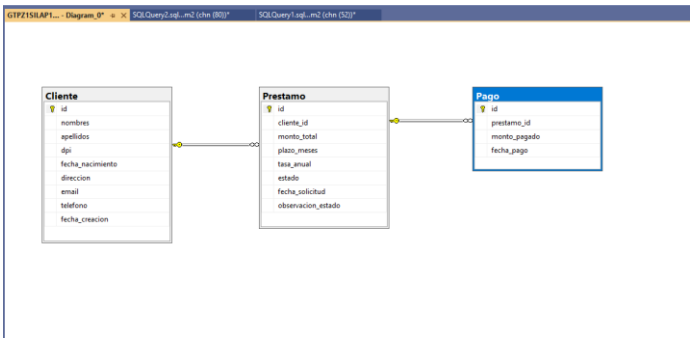
T-SQL Entidad Relación BD ch_exam2



Creación de Tablas (Cliente, Pago, Prestamos)

```sql
USE [ch_exam2]
GO

/****** Object:  Table [dbo].[Cliente]    Script Date: 24/09/2025 22:44:58 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Cliente](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [nombres] [nvarchar](100) NOT NULL,
    [apellidos] [nvarchar](100) NOT NULL,
    [dpi] [varchar](20) NOT NULL,
    [fecha_nacimiento] [date] NULL,
    [direccion] [nvarchar](200) NULL,
    [email] [nvarchar](150) NULL,
    [telefono] [nvarchar](30) NULL,
    [fecha_creacion] [datetime2](7) NOT NULL,
PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
    [dpi] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Cliente] ADD  DEFAULT (sysutcdatetime()) FOR [fecha_creacion]
GO
```

```sql
USE [ch_exam2]
GO

/****** Object:  Table [dbo].[Pago]    Script Date: 24/09/2025 22:45:35 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Pago](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [prestamo_id] [int] NOT NULL,
    [monto_pagado] [decimal](18, 2) NOT NULL,
    [fecha_pago] [datetime2](7) NOT NULL,
PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Pago] ADD  DEFAULT (sysutcdatetime()) FOR [fecha_pago]
GO

ALTER TABLE [dbo].[Pago]  WITH CHECK ADD  CONSTRAINT [FK_Pago_Prestamo] FOREIGN KEY([prestamo_id])
REFERENCES [dbo].[Prestamo] ([id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Pago] CHECK CONSTRAINT [FK_Pago_Prestamo]
GO

ALTER TABLE [dbo].[Pago]  WITH CHECK ADD CHECK  (([monto_pagado]>(0)))
GO
```

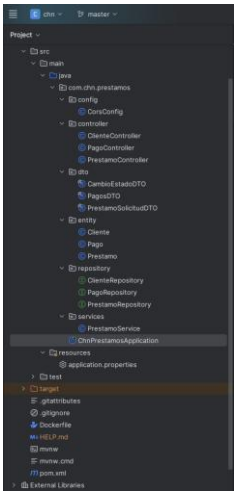
```sql
USE [ch_exam2]
GO

/****** Object:  Table [dbo].[Prestamo]    Script Date: 24/09/2025 22:46:00 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Prestamo](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [cliente_id] [int] NOT NULL,
    [monto_total] [decimal](18, 2) NOT NULL,
    [plazo_meses] [int] NOT NULL,
    [tasa_anual] [decimal](5, 2) NULL,
    [estado] [varchar](20) NOT NULL,
    [fecha_solicitud] [datetime2](7) NOT NULL,
    [observacion_estado] [nvarchar](500) NULL,
PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Prestamo] ADD  DEFAULT ('PENDIENTE') FOR [estado]
GO

ALTER TABLE [dbo].[Prestamo] ADD  DEFAULT (sysutcdatetime()) FOR [fecha_solicitud]
GO

ALTER TABLE [dbo].[Prestamo]  WITH CHECK ADD  CONSTRAINT [FK_Prestamo_Cliente] FOREIGN KEY([cliente_id])
REFERENCES [dbo].[Cliente] ([id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Prestamo] CHECK CONSTRAINT [FK_Prestamo_Cliente]
GO
```

Vista (mostrará el saldo pendiente)

```sql
USE [ch_exam2]
GO

/****** Object:  View [dbo].[vPrestamoSaldos]    Script Date: 24/09/2025 22:51:36 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[vPrestamoSaldos] AS
SELECT
  p.id              AS prestamo_id,
  p.cliente_id,
  p.monto_total,
  COALESCE(SUM(pg.monto_pagado),0) AS total_pagado,
  CASE WHEN p.monto_total - COALESCE(SUM(pg.monto_pagado),0) < 0
       THEN 0
       ELSE p.monto_total - COALESCE(SUM(pg.monto_pagado),0)
  END AS saldo_pendiente
FROM dbo.Prestamo p
LEFT JOIN dbo.Pago pg ON pg.prestamo_id = p.id
GROUP BY p.id, p.cliente_id, p.monto_total;
GO
```

Spring Boot

Rama del Proyecto



Properties

Sirve para configurar la aplicación sin necesidad de cambiar el código de Java.

```
spring.application.name=prestamos

spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=chn_exam;encrypt=false;TrustServerCertificate=true
spring.datasource.username=chn
spring.datasource.password=Password123!
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.open-in-view=false
```

Application

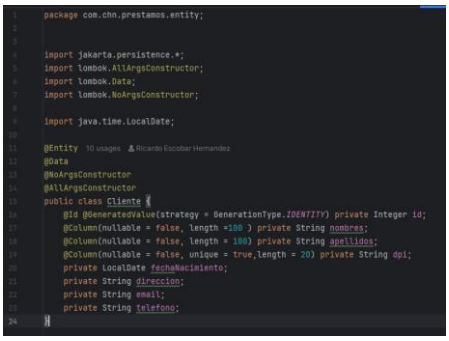
- Es la Clase Principal del Proyecto
- El método main (Donde arranca todo el proyecto)
- Arranca el contenedor de spring Boot
- Busca clases anotadas con @Controller, @Services, @Repository

```
package com.chn.prestamos;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication  Ricardo Escobar Hernandez
public class ChnPrestamosApplication {
    public static void main(String[]args) { SpringApplication.run(ChnPrestamosApplication.class, args); }
}
```

Package Entity

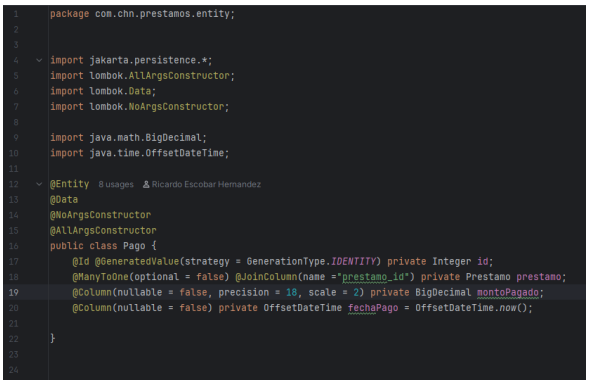
- Cada entidad es una tabla en la base de datos
- Cada atributo de la clase es una columna

@Cliente



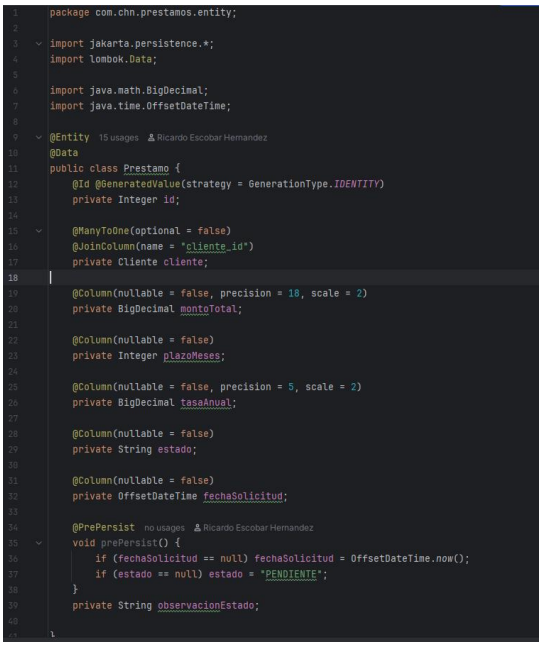
```java
package com.chn.prestamos.entity;


import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;

@Entity  10 usages  Ricardo Escobar Hernandez
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Cliente {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;
    @Column(nullable = false, length =100 ) private String nombres;
    @Column(nullable = false, length = 100) private String apellidos;
    @Column(nullable = false, unique = true,length = 20) private String dpi;
    private LocalDate fechaNacimiento;
    private String direccion;
    private String email;
    private String telefono;
}
```

@Pago



```java
package com.chn.prestamos.entity;


import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.math.BigDecimal;
import java.time.OffsetDateTime;

@Entity  8 usages  Ricardo Escobar Hernandez
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Pago {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;
    @ManyToOne(optional = false) @JoinColumn(name ="prestamo_id") private Prestamo prestamo;
    @Column(nullable = false, precision = 18, scale = 2) private BigDecimal montoPagado;
    @Column(nullable = false) private OffsetDateTime fechaPago = OffsetDateTime.now();

}
```

@Prestamo



```java
package com.chn.prestamos.entity;

import jakarta.persistence.*;
import lombok.Data;

import java.math.BigDecimal;
import java.time.OffsetDateTime;

@Entity  15 usages  Ricardo Escobar Hernandez
@Data
public class Prestamo {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @ManyToOne(optional = false)
    @JoinColumn(name = "cliente_id")
    private Cliente cliente;

    @Column(nullable = false, precision = 18, scale = 2)
    private BigDecimal montoTotal;

    @Column(nullable = false)
    private Integer plazoMeses;

    @Column(nullable = false, precision = 5, scale = 2)
    private BigDecimal tasaAnual;

    @Column(nullable = false)
    private String estado;

    @Column(nullable = false)
    private OffsetDateTime fechaSolicitud;

    @PrePersist  no usages  Ricardo Escobar Hernandez
    void prePersist() {
        if (fechaSolicitud == null) fechaSolicitud = OffsetDateTime.now();
        if (estado == null) estado = "PENDIENTE";
    }
    private String observacionEstado;

}
```

Repository

Es una interfaz que hereda de JpaRepository que tiene como función hacer operaciones en la base de datos sin escribir SQL manual

- Consultas básicas automáticas (findAll, findById, save, delete)
- Guardar entidades (save(Cliente) genera un INSERT INTO CLIENTE(…).
- Actualizar entidades (save(prestamo) genera un UPDATE
- Eliminar entidades (deleteById() genera un Delete

ClienteRepository

```java
package com.chn.prestamos.repository;

import com.chn.prestamos.entity.Cliente;
import org.springframework.data.jpa.repository.JpaRepository;
public interface ClienteRepository extends JpaRepository<Cliente, Integer> {  4 usages  Ricardo Escobar Hernandez

}
```

PagoRepository

```java
package com.chn.prestamos.repository;                                                                                    9

import com.chn.prestamos.entity.Pago;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.math.BigDecimal;
import java.util.List;

public interface PagoRepository extends JpaRepository<Pago, Integer> {  4 usages  Ricardo Escobar Hernandez
    @Query("select coalesce(sum(p.montoPagado),0) from Pago p where p.prestamo.id=:prestamoId")  1 usage  Ricardo Escobar Herna
    BigDecimal totalPagado(@Param("prestamoId") Integer prestamoId);
    List<Pago> findByPrestamoId(Integer prestamoId);  1 usage  Ricardo Escobar Hernandez
}
```

PrestamoRepository

```java
package com.chn.prestamos.repository;                                                                                    3

import com.chn.prestamos.entity.Prestamo;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface PrestamoRepository extends JpaRepository<Prestamo, Integer> {  6 usages  Ricardo Escobar Hernandez
    List<Prestamo> findByClienteId(Integer clienteId);  1 usage  Ricardo Escobar Hernandez
}
```

DTO

Transporta datos entre diferentes capas sin exponer directamente las entidades del dominio o bien los modelos de las bases de datos.

- Se encapsulan datos (Api REST O JSON)
- Separar responsabilidades (no se exponen las entidades en la BD)

Un ejemplo claro es en el caso de un controlador que puede devolver un PrestamoSolicitudDTO en lugar de la entidad completa. Viaja entre capas (controladores ---> servicios ---> frontend)

CambioEstadoDTO

```
package com.chn.prestamos.dto;

public record CambioEstadoDTO(  3 usages  Ricardo Escobar Hernandez
        String estado,  1 usage
        String observacionEstado  1 usage
) {}
```

PagosDTO

```
package com.chn.prestamos.dto;


import java.math.BigDecimal;

public record PagosDTO(  2 usages  Ricardo Escobar Hernandez
        Integer prestamo_id,  1 usage
        BigDecimal monto_pagado  1 usage
) {}
```

PrestamoSolicitudDTO

```
package com.chn.prestamos.dto;

public record PrestamoSolicitudDTO(  3 usages  Ricardo Escobar Hernandez
    Integer clienteId,  1 usage
    java.math.BigDecimal montototal,  1 usage
    Integer plazoMeses,  1 usage
    java.math.BigDecimal tasaAnual  1 usage

){}
```

Service

Se interpreta como una capa intermedia entre el controlador y el repositorio, en pocas palabras es la lógica del negocio (cálculos, condiciones, si un cliente puede o no solicitar un prestamos)

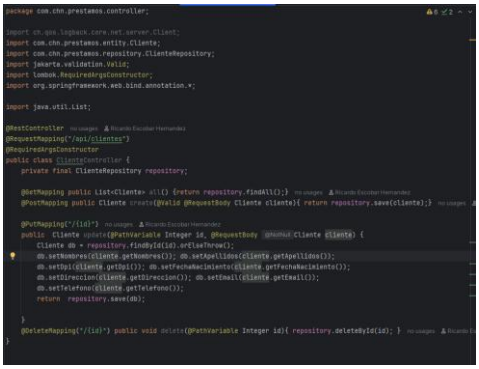
```java
package com.chn.prestamos.services;

import com.chn.prestamos.dto.CambioEstadoDTO;
import com.chn.prestamos.dto.PrestamoSolicitudDTO;
import com.chn.prestamos.entity.Prestamo;
import com.chn.prestamos.repository.ClienteRepository;
import com.chn.prestamos.repository.PagoRepository;
import com.chn.prestamos.repository.PrestamoRepository;
import lombok.Data;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.math.BigDecimal;

@Service  2 usages  Ricardo Escobar Hernandez
@RequiredArgsConstructor
@Data
public class PrestamoService {
    private final PrestamoRepository prestamoRepository;
    private final PagoRepository pagoRepository;
    private final ClienteRepository clienteRepository;

    public Prestamo solicitar( @NotNull PrestamoSolicitudDTO dto){  no usages  Ricardo Escobar Hernandez
        var cliente = clienteRepository.findById(dto.clienteId())
                .orElseThrow(() -> new RuntimeException("Cliente no encontrado"));
        Prestamo p = new Prestamo();
        p.setCliente(cliente);
        p.setMontoTotal(dto.montototal());
        p.setPlazoMeses(dto.plazoMeses());
        p.setTasaAnual(dto.tasaAnual());
        p.setEstado("PENDIENTE");
        return prestamoRepository.save(p);
    }

    public Prestamo cambiarEstado(Integer id, @NotNull CambioEstadoDTO dto){  no usages  Ricardo Escobar Hernandez
        Prestamo p = prestamoRepository.findById(id).orElseThrow();
        p.setEstado(dto.estado());
        p.setObservacionEstado(dto.observacionEstado());
        return  prestamoRepository.save(p);
    }

    public BigDecimal saldoPendiente(Integer prestamoId){  no usages  Ricardo Escobar Hernandez
        var p = prestamoRepository.findById(prestamoId).orElseThrow();
        BigDecimal pagado = pagoRepository.totalPagado(prestamoId);
        BigDecimal saldo = p.getMontoTotal().subtract(pagado);
        return saldo.compareTo(BigDecimal.ZERO) <0 ? BigDecimal.ZERO : saldo ;
    }
}
```

Controller

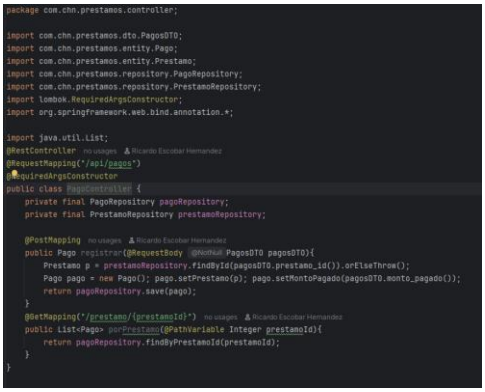
Esta capa que expone la API REST que quiere decir:

- Recibe las peticiones HTTP (GET, POST, PUT, DELET)
- Llama a los services para aplicar la lógica del negocio
- Devuelve respuesta

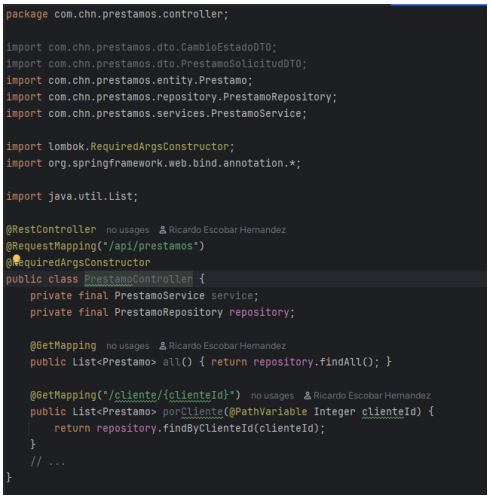
Cliente Controller



PagoController



PrestamosController



DockerFile

Se construye la imagen de la aplicación en Docker esto empaqueta la aplicación con todas sus dependencias la cual crea un contenedor listo para usar.

```
FROM eclipse-temurin:21-jdk-alpine
WORKDIR /app
COPY target/prestamos-0.0.1.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","/app/app.jar"]
```

Docker Compose

Se crea para definir u orquestar múltiples contenedores este us un ejemplo db +SQL+api

Donde primero se crea un contenedor con SQL (db) se conecta a la base de datos SQL y construye la aplicación en spring boot con el Dockerfile.

```
services:
  db:
    image: mcr.microsoft.com/mssql/server:2022-latest
    environment:
      ACCEPT_EULA: "Y"
      SA_PASSWORD: "Password123!"
      MSSQL_PID: "Developer"
    ports: ["1433:1433"]
    healthcheck:
      test: ["CMD","/opt/mssql-tools/bin/sqlcmd","-S","localhost","-U","sa","-P","Password123!","-Q","SELECT 1"]
      interval: 10s
      timeout: 5s
      retries: 20
    volumes:
      - ./db:/scripts:ro


  db-init:
    image: mcr.microsoft.com/mssql/server:2022-latest
    depends_on:
      db:
        condition: service_healthy
    volumes:
      - ./db:/scripts:ro
    entrypoint: ["bash","-lc","/opt/mssql-tools/bin/sqlcmd -S db -U sa -P 'Password123!' -i /scripts/01_schema.sql"]

  api:
    build:
      context: .
      dockerfile: Dockerfile
    depends_on:
      db:
        condition: service_healthy
      db-init:
        condition: service_completed_successfully
    environment:
      SPRING_DATASOURCE_URL: jdbc:sqlserver://db:1433;databaseName=chn_exam;encrypt=false;TrustServerCertificate=true
      SPRING_DATASOURCE_USERNAME: chn
      SPRING_DATASOURCE_PASSWORD: Password123!
      SPRING_JPA_HIBERNATE_DDL_AUTO: none
    ports: ["8080:8080"]
```