025-11-05T14:00:00Z" }

DELETE /api/payments/:id → paymentController.delete → Elimina un pago.
Response 204: No Content

Products /api/products

GET /api/products → productController.findAll → Lista todos los productos.
Query params sugeridos: available, organizationId, category, page, limit
Response 200:
[
{ "id": 1, "name": "Remera", "price": 25.0, "stock": 100 },
{ "id": 2, "name": "CD", "price": 10.0, "stock": 50 }
]

GET /:id → productController.findOne → Obtiene un producto por ID.

Response 200:
{ "id": 1, "name": "Remera", "description": "Remera oficial", "price":25.0, "stock":100 }

POST / → productController.create → Crea un nuevo producto.

Request Body (ejemplo)
{
“name”: “papa”,
“description”: “papas fritas”,
“price”: 2000,
“initialSotck”: 300,
“organizationid”: 1
}

Response 201:

{
“id”:5,
“name”: ”papa”,
“description”: “papas fritas”,
“price”: 2000,
“initialSotck”: 300,
“organizationid”:1
}

PUT /:id → productController.update → Actualiza datos de un producto.
Request:
{
“price”: 50,
“stock”: 100,
“minStock”: 10
}
Response 200:
{ “message”: “updated” }
Response 400:
{ “error”: “product not found”}

DELETE /:id → productController.delete → Desactiva o elimina un producto.
Response 200:
{ “message”: “product deactivated” }
Response 400:
{ “error” : “product not found” }

GET /products/:id/movements → productController.getStockMovements → Obtiene movimientos de stock.

Response 200:
[

{
“id”: 5,
“product_id”: 1,
“type”: “ajuste”,
“qty”: 100,
“note”: “stock”,
“created_at”: “<timestamp>”
“user_id”: 3,
“organization_id” = 1
},
{
“id”: 12,
“type”: “ajuste2”,
“qty”: 20,
“note: “utilizo stock”
}
]
Response 400:
{ “error”: “<message>” }

Debts /api/debts
POST / → debtController.create → Crea una nueva deuda.
Request:
{
"creditor": "proveedor1",
"amount": 15000,
"due_date": "2025-02-15",
"description": "Factura pendiente"
}
Response 201
{
"id": 10,
"creditor": "proveedor1",
"amount": 15000,
"amount_paid": 0,
"due_date": "2025-02-15",
"description": "Factura pendiente",
"created_at": "<timestamp>"
}
Response 500:
{ "error": "<mensaje>" }
GET / → debtController.findAll → Lista todas las deudas.
Response 200:
[
{
"id": 3,
"creditor": "proveedor1",
"amount": 15000,

```
    "amount_paid": 5000,
    "due_date": "2025-02-15",
    "description": "Factura mensual",
    "created_at": "2025-11-10T12:00:00Z"
  }
]
```

Response 500:

```
{ "error": "<mensaje>" }
```

GET /:id → debtController.findOne → Obtiene una deuda por ID.
Response 200:

```
{
  "id": 3,
  "creditor": "proveedor1",
  "amount": 15000,
  "amount_paid": 5000,
  "due_date": "2025-02-15",
  "description": "Factura mensual",
  "created_at": "2025-11-10T12:00:00Z"
}
```

Response 500:

```
{ "error": "<mensaje>" }
```


DELETE /:id → debtController.delete → Elimina una deuda.
Response 200:

```
{ "message": "Debt deleted" }
```

Response 500:

```
{ "error": "<mensaje>" }
```


Events /api/events
GET / → eventController.list → Lista todos los eventos.
Response 200:

```
[
  {
    "id": 5,
    "name": "Festival",
    "date": "2025-03-12",
    "ticket_price": 2500,
    "active": 1,
    "organization": "Tres 3 Dos"
  }
]
```

Response 500:

```
{ “error”: “<error message>” }
```


GET /:id → eventController.getById → Obtiene un evento por ID.
Response 200:

```
{
  "id": 5,
```

```
  "name": "Festival",
  "date": "2025-03-12",
  "ticket_price": 2500,
  "active": 1,
  "organization": "Tres 3 Dos"
}
```

Response 404:
{ “error”: “not found” }

POST / → eventController.create → Crea un nuevo evento.
Request:

```
{
  "name": "Festival",
  "date": "2025-03-12",
  "ticketPrice": 2500,
  "organizationId": 1
}
```

Response 201:

```
{
  "message": "Created",
  "id": 5
}
```

Response 400
{ “error”: “Missing fields”}

PUT /:id → eventController.update → Actualiza un evento existente.
Request:

```
{
  "name": "Navidad (cambia precio)",
  "ticketPrice": 3000
}
```

Response 200:
{ “message”: “updated”}
Response 400
{ “error”: “event not found” }

DELETE /:id → eventController.remove → Desactiva un evento.
Response 200:
{ “message”: “event deactivated” }
Response 400:
{ “error”: “event not found” }

GET /:id/products → eventController.listProducts → Lista productos asignados a un evento.
Response 200:

```
[
 {
  “id”: 10,
  “product_id”: 1,
```

“name”: “papas”,
“qty”: 20,
“stock”: 80
}
]

Response 400:
{ “error”: “<message>” }

POST /:id/products → eventController.assignProduct → Asigna un producto a un evento.
Request:

{
"productId": 3,
"qty": 10
}

Response 200:
{ "message": "Producto asignado correctamente" }
Response 400:
{ "error": "Cantidad supera el stock disponible" }

PUT /:eventId/products/:productId → eventController.updateAssignedProduct → Actualiza asignación.
Request:
{ "qty": 20 }
Response:
{ "message": "Asignación actualizada" }
Response 400:
{ "error": "No se encuentra la asignacion" }

DELETE /:eventId/products/:productId → eventController.removeAssignedProduct → Quita un producto.
Response 200:
{ "message": "Producto desasignado" }
Response 400:
{ "error": "Asignación de producto no encontrada" }

GET /:eventId/printPDF
Response 200:
PDF binario
Response 404:
{ "error": "Evento no encontrado" }
Response 500:
{ "error": "Error generando PDF" }

Organizations /api/organizations

GET / → organizationController.findAll → Lista todas las organizaciones.
Response 200:
[

```
 {
   "id": 1,
   "name": "Tres 3 Dos",
   "contact": "tres3dos@mail.com",
   "budget": 0
 },
 {
   "id": 2,
   "name": "Test",
   "contact": "test@test.ar",
   "budget": 0
 }
]
```

Response 400
{ “error”: “<message>” }

POST / → organizationController.create → Crea una nueva organización.
Request:

```
{
  “name”: “Tres 3 Dos”,
  “contact”: “tres3dos@mail.com”
}
```

Response: 201


```
{
   "id": 1,
   "name": "Tres 3 Dos",
   "contact": “tres3dos@mail.com”,
},
```

Response 400
{ “error”: “Organization name required”}
GET /:id → organizationController.findOne → Obtiene una organización por ID.
Response: 200

```
 {
  "id": 1,
  "name": "Tres 3 Dos",
  "contact": "tres3dos@mail.com",
  "budget": 10000
}
```

Response 400:
{ “error”: “<message>” }

PUT /:id/budget → organizationController.updateBudget → Modifica el presupuesto.
Request:

```
{
  “delta”: 5000
}
```

Response 200:

{
  "id": 1,
  "name": "Tres 3 Dos",
  "contact": "tres3dos@mail.com",
  "budget": 10000
}
Response 400
{ “error”: “delta must be a number” }
Roles /api/roles
GET /roles → Lista todos los roles disponibles.

Response 200
[
  { "id": 1, "name": "administrador_general" },
  { "id": 2, "name": "administrador" },
  { "id": 3, "name": "tesorero" }
]
Response 400
{ “error”: “Sin permisos para ver roles” }