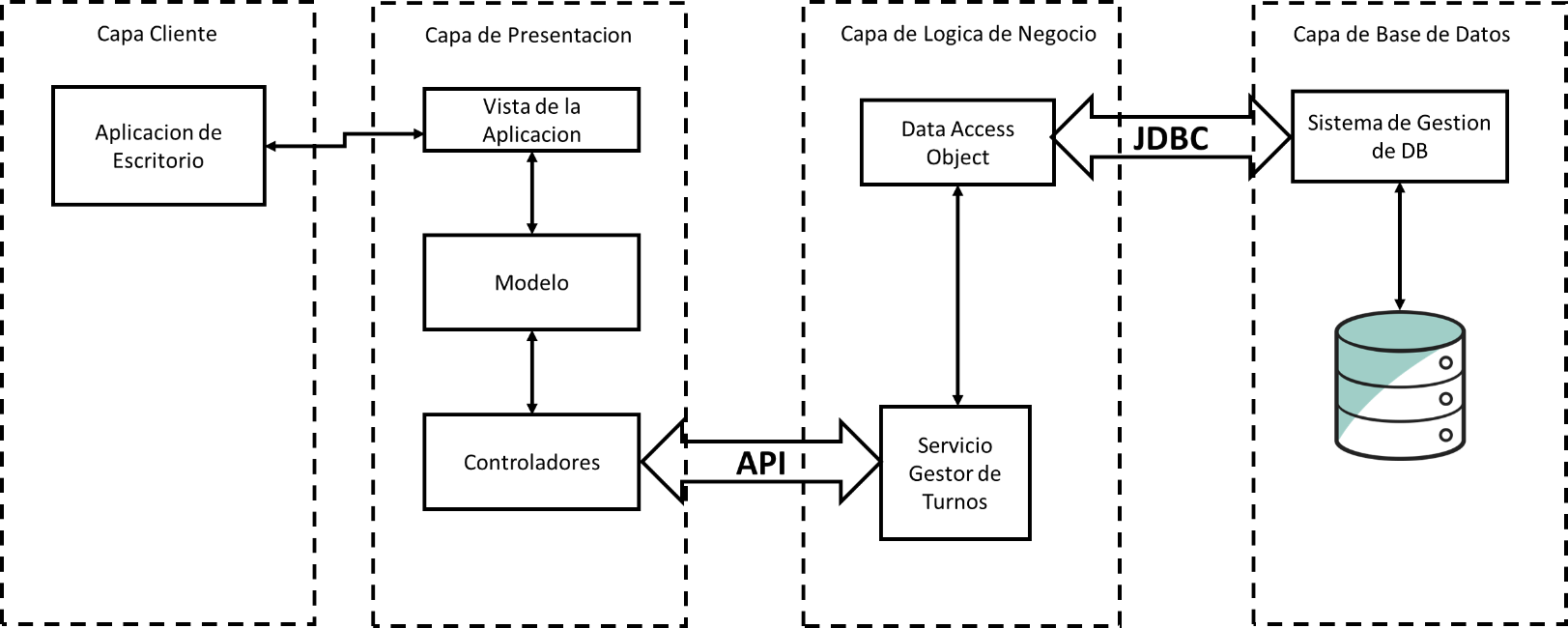
# Diagrama de Capas



# Diagrama de clases del sistema.

A picture containing text, diagram, receipt, line

Description automatically generated

# Modelo de datos (Base de Datos).

A picture containing text, screenshot, font, number

Description automatically generated

# Detalle de las tecnologías a utilizar.

* FrontEnd (Java Swing y Java): Utilizaremos Java Swing para el frontend de la aplicación, lo que nos brindara la interfaz para la parte visual de la aplicación mientras que el código para la arquitectura MVC estará en Java.
* Middleware (Custom API): Crearemos una API Custom que conectara los controladores en la capa de presentación con los servicios en la capa de lógica de negocio. La API también utilizara Java como lenguaje de programacion.
* BackEnd (Java Spring, JDBC, MySQL): Para la lógica de negocios y los servicios de la capa de lógica de negocio usaremos Java Swing, mientras que para la base de datos utilizaremos la API de JDBC para conectar la lógica de negocio con una base de datos MySQL, y tendremos la base de datos almacenada en el localhost del ordenador.
* Testing: Para el testing de la aplicación usaremos el framework de Junit a lo largo de todas las capas de la aplicación.

# Módulos a Desarrollar

* FrontEnd: En el front end nos encargaremos de establecer las vistas y modelos usando Java Swing para las vistas, y el modelo se encargara de armar las solicitudes que se enviaran a la custom API.
  + Pantalla de Log In: La pantalla de log in permitirá ingresar el DNI y la contrasenia del usuario. También tendrá un checkbox para determinar si el usuario es un admin (los pacientes solo podrán ingresar con el checkbox sin seleccionar y los admin solo con el checkbox seleccionado). Al hacer log-in se almacenaran los datos del usuario y se desplegara el frame global.
  + Frame global: Este tendrá una menubar con todas las opciones disponibles del ABMC de Turno. El frame tendrá un panel que ira cambiando según la opción seleccionada.
  + ABMC Turno: Para todas las pantallas del ABMC, si el usuario es un paciente por default la casilla de IdPaciente estará autollenada y no se podrá alterar. En cambio, si el usuario es un admin, la casilla empezara vacia y se podra colocar cualquier id de paciente o dejar en blanco para mostrar todos los pacientes.
    - Alta: Almacenara todos los datos del turno y a la hora de seleccionar el horario del turno, desplegara un panel con todas las fechas que todavía están disponibles.
    - Baja: Solicitara el Id del turno y tendrá un botón para proseguir.
    - Modificación: Funcionara igual que la pantalla de alta con la diferencia de que esta solicitara un ID de Turno para saber cual turno se esta modificando.
    - Consulta: Solicitara un ID de turno y mostrara todos los datos del turno. Si no se ingresa un Id de turno, mostrara todos los turnos agendados (en el caso de un usuario paciente, mostrara todos los datos de ese paciente, en el caso de un admin mostrara todos los turnos del ID del paciente ingresado además de tener un botón que permitirá la descarga de la grilla respuesta).
* BackEnd
  + Controladores: La API conectara directamente el Modelo de aplicación web con el controlador en la capa de negocio. Tendremos requests para cada una de las operaciones ABMC que recibirán un Turno.
  + Servicio gestor de Turnos: Dependiendo de si el turno tiene un IdPaciente seleccionado o no se realizara la operación para un turno/paciente individual o para todos los pacientes (salvo por la baja, esta siempre se realizara a un único turno). Tanto la lógica de regular que un paciente tenga solo 2 turnos por semana como la lógica de no poder cancelar turnos el mismo día estará en este modulo.
  + Data Access Object: Se encargara de conectar la DB con el servicio gestor de turnos y abstraerá la lógica de la base de datos.

# Justificación de la elección de tecnología y consideraciones.

Java es un excelente lenguaje para aplicaciones de escritorio, especialmente gracias a la librería de Swing que permite agilizar el proceso de creación de pantallas, además de que existen librerías como JDBC que facilitar la implementación de una API entre la aplicación y la base de datos, y MySQL es una base de datos bastante simplista que encaja perfectamente con el modelo de datos sencillo que utilizaremos en la aplicación. Con respecto al servidor online, Spring es uno de los frameworks más utilizados para aplicaciones web y de lejos la mejor opción de framework que nos brinda java para la capa de negocio. La única notación importante a hacer es que debido a la naturaleza de la aplicación en la que se está trabajando, en un entorno laboral real el frontend se realizaría en una página web en vez de una aplicación de escritorio y se sustituiría java por algún otro lenguaje como javascript o php, razón por la cual la arquitectura implementa una API custom, ya que el proyecto asume que la aplicación es un prototipo que se está enviando al centro de rehabilitación, de modo que más adelante la capa de negocio se podría montar en un servidor en la nube y funcionaria para conectar tanto con aplicaciones de escritorio como página web.