controladores:
- appointment controler:
url: "api/appointments"
Utiliza
métodos:
getAllAppointments: Obtiene todos los registros guardados de citas, para ello hace una petición al repositorio de pedidos e itera por cada objeto recibido para agregarlos dentro del array de citas que crea el método. Entonces comprueba que el array no se encuentre vacío, en cuyo caso devuelve los códigos HTTP No Content u Ok.

getAppointmentById: acepta por parámetro un input de tipo long que casa con la url usando la anotación de @PathVariable.
Con la creación de un optional el repositorio hace una llamada al metodo correspondiente para buscar citas por id y devuelve lo correspondiente, el optional guardará el resultado tanto si este existe o resulta en un resultado nulo. Entonces comprueba el resultado del optional con su función "isPresent()" para comprobar la existencia del resultado o si es una respuesta nula. En cuyo caso devolverá el status Ok o NotFound dependiendo de la existencia del resultado.

createAppointment: Acepta por parámetro un objeto de tipo Appointment que utiliza la anotación @RequestBody para que al ejecutarse realice la función POST con los datos de la cita.

deleteAppointment: Acepta un parámetro de tipo long que

se enlaza con la url con la anotación "@PathVariable". Primero busca si existe una cita con el id pasado por parámetro, este resultado se guarda en un optional del cual se comprueba el resultado utilizando su función isPresent(), en caso de que exista devuelve el status ok y en caso de no existir devolverá NOT FOUND

deleteAllAppointments: Utiliza el método del repositorio para borrar todos los registros de citas ,
devuelve entonces el resultado ok.

Dentro de todos estos métodos se devuelve un ResponseEntity<> con el Status de la respuesta correspondiente al resultado de la petición.

Para que esto funcione utiliza las anotaciones @GetMapping(), @PostMapping() y @DeleteMapping() con la url correspondiente a el controlador.

REPOSITORIOS:
Los repositorios interfaces que extienden de JpaRepositorio que utilizan <Entidad, Long(ID)>, cada interfaz implementa los métodos findAll(), save y delete usando la entidad correspondiente a la clase. La interfaz RoomRepository aparte define el método findByRoomName y deleteByRoomName ya que el nombre de la habitación es lo que se está definiendo como Id para este objeto.

Entidades:

Appointment:
Posee un id autogenerado por jpa, contiene un objeto de tipo pacient, otro de tipo doctor y un último de tipo room los cuales tienen una relación @ManyToOne definida por la anotación con el tipo CascadeType.ALL y hace referencia a la columna que actúa como id con este mismo nombre excepto en el caso de room que usa roomName. Por último tiene dos atributos de tipo LocalDateTime que contienen la fecha y hora de comienzo y de fin de la cita.
Contiene los getters y setters para los atributos y un método overlaps que acepta por parámetro un objeto appointment con el cuál se vaya a comparar una cita con la pasada por parámetro, en caso de que coincidan las fechas y habitaciones devolverá true porque habrá una coincidencia, en caso de que una cita termine después del comienzo de otra o en el caso de que una cita comience después del comienzo pero termine antes de su finalización devolverá true, en caso de que no se cumpla ninguno de los casos devolverá false.

Doctor:
Crea la tabla doctors usando la anotación @Table en la que le establece el nombre "doctors", esta clase extiende de la clase Person, donde se definen los atributos firstName, lastName, age y email, en la clase doctor se define el id de tipo long autogenerado por jpa, la clase contiene sus getters y setters y ningún método extra.

Pacient:
Crea la tabla pacients usando la anotación @Table en la que

se establece el nombre "pacients", esta clase extiende de la clase Person donde se definen los atributos firstName, lastName, age y email.

Person:
Sirve como clase padre para los tipos Pacient y Doctor, esta clase usa la anotación @MappedSuperclass que sirve para que jpa incluya los atributos de manera persistente como si fueran definidos directamente en las clases hijas

Room:
Objeto que utiliza el nombre de la habitación como identificador gracias a la anotación @Id, de esta manera jpa reconocerá este atributo como la tabla de identificación a la que podrá hacerse referencia desde la tabla appointments.

JacksonConfiguration:
Se utiliza para establecer la configuración en el formato de las fechas, spring lo reconoce como un archivo de configuracion por la anotación @Configuration y utiliza un @Bean para enlazar dentro de esta clase el objeto Jackson2ObjectMapperBuilderCustomizer que es usado para crear el constructor del formateador y deserializador de fechas.

ServletInicializer:
LLama a la clase que actúa como main definida con la anotación @SpringBootApplication (TechhubApplication.java) para ejecutarla a través de un servlet de Springboot.