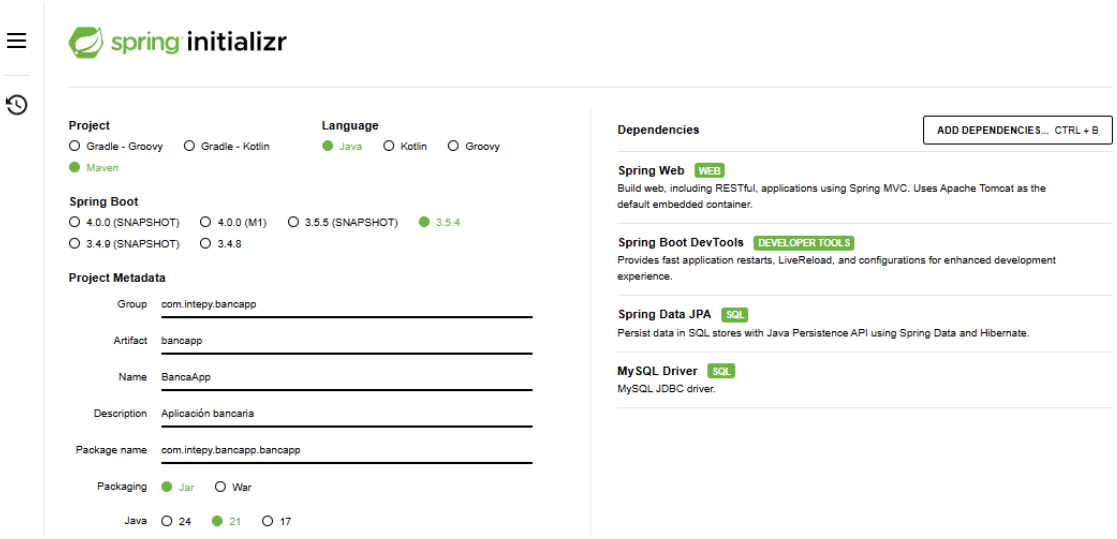
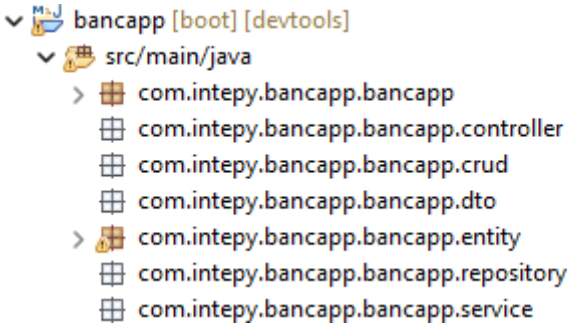
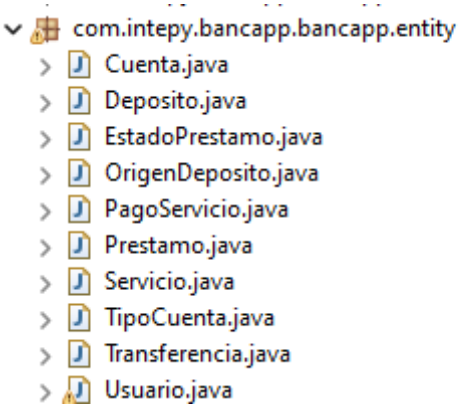
Java Web


spring initializr
Project
Gradle - Groovy
Gradle - Kotlin
Maven
Language
Java
Kotlin
Groovy
Spring Boot
4.0.0 (SNAPSHOT)
4.0.0 (M1)
3.5.5 (SNAPSHOT)
3.5.4
3.4.9 (SNAPSHOT)
3.4.8
Project Metadata
Group com.intepy.bancapp
Artifact bancapp
Name BancaApp
Description Aplicación bancaria
Package name com.intepy.bancapp.bancapp
Packaging Jar War
Java 24 21 17
Dependencies
ADD DEPENDENCIES... CTRL + B
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
MySQL Driver SQL
MySQL JDBC driver.


bancapp [boot] [devtools]
src/main/java
com.intepy.bancapp.bancapp
com.intepy.bancapp.bancapp.controller
com.intepy.bancapp.bancapp.crud
com.intepy.bancapp.bancapp.dto
com.intepy.bancapp.bancapp.entity
com.intepy.bancapp.bancapp.repository
com.intepy.bancapp.bancapp.service


com.intepy.bancapp.bancapp.entity
Cuenta.java
Deposito.java
EstadoPrestamo.java
OrigenDeposito.java
PagoServicio.java
Prestamo.java
Servicio.java
TipoCuenta.java
Transferencia.java
Usuario.java

Usuario.java

```java
package com.intepy.bancapp.bancapp.entity;


import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.List;

@Entity
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nombre;
    private String email;

    @OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL)
    private List<Cuenta> cuentas = new ArrayList<>();

    @OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL)
    private List<Prestamo> prestamos = new ArrayList<>();

    // Constructores, getters y setters
}
```

Cuenta.java

```java
package com.intepy.bancapp.bancapp.entity;

import java.util.ArrayList;

@Entity
public class Cuenta {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String numero;
    private Double saldo;

    @ManyToOne
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;

    @ManyToOne
    @JoinColumn(name = "tipo_cuenta_id")
    private TipoCuenta tipoCuenta;

    @OneToMany(mappedBy = "cuenta", cascade = CascadeType.ALL)
    private List<Deposito> depositos = new ArrayList<>();

    @OneToMany(mappedBy = "cuentaOrigen")
    private List<Transferencia> transferenciasOrigen = new ArrayList<>();

    @OneToMany(mappedBy = "cuentaDestino")
    private List<Transferencia> transferenciasDestino = new ArrayList<>();

    //constuctores, getters and setter
}
```

Deposito.java

```java
package com.intepy.bancapp.bancapp.entity;

import jakarta.persistence.Entity;

@Entity
public class Deposito {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Double monto;

    @ManyToOne
    @JoinColumn(name = "cuenta_id")
    private Cuenta cuenta;

    @ManyToOne
    @JoinColumn(name = "origen_id")
    private OrigenDeposito origen;

    //constructor, getter and setter
}
```

OrigenDeposito.java

```java
package com.intepy.bancapp.bancapp.entity;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class OrigenDeposito {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String descripcion;

    @OneToMany(mappedBy = "origen")
    private List<Deposito> depositos;
}
```

Prestamo.java

```java
package com.intepy.bancapp.bancapp.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class Prestamo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Double monto;

    @ManyToOne
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;

    @ManyToOne
    @JoinColumn(name = "estado_id")
    private EstadoPrestamo estado;

    //constructores, getters and setters
}
```

EstadoPrestamo.java

```java
package com.intepy.bancapp.bancapp.entity;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class EstadoPrestamo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String descripcion; // Aprobado, Pendiente, Rechazado

    @OneToMany(mappedBy = "estado")
    private List<Prestamo> prestamos;
}
```

Servicio.java

```java
package com.intepy.bancapp.bancapp.entity;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class Servicio {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nombre; // Agua, Luz, Internet

    @OneToMany(mappedBy = "servicio")
    private List<PagoServicio> pagos;
}
```

PagoServicio.java

```java
package com.intepy.bancapp.bancapp.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class PagoServicio {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Double monto;

    @ManyToOne
    @JoinColumn(name = "cuenta_id")
    private Cuenta cuenta;

    @ManyToOne
    @JoinColumn(name = "servicio_id")
    private Servicio servicio;
}
```

Transferencia.java

```java
package com.intepy.bancapp.bancapp.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class Transferencia {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Double monto;

    @ManyToOne
    @JoinColumn(name = "cuenta_origen_id")
    private Cuenta cuentaOrigen;

    @ManyToOne
    @JoinColumn(name = "cuenta_destino_id")
    private Cuenta cuentaDestino;

    //cosntructores, getters and setters
}
```

TipoCuenta.java

```java
package com.intepy.bancapp.bancapp.entity;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class TipoCuenta {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nombre; // Ahorro, Corriente, etc.

    @OneToMany(mappedBy = "tipoCuenta")
    private List<Cuenta> cuentas;
}
```

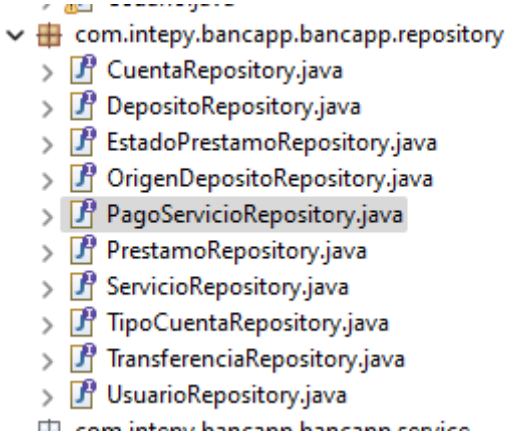
Java Web

Configura la conexión a la base de datos

application.properties

```
spring.application.name=BancaApp
spring.datasource.url=jdbc:mysql://localhost:3306/bd_ueno?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

Repositories, crear las interfaces



UsuarioRepository.java

```
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.Usuario;
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {}
```

CuentaRepository.java

```
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.Cuenta;

public interface CuentaRepository extends JpaRepository<Cuenta, Long> {}

```

TipoCuentaRepository.java

```
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.TipoCuenta;

public interface TipoCuentaRepository extends JpaRepository<TipoCuenta, Long> {}
```

DepositoRepository.java

```java
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.Deposito;

public interface DepositoRepository extends JpaRepository<Deposito, Long> {}

```

*PrestamoRepository.java

```java
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;
public interface PrestamoRepository extends JpaRepository<Prestamo, Long> {}

```

OrigenDepositoRepository.java

```java
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.intepy.bancapp.bancapp.entity.OrigenDeposito;

public interface OrigenDepositoRepository extends JpaRepository<OrigenDeposito,Long> {}
```

TransferenciaRepository.java

```java
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.Transferencia;

public interface TransferenciaRepository extends JpaRepository<Transferencia, Long> {}

```

EstadoPrestamoRepository.java

```java
package com.intepy.bancapp.bancapp.repository;
import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.EstadoPrestamo;
public interface EstadoPrestamoRepository extends JpaRepository<EstadoPrestamo, Long> {}
```

ServicioRepository.java

```java
package com.intepy.bancapp.bancapp.repository;
import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.Servicio;
public interface ServicioRepository extends JpaRepository<Servicio, Long> {}
```

*PagoServicioRepository.java

```java
package com.intepy.bancapp.bancapp.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.intepy.bancapp.bancapp.entity.PagoServicio;

public interface PagoServicioRepository extends JpaRepository<PagoServicio, Long> {}

```

Java Web

Configurar la base de datos

application.properties

```
spring.application.name=BancaApp
spring.datasource.url=jdbc:mysql://localhost:3306/bd_ueno?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

Services

UsuarioService. | CuentaService.j | PrestamoService | Usuario.java | Cuenta.java

```
package com.intepy.bancapp.bancapp.service;

import java.util.List;

@Service
public class UsuarioService {

    @Autowired
    private UsuarioRepository usuarioRepository;

    public List<Usuario> listarUsuarios() {
        return usuarioRepository.findAll();
    }

    public Optional<Usuario> obtenerUsuarioPorId(Long id) {
        return usuarioRepository.findById(id);
    }

    public Usuario guardarUsuario(Usuario usuario) {
        return usuarioRepository.save(usuario);
    }

    public Usuario actualizarUsuario(Long id, Usuario usuarioActualizado) {
        return usuarioRepository.findById(id)
                .map(usuario -> {
                    usuario.setNombre(usuarioActualizado.getNombre());
                    usuario.setEmail(usuarioActualizado.getEmail());
                    return usuarioRepository.save(usuario);
                })
                .orElseThrow(() -> new RuntimeException("Usuario no encontrado"));
    }

    public void eliminarUsuario(Long id) {
        usuarioRepository.deleteById(id);
    }
}
```

UsuarioService. | CuentaService.j | PrestamoService | Usuario.java | Cuenta.java

```java
package com.intepy.bancapp.bancapp.service;
import java.util.List;

@Service
public class CuentaService {

    @Autowired
    private CuentaRepository cuentaRepository;

    public List<Cuenta> listarCuentas() {
        return cuentaRepository.findAll();
    }

    public Optional<Cuenta> obtenerCuentaPorId(Long id) {
        return cuentaRepository.findById(id);
    }

    public Cuenta guardarCuenta(Cuenta cuenta) {
        return cuentaRepository.save(cuenta);
    }

    public Cuenta actualizarCuenta(Long id, Cuenta cuentaActualizada) {
        return cuentaRepository.findById(id)
                .map(cuenta -> {
                    cuenta.setNumero(cuentaActualizada.getNumero());
                    cuenta.setSaldo(cuentaActualizada.getSaldo());
                    return cuentaRepository.save(cuenta);
                })
                .orElseThrow(() -> new RuntimeException("Cuenta no encontrada"));
    }

    public void eliminarCuenta(Long id) {
        cuentaRepository.deleteById(id);
    }
}
```

Java Web

UsuarioService. | CuentaService.j | PrestamoService | Usuario.java | Cuenta.java

```java
package com.intepy.bancapp.bancapp.service;

import java.util.List;

@Service
public class PrestamoService {

    @Autowired
    private PrestamoRepository prestamoRepository;

    public List<Prestamo> listarPrestamos() {
        return prestamoRepository.findAll();
    }

    public Optional<Prestamo> obtenerPrestamoPorId(Long id) {
        return prestamoRepository.findById(id);
    }

    public Prestamo guardarPrestamo(Prestamo prestamo) {
        return prestamoRepository.save(prestamo);
    }

    public Prestamo actualizarPrestamo(Long id, Prestamo prestamoActualizado) {
        return prestamoRepository.findById(id)
                .map(prestamo -> {
                    prestamo.setMonto(prestamoActualizado.getMonto());
                    prestamo.setEstado(prestamoActualizado.getEstado());
                    return prestamoRepository.save(prestamo);
                })
                .orElseThrow(() -> new RuntimeException("Préstamo no encontrado"));
    }

    public void eliminarPrestamo(Long id) {
        prestamoRepository.deleteById(id);
    }
}
```

Java Web

Controllers

```java
@RestController
@RequestMapping("/api/usuarios")
public class UsuarioController {

    @Autowired
    private UsuarioService usuarioService;

    @GetMapping
    public List<Usuario> listarUsuarios() {
        return usuarioService.listarUsuarios();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Usuario> obtenerUsuario(@PathVariable Long id) {
        return usuarioService.obtenerUsuarioPorId(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public Usuario crearUsuario(@RequestBody Usuario usuario) {
        return usuarioService.guardarUsuario(usuario);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Usuario> actualizarUsuario(@PathVariable Long id, @RequestBody Usuario usuario) {
        try {
            return ResponseEntity.ok(usuarioService.actualizarUsuario(id, usuario));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarUsuario(@PathVariable Long id) {
        usuarioService.eliminarUsuario(id);
        return ResponseEntity.noContent().build();
    }
}
```

```java
@RestController
@RequestMapping("/api/cuentas")
public class CuentaController {

    @Autowired
    private CuentaService cuentaService;

    @GetMapping
    public List<Cuenta> listarCuentas() {
        return cuentaService.listarCuentas();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Cuenta> obtenerCuenta(@PathVariable Long id) {
        return cuentaService.obtenerCuentaPorId(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public Cuenta crearCuenta(@RequestBody Cuenta cuenta) {
        return cuentaService.guardarCuenta(cuenta);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Cuenta> actualizarCuenta(@PathVariable Long id, @RequestBody Cuenta cuenta) {
        try {
            return ResponseEntity.ok(cuentaService.actualizarCuenta(id, cuenta));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarCuenta(@PathVariable Long id) {
        cuentaService.eliminarCuenta(id);
        return ResponseEntity.noContent().build();
    }
}
```

Java Web

```java
@RestController
@RequestMapping("/api/depositos")
public class DepositoController {

    @Autowired
    private DepositoService depositoService;

    @GetMapping
    public List<Deposito> listarDepositos() {
        return depositoService.listarDepositos();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Deposito> obtenerDeposito(@PathVariable Long id) {
        return depositoService.obtenerDepositoPorId(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public Deposito crearDeposito(@RequestBody Deposito deposito) {
        return depositoService.guardarDeposito(deposito);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Deposito> actualizarDeposito(@PathVariable Long id, @RequestBody Deposito deposito) {
        try {
            return ResponseEntity.ok(depositoService.actualizarDeposito(id, deposito));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarDeposito(@PathVariable Long id) {
        depositoService.eliminarDeposito(id);
        return ResponseEntity.noContent().build();
    }
}
```

```java
@RestController
@RequestMapping("/api/pagos")
public class PagoServicioController {

    @Autowired
    private PagoServicioService pagoServicioService;

    @GetMapping
    public List<PagoServicio> listarPagos() {
        return pagoServicioService.listarPagos();
    }

    @GetMapping("/{id}")
    public ResponseEntity<PagoServicio> obtenerPago(@PathVariable Long id) {
        return pagoServicioService.obtenerPagoPorId(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public PagoServicio crearPago(@RequestBody PagoServicio pago) {
        return pagoServicioService.guardarPago(pago);
    }

    @PutMapping("/{id}")
    public ResponseEntity<PagoServicio> actualizarPago(@PathVariable Long id, @RequestBody PagoServicio pago) {
        try {
            return ResponseEntity.ok(pagoServicioService.actualizarPago(id, pago));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarPago(@PathVariable Long id) {
        pagoServicioService.eliminarPago(id);
        return ResponseEntity.noContent().build();
    }
}
```

Java Web

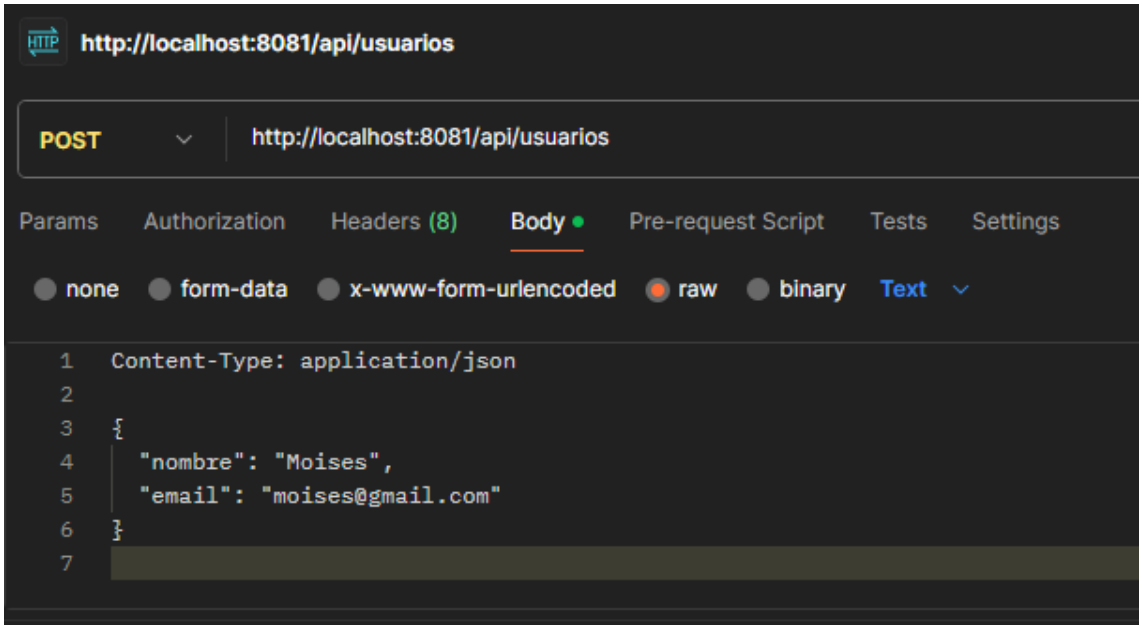
```java
@RestController
@RequestMapping("/api/prestamos")
public class PrestamoController {

    @Autowired
    private PrestamoService prestamoService;

    @GetMapping
    public List<Prestamo> listarPrestamos() {
        return prestamoService.listarPrestamos();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Prestamo> obtenerPrestamo(@PathVariable Long id) {
        return prestamoService.obtenerPrestamoPorId(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public Prestamo crearPrestamo(@RequestBody Prestamo prestamo) {
        return prestamoService.guardarPrestamo(prestamo);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Prestamo> actualizarPrestamo(@PathVariable Long id, @RequestBody Prestamo prestamo) {
        try {
            return ResponseEntity.ok(prestamoService.actualizarPrestamo(id, prestamo));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarPrestamo(@PathVariable Long id) {
        prestamoService.eliminarPrestamo(id);
        return ResponseEntity.noContent().build();
    }
}
```

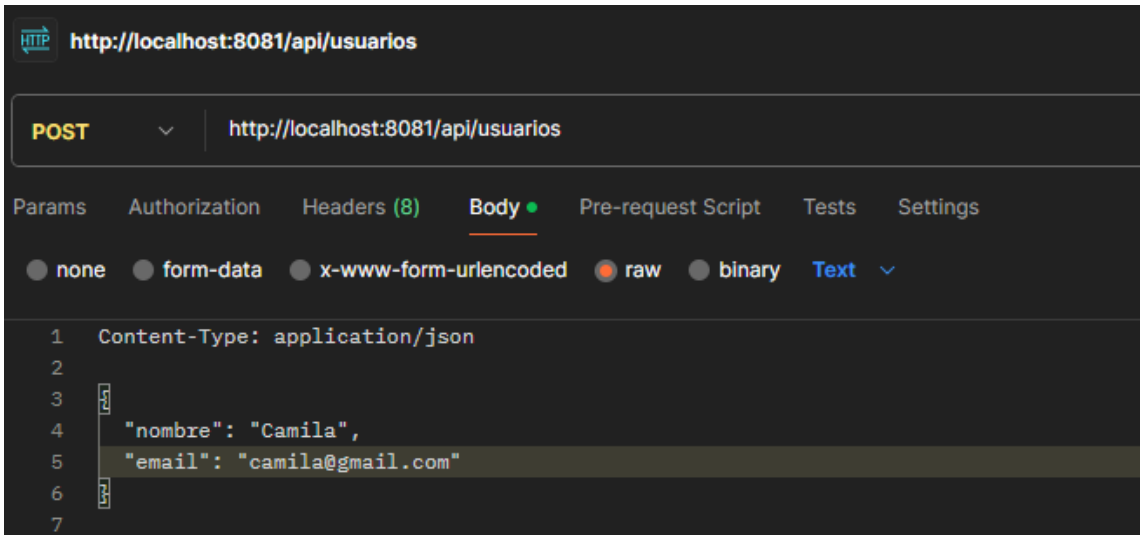
```java
@RestController
@RequestMapping("/api/transferencias")
public class TransferenciaController {

    @Autowired
    private TransferenciaService transferenciaService;

    @GetMapping
    public List<Transferencia> listarTransferencias() {
        return transferenciaService.listarTransferencias();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Transferencia> obtenerTransferencia(@PathVariable Long id) {
        return transferenciaService.obtenerTransferenciaPorId(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping
    public Transferencia crearTransferencia(@RequestBody Transferencia transferencia) {
        return transferenciaService.guardarTransferencia(transferencia);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Transferencia> actualizarTransferencia(@PathVariable Long id, @RequestBody Transferencia transferencia) {
        try {
            return ResponseEntity.ok(transferenciaService.actualizarTransferencia(id, transferencia));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> eliminarTransferencia(@PathVariable Long id) {
        transferenciaService.eliminarTransferencia(id);
        return ResponseEntity.noContent().build();
    }
}
```

Java Web

Probar el sistema utilizando Postman, según el siguiente flujo:

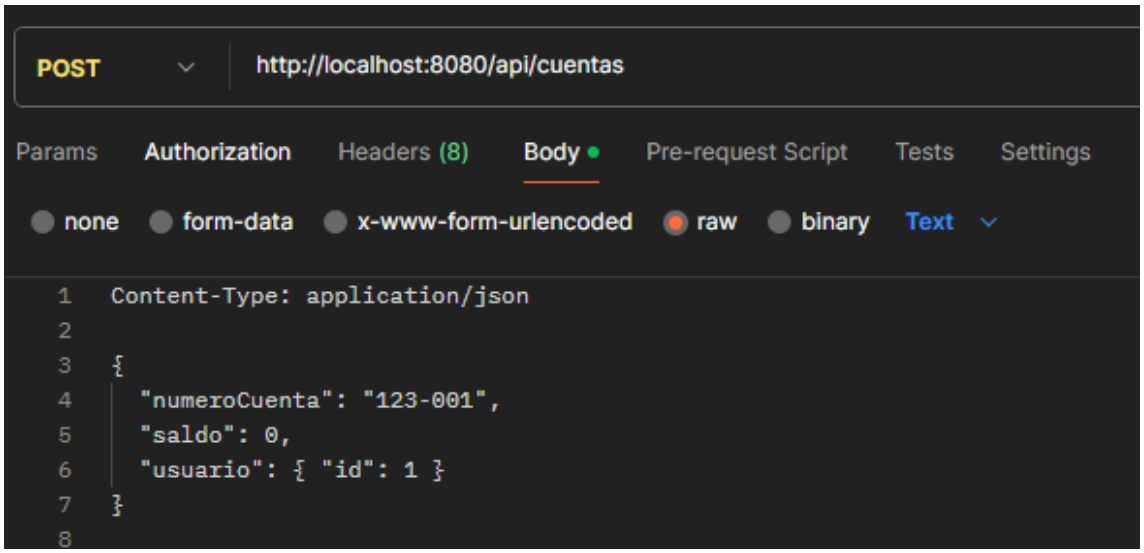
1. Crear usuarios: Moisés y Camila.
2. Crear cuentas para ellos.
3. Crear servicios (ANDE y ESSAP).
4. Conceder un préstamo a Moisés.
5. Hacer un depósito en la cuenta de Moisés con el dinero del préstamo.
6. Realizar una transferencia desde Moisés a Camila.
7. Realizar un pago de servicio (ANDE) desde la cuenta de Moisés.

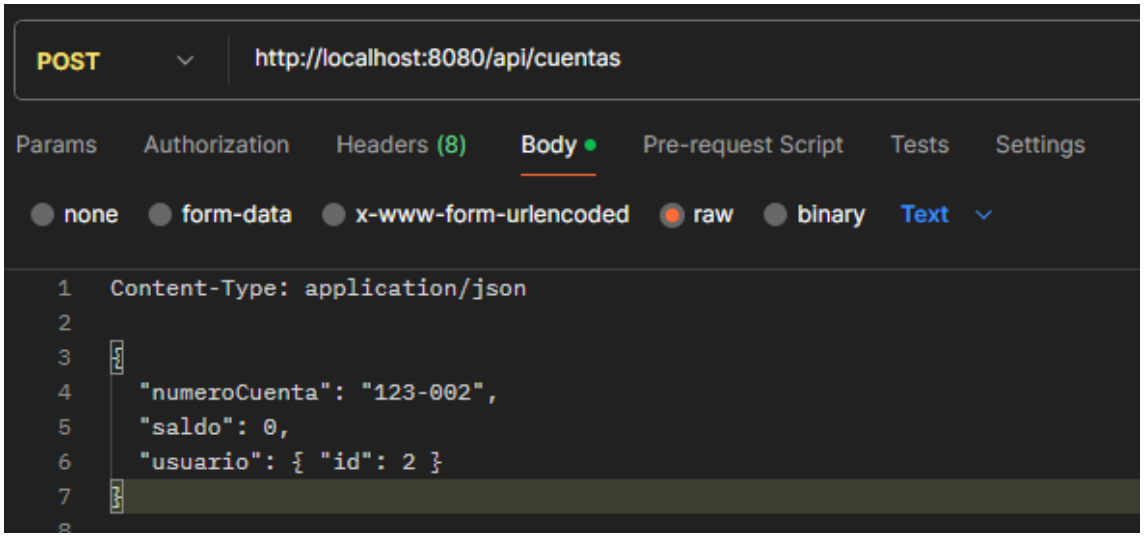




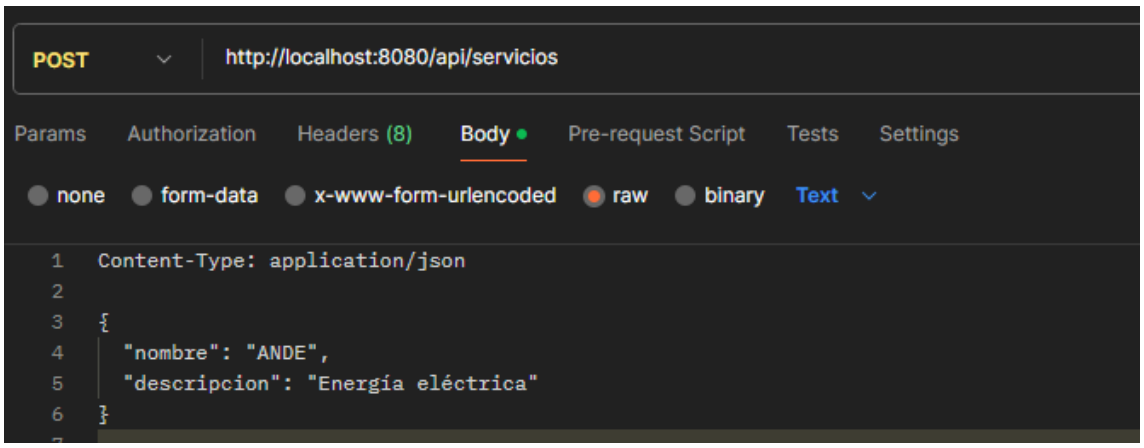
Java Web

Cuentas:

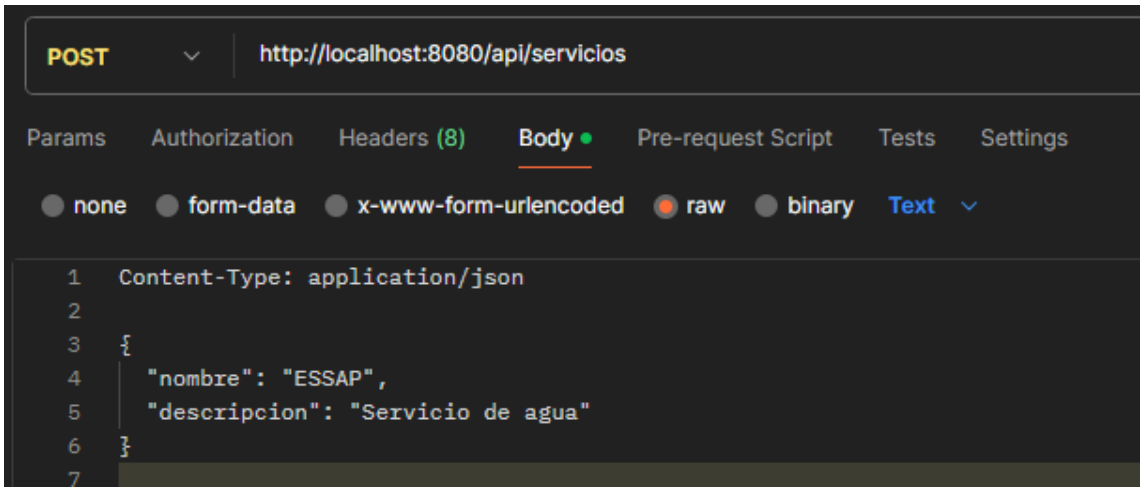




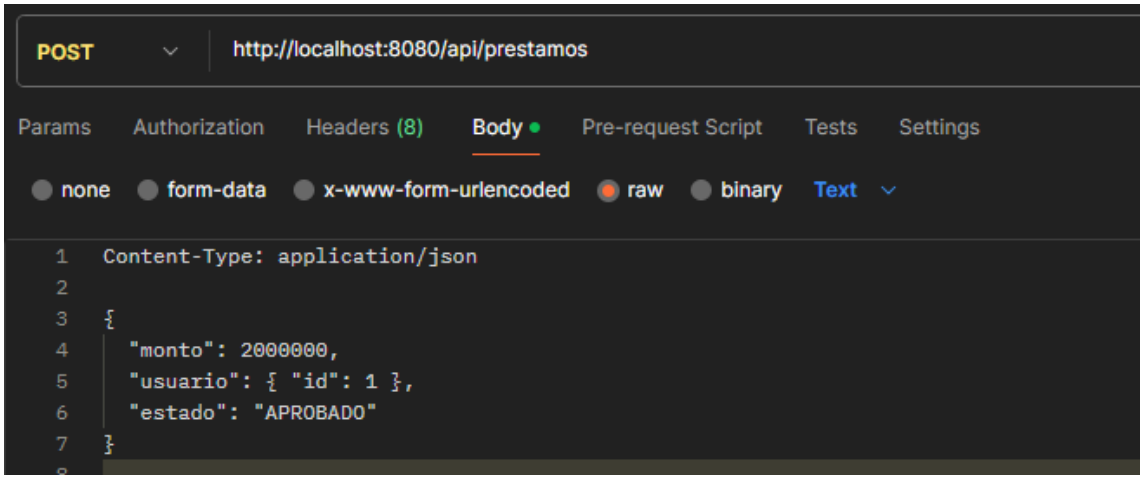
Servicios



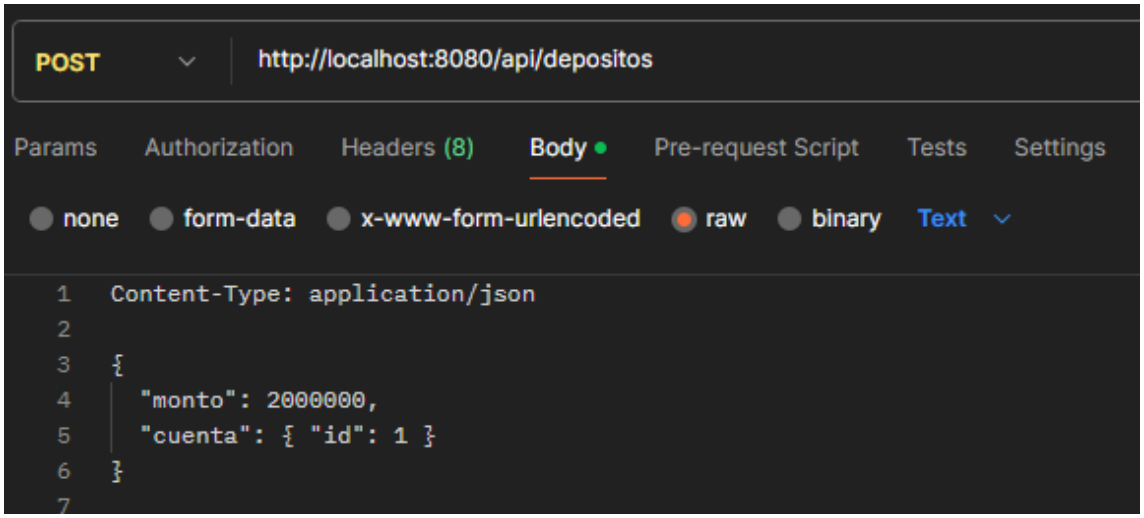
Java Web



Prestamos

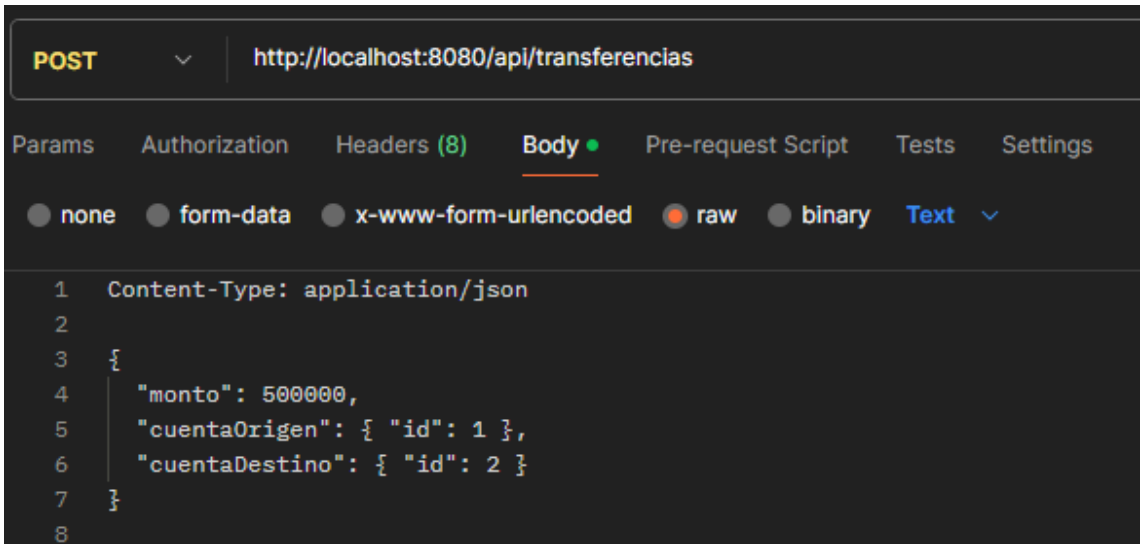


Depósito de préstamo en cuenta

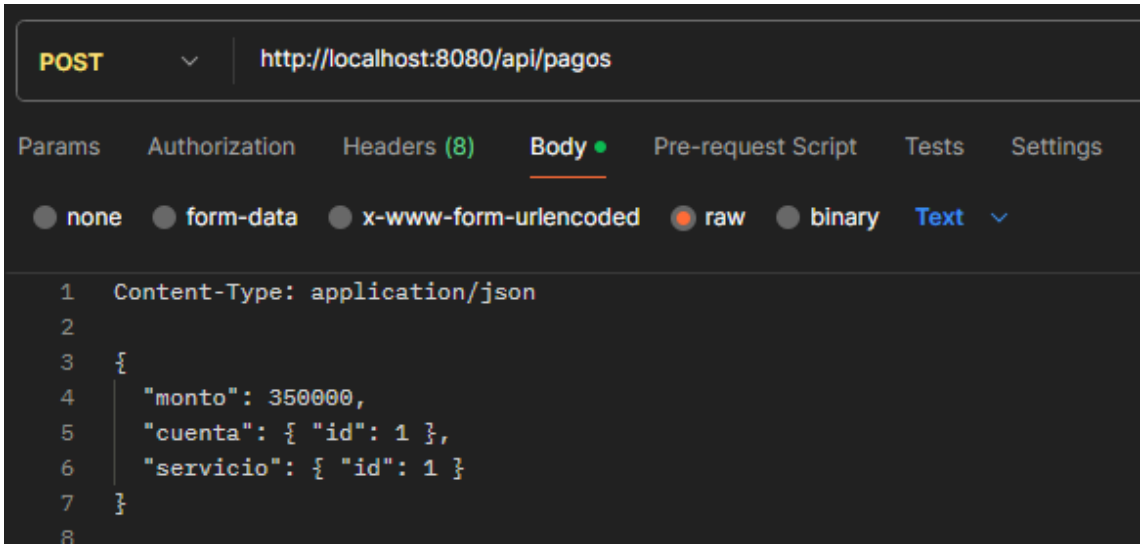


Java Web

Transferencias



Pagos



Probar las demás endpoint