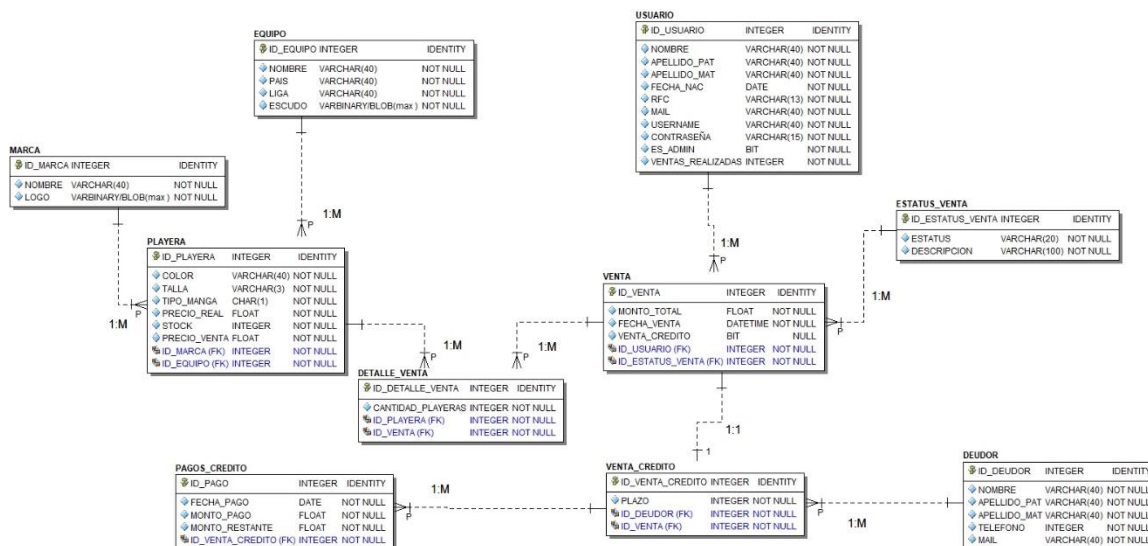


Introducción

Mi proyecto final consiste en un sistema de gestión de ventas de playeras de fútbol.

El diagrama Entidad-Relación del proyecto es el siguiente:

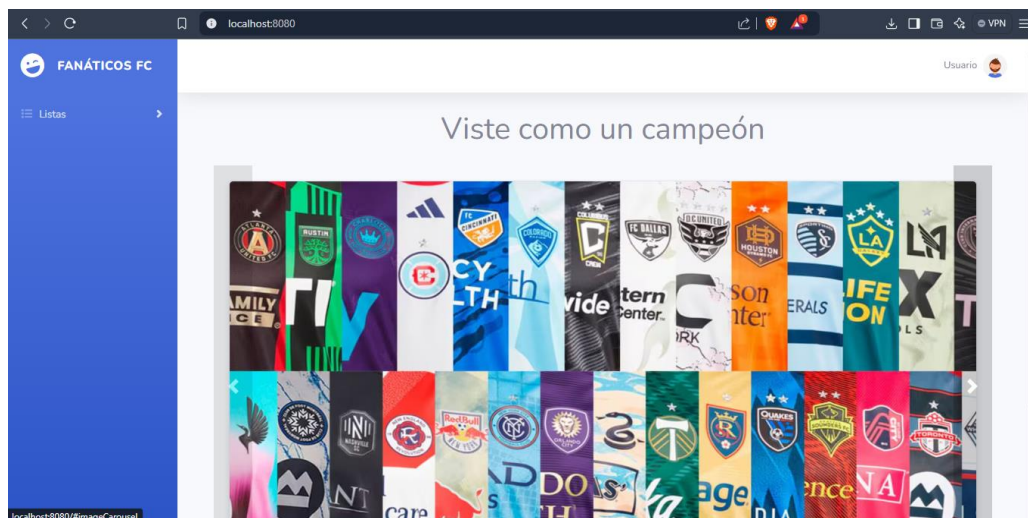


Para el avance del proyecto del módulo 10, se solicitan cuatro tablas, en las cuales se incluyen:

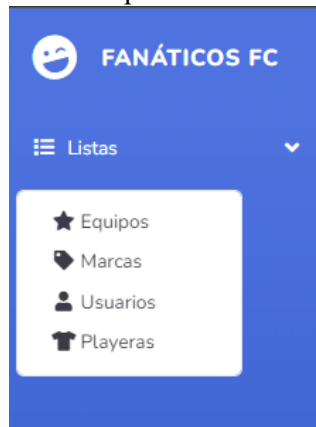
- **Tablas catálogo**
 - Equipo
 - Marca
 - Usuario
- **Tablas relacionadas**
 - Playera (depende de Marca y Equipo).

I. Capturas de pantalla del funcionamiento.

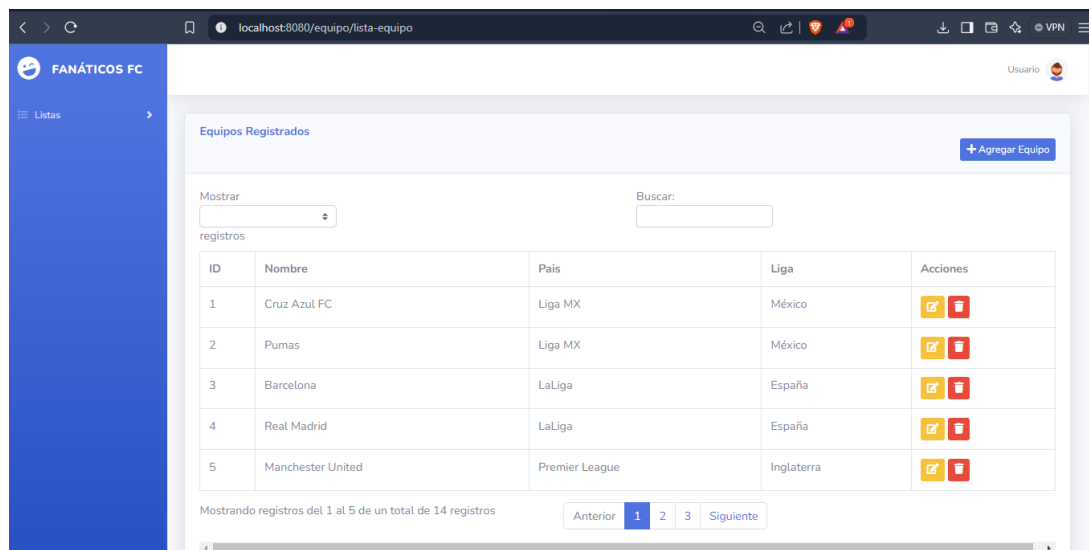
Una vez iniciada la aplicación en el navegador entramos a la siguiente URL: <http://localhost:8080> y se desplegará la siguiente página:



En la barra lateral izquierda, presionamos sobre el menú “Listas” y seleccionamos una de las opciones:



Para una demostración, seleccionamos “Equipo” y se mostrará una tabla de los equipos registrados:



Para realizar las operaciones CRUD solicitadas existen varios botones que realizan estas tareas.



En la parte superior derecha tenemos el botón “Agregar equipo” que nos redireccionará a un formulario para completar los datos y registrar un nuevo equipo en el sistema.

The screenshot shows a form titled "Agregar nuevo equipo". The form has a section "Datos del Equipo" with three input fields: "Nombre del Equipo*" (containing "Juventus"), "País*" (containing "Italia"), and "Liga*" (containing "Serie A"). At the bottom right of the form are two buttons: "Cancelar" and "Guardar Equipo".

Para el equipo, se debe agregar: nombre, país y liga a la que pertenece el equipo. Una vez llenados los datos, se oprime el botón Guardar Equipo y nos mostrará la lista con el nuevo equipo registrado.

El equipo se guardó correctamente











Equipos Registrados

+ Agregar Equipo

Mostrar

Buscar:

registros

ID	Nombre	País	Liga	Acciones
11	Feynoord	Eredivise	Holanda	 
12	Atlético de Madrid	La Liga	España	 
13	Atlas	Liga mx	México	 
14	Guadalajara	Liga mx	México	 
19	Juventus	Serie A	Italia	 

Mostrando registros del 11 al 15 de un total de 15 registros

Anterior

1

2

3

Siguiente

Es importante mencionar que se cuentan con validaciones para asegurar la consistencia de datos:

ERROR

Datos del Equipo

Nombre del Equipo*

123

Solo se permiten letras, incluyendo acentos y ñ

País*

34234

Solo se permiten letras, incluyendo acentos y ñ

Liga*

3665.

Solo se permiten letras, incluyendo acentos y ñ

Para editar un equipo existente, se selecciona el botón color amarillo el cuál nos llevará de nuevo al formulario para realizar los cambios necesarios y guardar el equipo.

Modificar Equipo

Formulario para modificar un equipo existente.


Datos del Equipo

Nombre del Equipo*

País*

Liga*

Y la última acción realiza la eliminación de un registro, el botón rojo con un bote de basura, al presionarlo nos mostrará un mensaje para confirmar la acción.

 FANÁTICOS FC

listas

localhost:8080 dice
¿Estás seguro de eliminar este equipo?











Aceptar Cancelar

Equipos Registrados

+ Agregar Equipo

Mostrar: 1 registros

Buscar:

ID	Nombre	País	Liga	Acciones
11	Feynoord	Eredivise	Holanda	 
12	Atlético de Madrid	La Liga	España	 
13	Atlas	Liga mx	México	 
14	Guadalajara	Liga mx	México	 
19	Juventus	Serie A	Italia	 

Mostrando registros del 11 al 15 de un total de 15 registros

Anterior 1 2 3 Siguiente

Al presionar aceptar, el equipo se borrará de forma permanente.

✓ El equipo se borró correctamente.









Equipos Registrados

+ Agregar Equipo

Mostrar

Buscar:

registros

ID	Nombre	Pais	Liga	Acciones
11	Feynoord	Eredivise	Holanda	 
12	Atlético de Madrid	La Liga	España	 
13	Atlas	Liga mx	México	 
14	Guadalajara	Liga mx	México	 

Mostrando registros del 11 al 14 de un total de 14 registros

Anterior

1

2

3

Siguiente

Para las tablas de “Marca” y “Usuario” se realiza el mismo procedimiento.

Tabla de Marcas











Marcas Registradas

+ Agregar Marca

Mostrar

Buscar:

registros

ID	Nombre	Acciones
1	Nike	 
2	Adidas	 
3	Puma	 
4	Under Armour	 
5	Charly	 

Mostrando registros del 1 al 5 de un total de 11 registros

Anterior

1

2

3

Siguiente

Formulario para registrar marcas

Agregar nueva marca

Datos de la Marca

Nombre*

Cancelar

Guardar Marca

Tabla de Usuarios

Usuarios Registrados									
									+ Agregar Usuario
Filtrar por:		Buscar:							
Usuarios									
Nombre	Apellido Paterno	Apellido Materno	Fecha de Nacimiento	RFC	Email	Username	Es administrador?	Ventas Realizadas	Acciones
Ana	García	López	1990-01-01	GALA9001017D5	ana_garcia@correo.com	ana_garcia	Sí	2	Editar Eliminar
Juan	Pérez	Martínez	1985-07-15	PEMJ8507159O4	juan_perez@correo.com	juan_perez	Sí	1	Editar Eliminar
María	Gómez	Hernández	2000-12-24	GOHM001224R2A	maria_gomez@correo.com	maria_gomez	Sí	1	Editar Eliminar
Laura	Pérez	Morales	1995-08-25	PEMO9508258R4	laura_p@correo.com	laura_perez	No	5	Editar Eliminar
David	Jiménez	Torres	2017-02-08	JITO8712024W3	david_j@correo.com	david_jimenez	No	0	Editar Eliminar

Formulario para registrar un usuario

Agregar nuevo usuario

Datos del Usuario

Nombre*

Apellido Paterno*

Apellido Materno*

Fecha de Nacimiento*

dd/mm/aaaa

RFC*

Email*

Nombre de Usuario*

Contraseña*

☐ Es admin

Ventas realizadas

0

Cancelar

Guardar Usuario

Como se mencionó anteriormente, todos los formularios cuentan con validaciones.

Datos del Usuario

Nombre*

31

Solo se permiten letras, incluyendo acentos y ñ

Apellido Paterno*

123

Solo se permiten letras, incluyendo acentos y ñ

Apellido Materno*

12

Solo se permiten letras, incluyendo acentos y ñ

Fecha de Nacimiento*

dd/mm/aaaa

La fecha de nacimiento debe estar en el pasado

RFC*

EQW43

El RFC debe tener exactamente 13 caracteres

Email*

wef

El correo debe ser válido

Nombre de Usuario*

q

Contraseña*

La contraseña debe tener entre 8 y 12 caracteres

☐ Es admin

Ventas realizadas

0

Cancelar

Guardar Usuario

Para la tabla relacionada, al presionar en el menú la opción de Playera, se muestra la lista que ya incluye campos de los catálogos “Marca” y “Equipo”.











Playeras Registradas

+ Agregar Playera

Mostrar

registros

Buscar:

ID	Color	Talla	Tipo Manga	Precio Real	Stock	Precio Venta	Marca	Equipo	Acciones
1	Azul	CH	CORTA	800	10	1299	Pirma	Cruz Azul FC	 
2	Azul	M	CORTA	800	10	1299	Pirma	Cruz Azul FC	 
3	Blanco	CH	CORTA	700	15	1199	Nike	Pumas	 
4	Vino	XG	CORTA	500	20	1499	Nike	Barcelona	 
5	Blanco	M	LARGA	900	25	1699	Adidas	Real Madrid	 

Mostrando registros del 1 al 5 de un total de 16 registros

Anterior

1

2

3

4

Siguiente

El formulario para agregar o editar una playera es el siguiente:

Agregar nueva playera

Datos de la Playera

Color*	Talla*
<input type="text"/>	<input type="text"/>
Tipo de Manga*	Stock*
<input type="text" value="CORTA"/>	<input type="text"/>
Precio de Venta*	Precio Real*
<input type="text"/>	<input type="text"/>
Equipo	Marca
<input type="text" value="Cruz Azul FC"/>	<input type="text" value="Adidas"/>

Dentro de los campos de “Equipo” y “Marca” se despliega una lista para seleccionar el de nuestra preferencia.

Selección de equipo y marca:

Selección de equipo

- Cruz Azul FC
- Pumas
- Barcelona
- Real Madrid
- Manchester United
- Liverpool FC
- Bayern Munich
- Borussia Dortmund
- Boca Juniors
- River Plate
- Feyenoord
- Atlético de Madrid
- Atlas
- Guadalajara

Cruz Azul FC

Selección de marca

- Adidas
- Charly
- DemoChanged
- Joma
- Nike
- Panam
- Pirma
- Puma
- Rayo Negro
- Umbro
- Under Armour

Adidas

II. Indicaciones especiales.

1. Configuraciones necesarias para la aplicación de consumo

Para realizar la parte del consumo de la API se separaron en los siguientes paquetes:

Para la parte rest, se crea el controlador que procesará los end-point solicitados, por ejemplo para el modelo PlayeraDto se muestran los siguientes:

```
@RestController no usages
@RequestMapping("/api/playeradto")
public class PlayeraDtoController {
```

```
/**
 * El end-point regresa la lista de todas las playeras existentes.
 * @return List
 */

@GetMapping("/listar-playeras")
public ResponseEntity<List<PlayeraDto>> getAll() { return ResponseEntity.ok(playeraDtoService.getPlayerasList()); }

/**
 * El end-point regresa la playera con un id específico.
 * @return Playera
 */

@GetMapping("/{id}") no usages
public ResponseEntity<PlayeraDto> getPlayeraDtoById(@PathVariable Integer id) {...}
```

```
/**
 * El end-point elimina el registro con un id específico.
 * @return Boolean
 */

@DeleteMapping("/id") no usages
public ResponseEntity<Boolean> deletePlayeraDto(@PathVariable Integer id) {...}

/**
 * El end-point registra una nueva playera.
 * @return Playera
 */

@PostMapping("/crear") no usages
public ResponseEntity<PlayeraDto> crearPlayeraDto(@RequestBody PlayeraDto playeraDto) throws URISyntaxException, ParseException,
```

```
/**
 * El end-point modifica una playera existente o la crea si no existe.
 * @return Playera
 */

@PutMapping("/{id}") no usages
public ResponseEntity<PlayeraDto> modificarPlayeraDto(@PathVariable Integer id, @RequestBody PlayeraDto playeraDto) throws URISyn
```

Que, a su vez, estos end-points se comunican con la parte del servicio:

```
@Override 1 usage
public List<PlayeraDto> getPlayerasList() {
    List<Playera> playeras = playeraRepository.findAll();
    return playeras.stream().map(this::toDto)
        .collect(Collectors.toList());
}

@Override 1 usage
public PlayeraDto updatePlayera(PlayeraDto playera) throws ParseException {
    Playera playeraUpdated = playeraRepository.save(this.toEntity(playera));
    return toDto(playeraUpdated);
}

@Override 2 usages
public PlayeraDto createPlayera(PlayeraDto playera) throws ParseException {
    Playera playeraGuardada = playeraRepository.save(this.toEntity(playera));
    return toDto(playeraGuardada);
}
```

```
@Override 1 usage
public boolean deletePlayera(Integer id) {
    Optional<Playera> playera = playeraRepository.findById(id);
    if (playera.isPresent()){
        playeraRepository.deleteById(id);
        return true;
    }else {
        return false;
    }
}

@Override 2 usages
public Optional<PlayeraDto> getPlayeraById(Integer id) {
    Optional<Playera> playera = playeraRepository.findById(id);
    if(playera.isPresent()){
        PlayeraDto playeraDto = toDto(playera.get());
        return Optional.of(playeraDto);
    }
    else {
        return Optional.empty();
    }
}
```

Para poder usar un objeto DTO, se debe realizar una conversión de la entidad al modelo DTO y viceversa.

```
private PlayeraDto toDto(Playera playera){ 4 usages
    PlayeraDto playeraDto = modelMapper.map(playera, PlayeraDto.class);
    playeraDto.setMarca(playera.getMarca().getNombre()); //No se revisa si != null ya que es obligatoria.
    playeraDto.setEquipo(playera.getEquipo().getNombre());
    return playeraDto;
}
```

```
private Playera toEntity(PlayeraDto playeraDto) throws ParseException{ 2 usages
    Playera playera = modelMapper.map(playeraDto, Playera.class);
    if(playeraDto.getMarca() != null && !playeraDto.getMarca().isEmpty()){
        Marca marca = marcaRepository.findByNombre(playeraDto.getMarca());
        if(marca == null){
            throw new MarcaNotFoundException("La marca no existe");
        }else{
            playera.setMarca(marca);
        }
    }
    if(playeraDto.getEquipo() != null && !playeraDto.getEquipo().isEmpty()){
        Equipo equipo = equipoRepository.findByNombre(playeraDto.getEquipo());
        if(equipo==null){
            throw new EquipoNotFoundException("El equipo no existe");
        }else{
            playera.setEquipo(equipo);
        }
    }
    return playera;
}
```

Para la parte Front -end que su funcionamiento fue explicado al inicio del documento, se diseñó el controller en el que a diferencia de regresar un *ResponseEntity* se regresa el documento HTML de la interfaz para el usuario.

Utilizando el mismo ejemplo sobre la Clase Playera:

```
/**
 * Este controlador se define para el front-end de una playera, contiene todas las vistas
 * que serán visibles dentro del navegador.
 *
 * @author Rodrigo Martínez Zambrano
 * @version 0.0.1
 */

@Controller no usages
@RequestMapping("/playera")
public class PlayeraFrontController {
```

```
/**
 * El end-point mostrará una tabla con todas las playeras y sus características.
 * @return String
 */

@GetMapping(path = "/lista-playera") no usages
public String getAllPlayeras(Model model){
    model.addAttribute(attributeName: "playera",playeraService.getAll());
    return "/playera/lista-playera";
}

/**
 * El end-point muestra un formulario para ingresar los datos de una playera y
 * registrarla en el sistema.
 * @return String
 */

@GetMapping("/alta-playera") no usages
public String altaPlayera(Model model){
    Playera playera = new Playera();
    List<Equipo> equipos = equipoService.getEquiposList();
    List<Marca> marcas = marcaService.getMarcasList();

    model.addAttribute(attributeName: "equipo",equipos);
    model.addAttribute(attributeName: "marca",marcas);
    model.addAttribute(attributeName: "playera",playera);
    model.addAttribute(attributeName: "contenido", attributeValue: "Agregar nueva playera");
    return "/playera/alta-playera";
}
```

```
/**
 * El end-point realiza la acción de guardar la playera en el sistema.
 * @return String
 */

@PostMapping("/salvar-playera") no usages
public String salvarPlayera(@Valid @ModelAttribute("playera") Playera playera,
                           BindingResult result,
                           Model model,
                           RedirectAttributes flash) {
    List<Equipo> equipos = equipoService.getEquiposList();
    List<Marca> marcas = marcaService.getMarcasList();

    model.addAttribute(attributeName: "equipo", equipos);
    model.addAttribute(attributeName: "marca", marcas);

    if (result.hasErrors()) {
        model.addAttribute(attributeName: "contenido", attributeValue: "ERROR. No debe ser vacío");
        return "/playera/alta-playera";
    }

    playeraService.crearPlayera(playera);
    System.out.println(playera);
    flash.addFlashAttribute(attributeName: "success", attributeValue: "La playera se guardó correctamente");

    return "redirect:/playera/lista-playera";
}

/**
 * El end-point realiza la acción de eliminar una playera existente.
 * @return String
 */

@GetMapping("/eliminar-playera/{id}") no usages
public String eliminarPlayera(@PathVariable Integer id, RedirectAttributes flash) {
    try {
        playeraService.deletePlayera(id);
        flash.addFlashAttribute(attributeName: "success", attributeValue: "La playera se borró correctamente.");
    } catch (DataIntegrityViolationException e) {
        // Este error ocurre cuando hay violación de integridad referencial
        flash.addFlashAttribute(attributeName: "error",
                                attributeValue: "No se puede eliminar la playera.");
    } catch (Exception e) {
        // Para cualquier otro error inesperado
        flash.addFlashAttribute(attributeName: "error", e.getMessage());
    }

    return "redirect:/playera/lista-playera";
}
```

```
/**
 * El end-point muestra un formulario para modificar información de una playera existente.
 * @return String
 */

@GetMapping("/editar-playera/{id}") no usages
public String modificarPlayera(@PathVariable Integer id, Model model){
    Playera playera = playeraService.getPlayeraById(id);
    List<Equipo> equipos = equipoService.getEquiposList();
    List<Marca> marcas = marcaService.getMarcasList();

    model.addAttribute(attributeName: "equipo", equipos);
    model.addAttribute(attributeName: "marca", marcas);
    model.addAttribute(attributeName: "playera", playera);
    model.addAttribute(attributeName: "contenido", attributeValue: "Modificar Playera");
    model.addAttribute(attributeName: "subtitulo", attributeValue: "Formulario para modificar una playera existente.");
    return "/playera/alta-playera";
}
```

Estos controladores se comunican con el servicio para poder realizar las peticiones correctamente. En este caso, se utilizó WebFlux para realizar dicha tarea.

```
@Service 2 usages
public class PlayeraFrontService {
    @Autowired 5 usages
    WebClient webClient;

    public List<Playera> getAll(){
        Mono<List<Playera>> playerasMono = webClient.get() RequestHeadersUriSpec<capture of ?>
            .uri( uri: "/api/playera/listar-playeras") capture of ?
            .retrieve() ResponseSpec
            .bodyToFlux(Playera.class) Flux<Playera>
            .collectList();
        return playerasMono.block();
    }

    public Playera getPlayeraById(Integer id){ 1 usage
        Mono<Playera> playeraMono = webClient.get() RequestHeadersUriSpec<capture of ?>
            .uri( uri: "/api/playera/{id}", id) capture of ?
            .retrieve() ResponseSpec
            .bodyToMono(Playera.class);
        return playeraMono.block();
    }
}
```

```
public void actualizaPlayera(Playera playera) { no usages
    webClient.put() RequestBodyUriSpec
        .uri( uri: "/api/playera/{id}", playera.getIdPlayera()) RequestBodySpec
        .bodyValue(playera) RequestHeadersSpec<capture of ?>
        .retrieve() ResponseSpec
        .bodyToMono(Playera.class) Mono<Playera>
        .block();
}

public Playera crearPlayera(Playera playera){ 1 usage
    return webClient.post() RequestBodyUriSpec
        .uri( uri: "/api/playera/crear") RequestBodySpec
        .bodyValue(playera) RequestHeadersSpec<capture of ?>
        .retrieve() ResponseSpec
        .bodyToMono(Playera.class) Mono<Playera>
        .block();
}
```

2. Pruebas de CRUD en Postman

Ahora se consumirá la API desde Postman para el catálogo de Usuarios.

- Operación GET ALL al catálogo, se realiza mediante el siguiente end-point:

<http://localhost:8080/api/usuario/listar-usuarios>

Dará como resultado un JSON de UsuarioDto.

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/api/usuariodto/listar-usuarios
- Status:** 200 OK
- Time:** 141 ms
- Size:** 1.49 KB
- Response Type:** JSON
- Response Body:** A JSON array containing three user objects.

```
[
  {
    "idUsuario": 1,
    "nombre": "Ana",
    "apellidoPat": "García",
    "apellidoMat": "López",
    "fechaNac": "1990-01-01",
    "username": "ana_garcia",
    "ventasRealizadas": 2
  },
  {
    "idUsuario": 2,
    "nombre": "Juan",
    "apellidoPat": "Pérez",
    "apellidoMat": "Martínez",
    "fechaNac": "1985-07-15",
    "username": "juan_perez",
    "ventasRealizadas": 1
  },
  {
    "idUsuario": 3,
    "nombre": "María",
    "apellidoPat": "Gómez",
    "apellidoMat": "Hernández",
    "fechaNac": "2000-12-24",
    "username": "maria_gomez",
    "ventasRealizadas": 1
  }
]
```


- Petición GET Usuario: <http://localhost:8080/api/usuariodto/4>

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** <http://localhost:8080/api/usuariodto/4>
- Status:** 200 OK
- Response Time:** 21 ms
- Response Size:** 313 B
- Response Body (JSON):**

```
{
  "idUsuario": 4,
  "nombre": "Laura",
  "apellidoPat": "Pérez",
  "apellidoMat": "Morales",
  "fechaNac": "1995-08-25",
  "username": "laura_perez",
  "ventasRealizadas": 5
}
```

- Petición POST de un usuario, en este caso debido a que la base de datos tiene restricciones de NOT NULL se debe crear un nuevo usuario completo, es decir, sin usar UsuarioDto, la única columna que se excluye es el Id del usuario debido a que se genera automáticamente. <http://localhost:8080/api/usuario/crear>

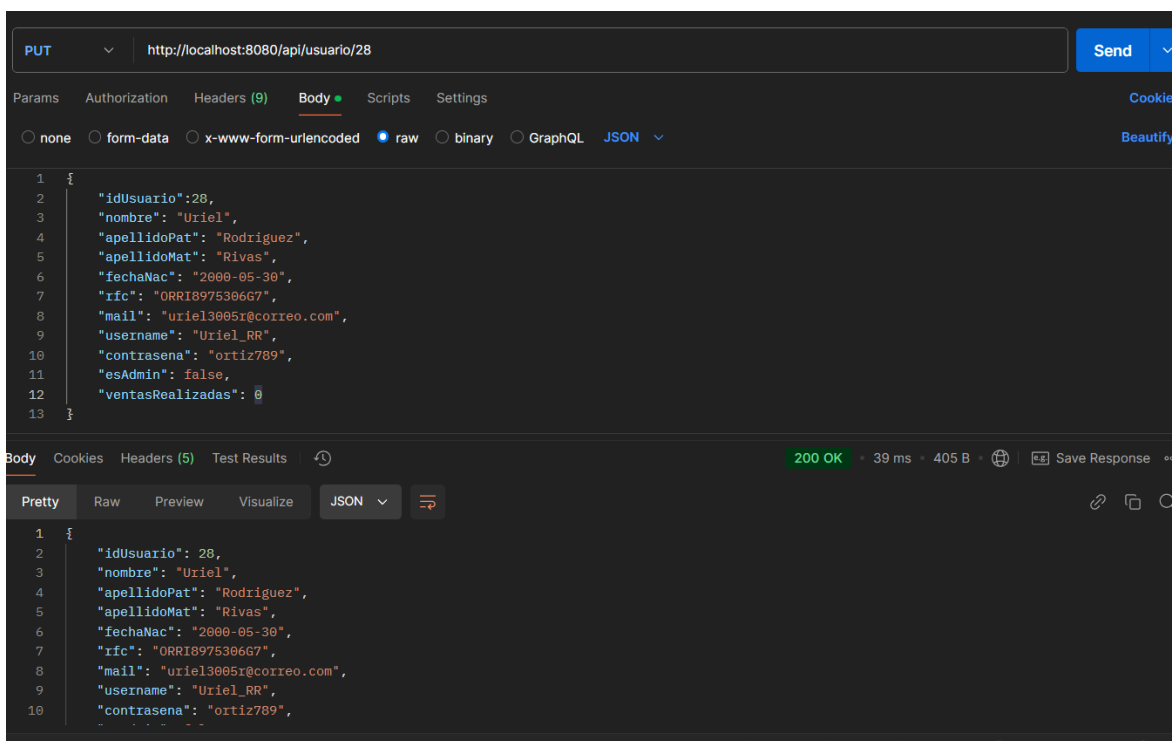
The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** <http://localhost:8080/api/usuario/crear>
- Status:** 201 Created
- Response Time:** 123 ms
- Response Size:** 435 B
- Request Body (JSON):**

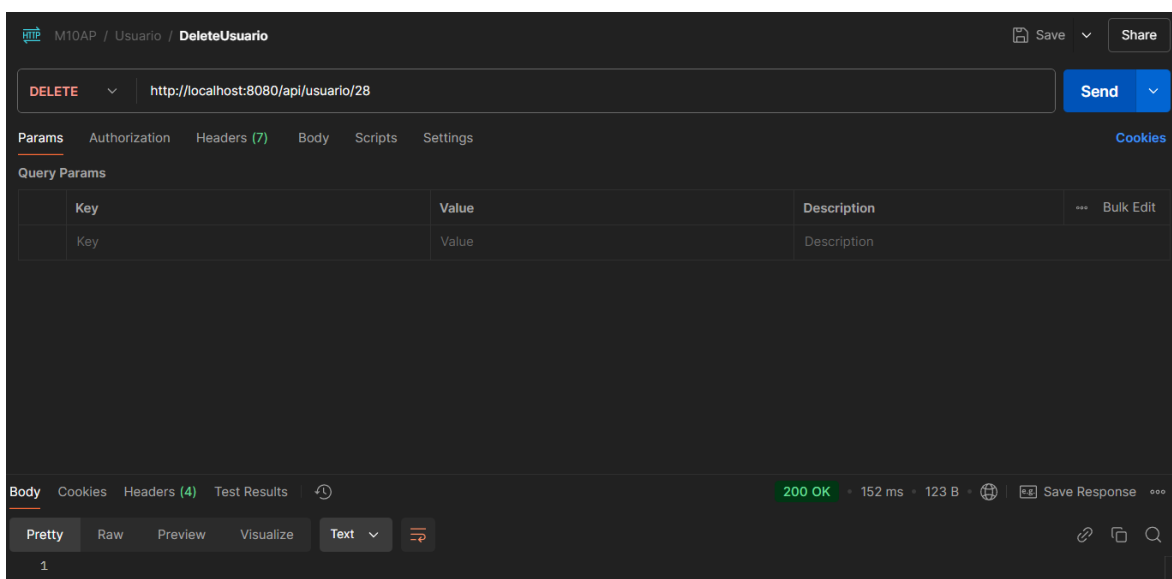
```
{
  "nombre": "Uriel",
  "apellidoPat": "Rodriguez",
  "apellidoMat": "Rivas",
  "fechaNac": "1989-05-30",
  "rfc": "ORRI8975306G7",
  "mail": "fernandoo77r@correo.com",
  "username": "fer_ortiz",
  "contrasena": "ortiz789",
  "esAdmin": false,
  "ventasRealizadas": 0
}
```
- Response Body (JSON):**

```
{
  "idUsuario": 28,
  "nombre": "Uriel",
  "apellidoPat": "Rodriguez",
  "apellidoMat": "Rivas",
  "fechaNac": "1989-05-30",
  "rfc": "ORRI8975306G7",
  "mail": "fernandoo77r@correo.com",
  "username": "fer_ortiz",
  "contrasena": "ortiz789",
  "esAdmin": false,
  "ventasRealizadas": 0
}
```

- Petición PUT, se realiza la actualización de datos de un usuario. A diferencia de la imagen anterior, se observa que se modificaron los datos de la Fecha de Nacimiento (fechaNac), el “mail” y “username”. <http://localhost:8080/api/usuario/28>



- Petición DELETE. Se borró el usuario creado para este ejemplo. <http://localhost:8080/api/usuario/28>



De igual manera que del lado del Front-end de la aplicación, en la API se manejan excepciones en caso de errores, por ejemplo con la playera, si se quiere registrar una playera con un equipo o una marca que no existen, se generan excepción y se muestra un mensaje amigable al usuario.

```
1 {
2   "color": "Azul",
3   "talla": "CH",
4   "tipoManga": "CORTA",
5   "precioReal": 800,
6   "stock": 10,
7   "precioVenta": 1299,
8   "marca": {
9     "idMarca": 3
10  },
11  "equipo": {
12    "idEquipo": 38
13  }
14 }
```

Body Cookies Headers (5) Test Results ↻

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "statusCode": "409 CONFLICT",
3   "mensaje": "El equipo con ID: 38 no existe.",
4   "timestamp": "2024-11-19T11:28:43.8759206"
5 }
```

```
1 {
2   "color": "Azul",
3   "talla": "CH",
4   "tipoManga": "CORTA",
5   "precioReal": 800,
6   "stock": 10,
7   "precioVenta": 1299,
8   "marca": {
9     "idMarca": 33
10  },
11  "equipo": {
12    "idEquipo": 3
13  }
14 }
```

Body Cookies Headers (5) Test Results ↻

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "statusCode": "409 CONFLICT",
3   "mensaje": "La marca con ID: 33 no existe.",
4   "timestamp": "2024-11-19T11:29:07.7805532"
5 }
```