# BACK-END

## I [Introducción a Spring Boot, Estereotipos e Inyección de dependencias]

### <https://docs.spring.io/spring-boot/docs/1.5.1.RELEASE/reference/html/getting-started-system-requirements.html> ***versiones*** Java-SpringBoot-Spring-Servlet [22:54]

### Objetivos del curso [32:07]

### Configuración Variables de entorno [44:44]

### Versiones y ediciones del JDK [46:46]

### Configuración JDK en Eclipse [55:07]

### Formas de crear u nuevo proyecto [56:06]

### Explicando el archivo ***pom.xml***

### application.properties configuración

### Appendix ***application.properties*** [*https://docs.spring.io/spring-boot/docs/current/reference/html/appendix-application-properties.html*](https://docs.spring.io/spring-boot/docs/current/reference/html/appendix-application-properties.html)

### DevTools librería [1:15:00]

### Integer o int en propiedad [1:16:00]

### Ctrl + 1, Implementar métodos de una interfaz

### @RestController [1:25:40]

### Ctrl + Shift + O importar

### Inyección de dependencias [1:28:50]

### @Qualifer, DESACOPLAR CODIGO, SCOPE singleton | prototype [1:35:00]

### @Configuration [1:37:34]

### Estereotipos @Repository @Service @Component @Controller [1:38:44]

### @ComponentScan [1:47:00]

### *PERSONALIZAR* *BANNER* DE *CONSOLA* [1:50:40], <https://devops.datenkollektiv.de/banner.txt/index.html>

## II [Inicio proyecto curso, spring JPA Repository, Servicios REST]

### Modelo de base de datos de la aplicación [00:4:10]

### Perspectiva del proyecto back-end es del tipo monolito [00:8:17]

### Perspectiva front-end [00:8:40]

### Perspectiva global del proyecto [00:10:46]

### Diagrama de despliegue del back-end [00:10:47]

### Pantalla de la aplicación [00:11:40]

### Creación del proyecto del curso [00:13:41]

### Configuración de la conexión a la base de datos [00:22:04]

### **JPQL** (java persistence query language) [00:24:06]

### Jpa-hibernate [00:29:24]

### MODEL

#### @Entity, @Table, @Id, @*GeneratedValue*, @Column, nomenclatura de nombre campos de la tabla [00:34:49]

#### Alt + Shift + S búsqueda en google (STS shortcuts) atajo de teclado generar accesores en el bean

#### REFERENCIA A VIDEOS DE GENERICOS mitocode Java 7 se AVANZADO | 3 videos genéricos

### DAO

#### JpaRepository, @Repository [00: 50:45]

### SERVICE [00: 54:58], NOTA: se accesan métodos que no devuelven referencias, ejemplo dao.getOne (int id) devuelve una referencia al objeto y esto marcara errores de ejecución

#### OPTIONAL CLASS [1:02:11]

#### JBOOSTOOLS En MarcketPlace de Eclipse para hacer ingeniería Inversa BD 🡪 Java Clases [1:06:44]

#### @Service [1:08:00]

### CONTROLLER

#### @RestController, @RequestMapping, @Autowired, @GetMapping [1:08:36]

#### EJCUTAR Y PROBAR APP [1:11:15]

#### @PathVariable [1:14:00]

#### @PostMapping [1:16:05]

#### @PutMapping [1:16:30]

#### @DeleteMapping [1:16:45]

#### POSTMAN PRUEBAS [1:17:20]

#### PRODUCES, CONSUMES en @GetMapping [1:24:34]

#### Modelo de madurez Richardson <https://martinfowler.com/articles/richardsonMaturityModel.html>

#### DUDAS SOBRE TEMAS DEL CURSO [curso@mitodode.com](mailto:curso@mitodode.com)

## PARTE III [Spring DATA JPA (Mapeo de tablas)]

### ***Medico.java***, @Entity, @Table, @Id, @GeneratedValue, @Column [0:00:52]

#### JPQL [0:05:12]

### Examen.java [0:07:37]

### ***Menu.java*** [0:10:23]

#### MATERIAL ICONS <https://material.io/resources/icons/?style=baseline> [0:11:00]

### Especialidad.java [0:12:09]

### ***Rol.java*** [0:13:15]

### Consulta.java [0:15:00]

#### Trabajando con esquemas [0:15:25]

#### La propiedad paciente

##### Se usa “**private Paciente paciente** “y no Integer idPaciente porque se va a utilizar sentencias de JPQL porque esta sentencias son orientadas a objetos

##### ***@ManyToOne*** Paciente paciente. Para “mapear” indicar que un paciente puede tener una o muchas consultas;

##### En la clase paciente no necesita la anotación @OneToMany private List<Consulta> consulta; Porque este no es un caso de maestro detalle [18:26]

#### Mapeo Paciente @JoinColumn(name = "id\_paciente", nullable = false, foreignKey = @ForeignKey(name ="fk\_consulta\_especialidad")) Donde: **id\_paciente** es el campo que se creara al crearse la tabla así como el nombre de llave foránea [28:56]

#### @OneToMany List<DetalleConsulta> **DetalleConsulta** se trata del caso maestro-detalle; mappedBy = ”**consulta**”, donde consulta hace referencia a la **propiedad** **consulta** en la clase **DetalleConsulta.java** [46:47]

### ***Medico.java*** el mapeo con consulta es análogo al mapeo de *consulta-paciente* [0:33:40]

### ***Especialidad.java*** el mapeo con consulta es análogo al mapeo de *consulta-paciente* [0:33:45]

#### **LocalDate** y **LocalDateTime**; SpringBoot 2.x 🡪 java8 o posterior entonces jsr310, json ISODate YYYY-MM-DDHH:MM:SS [0:36:34]

### DetalleConsulta.java [0:47:00]

#### @ManyToOne private Consulta consulta, por estar en el esquema de maestro-detalle [0:46:48] revisar [46:47]

### ConsultaExamen.java [1:09:10]

#### Llave primaria compuesta, en forma **manual** la otra es @ManyToMany [1:10:10] Referencia de lectura: [www.objectdb.com](http://www.objectdb.com)

#### Las referencias foráneas a las llaves compuestas se hacen en la clase **ConsultaExamenPK.java** [1:12:25]

#### Para que el contenido de la llave compuesta sea único se deberá sobre escribir el método equals [1:28:40]

#### @ManyToOne de la tabla que corresponde

#### Alt+Shift+s y seleccionamos Generate hashCode() and equals()… muestra una advertencia que se están sobre escribiendo objetos “Consulta consulta y Examen examen” y no Integer, por lo que hay que sobre escribir en cada objeto “Consulta.java y Examen.java” solamente en el atributo que los hace únicos “id” los métodos Generate hashCode() and equals()…

### **Usuario** - Rol [1:35:05]

#### @ManyToMany el mapeo se puede hacer en cualquiera de las dos tablas [1:36:36]

#### @JoinTable generar una tabla intermedia de nombre “**usuario\_rol**” y @JoinColumn creara la columna “**id\_usuario**” que estará referenciando a la valor de la PK en idUsuario en Usuario.java y por otro lado con **inverseJoinColumns** se creara el campo **id\_rol** referenciando al valor de la PK idRol en Rol.java

#### **FetchType** por defecto es lazy ‘tardía’, con EAGER obtendremos los elementos anidados de en una consulta, con lazy no [1:47:20]

### **Menu - Rol** caso similar a *Usuario – Rol* [1:51:22]

### Maven Install si se ha hecho un Maven Clean

### Para generar las clases a partir de las tablas de la base de datos se debe instalar el plugin jboos tools 4.15.Final desde eclipse Marketplace 🡪 clic derecho sobre el proyecto y el menú desplegable seleccionar jpa tools 🡪 generar entidades desde la base de datos 🡪 proporcionar parámetros de conexión a la BD 🡪 seleccionar tablas

## IV [Buenas prácticas en servicios REST (Validaciones, Excepciones, Documentación, Pruebas Stress, Richardson)]

### Buenas Prácticas [0:07:24]

#### Validaciones, https://www.baeldung.com/javax-validation dependency Spring-boot-starter-validation [0:08:56]

##### Paciente @NotNull, @Size, @Email, @Pattern, @Valid [0:09:20]

##### Idioma de mensajes de error message.properties [0:17:45]

##### MessageConfig.java @Configuration [0:19:55]

##### Prueba de funcionamiento UtilController.Java [0:24:37]

##### Conexión a múltiples data sources [0:28:40]

##### ResponseEntity [0:30:10]

##### Consumir Servicios Rest desde otro servicio rest [0:41:50]

##### Manejo de Errores [0:42:44]

##### Excepciones personalizadas, @ResponseStatus [0:44:26]

##### Formato de mensaje de error, ExceptionResponse.java [0:49:00]

##### Manejador de errores ‘Handler’ ReponseExceptionHandler.java Alt+Shift+s implementar [0:49:50]

##### @ControllerAdvice interceptor de errores [0:51:12

#### Documentación [1:07:34]

##### Agregar librerías la pom.xml springdoc-openapi-core, classgraph, springdoc-openapi-iu

##### Agregar información adicional @Schema [1:13:00]

#### Pruebas de Stress [1:15:10]

#### Modelo de madures de Richardson Teoría [1:20:54]

##### Bloque de localización en PacienteController.registrar [1:24:40]

## V [Spring Hateoas, Refactoring, Transacciones en Spring]

### Nivel 3 Modelo de Richardson Metada en los Servicios

#### Dependencias application.properties SpringHateoas [0:1:50]

##### HipperMedia PacienteController.java listarPorId @RequestParam [0:02:38]

### Uso de genéricos, refactorizando las clases del paquete DAO [0:12:55]

#### IGenericRepo.java @NoRepositoryBean [0:15:15]

### Refactorizando la capa DAO ICrud.java [0:20:30]

### Refactorizando la capa Service [0:24:00]

### Una vez terminado el paso anterior lo único que hay que hacer es copiar y pagar cambiándole el nombre correspondiente a cada objeto en la capa correspondiente [0:41:50]

### Hacer pruebas para cada Controller en POSTMAN [0:45:45]

### Para facilitar las pruebas en Objetos anidados <http://www.jsonschema2pojo.org/> [0:55:25]

### Maestro-Detalle Inserción de información [0:58:45]

### Para evitar inserción infinita @JsonIgnore [1:03:33]

### Agregando método registrarTransaccional para insertar en Maestro detalle [1:06:05]

### Insertar en tabla Muchos-Muchos ‘Consulta-Examen’ @Query [1:13:35]

### Definición y uso de DTO ConsultaListaExamenDTO.java @Modifying, @Transactional [1:20:40]

### Moviendo @Transactional a un ámbito Scope mayor de lógica de negocio Service, para que al insertar se inserte todo o nada [1:32:50]

# FRONT - END

## VI - Introducción a Angular 10, Encuadernación y Comunicación entre componentes

#### Instalación de angular <https://www.youtube.com/watch?v=Ho7GazIFYTM>

### En línea de comando ejecute: npm install -g npm después npm install -g @angular/cli

### Integrar GitBash a Visual Studio Code https://www.youtube.com/watch?v=EVh36sDppcs

### Comprobar la instalación de Angular CLI verificando la versión instalada *ng –version*

### Instalar o actualizar a la última versión disponible de angular CLI desde línea de comandos npm install –g @angular/cli [0:09:40]

### Generar un nuevo proyecto desde VSC con la consola bash: ng new demo [0:10:12]

### Routing [0:10:40]

### Plugins para VCS [0:12:30]

### Reconstrucción de la carpeta node\_modules, dentro del proyecto **npm install** [0:18:35]

### Dependencias Angular [0:21:21]

### Complementos IntelliSence for class names [0:24:08]

### Ejecutar el proyecto dentro de la carpeta del proyecto **ng serve -o | ng s –o** [0:25:10]

### Descripción de las carpetas del proyecto [0:28:20]

### String Interpolation Sintaxis [0:38:03]

### Componentes [0:40:38]

### Crear un componente de forma manual [0:41:40]

### Autocompletar **a-component** genera la plantilla del componente

### Crear el componente de forma automática mediante línea de comando ng generate component compb –skipTests | ng g c compb --skipTests [0:49:50]

### Ejemplo de una calculadora [0:54:20]

### PRINCIPALES COMANDOS

#### Instalar Angular CLI (ejecutar esto solo una vez por maquina): npm install -g @angular/cli

#### Ejecutar proyecto angular ng serve | ng serve -o | ng s | ng s -o

#### Crear un nuevo componente: ng generate component <nombre\_componente> | ng g c <nombre\_componente>

#### Crear un nuevo componente sin el archivo pruebas: ng g c nombre\_componente --skipTests

#### Descargar nuevamente dependencias para la carpeta node\_modules: npm install | npm i

### Concepto One Way Binding [0:55:30]

### Eventos en Angular [0:55:10]

### Concepto Two Way Binding, ngModel [1:09:50]

### Comunicación entre componentes caso 1 de Padre a Hijo @Input() [1:15:21]

### Caso 2 de Hijo a Padre [1:21:45]

## VI - Angular UI (Material, PrimeNG, Bootstrap), Inicio proyecto FRONT-END

### AngularMaterial, Bootstrap, PrimeNG, (todos son response en dispositivos diferentes) [0:01:24]

#### Generando los proyectos ng new ang-bootstrap, ng new ang-material, ng new ang-prime [0:01:15]

### Instalar las librerías Bootstrap **npm install Bootstrap jquey Popper.js** --save [0:11:40]

### Instalar las librerías de Angular Material [**https://material.angular.io/components/categories**](https://material.angular.io/components/categories)[0:16:00]

### Instalando librerías de PrimeNG <https://material.angular.io/components/list/overview> [0:25:55]

### Creación del proyecto del curso ng new mediaapp-frontend [0:33:10]

#### Atajo de teclado Ctrl+p para burcar archivos

#### Agregar Angular Material al proyecto **ng add @angular/material** [38:30]

#### Crear la carpeta src/app/**pages** contendrá todos los componentes (paciente, medico, etc.) a lo largo del proyecto [41:43]

#### Crear el componente para paciente ng g c pages/paciente –skipTests [41:50]

#### *Crear el componente para medico* **ng g c pages/medico –skipTests** [42:22]

#### Crear la carpeta \_model almacenara los objetos del modelo ejemplo paciente.ts [43:44]

#### Crear la carpeta \_service almacenara los objetos de conexión service a los servicios del backend

#### Crear la clase modelo **Paciente** con sus atributos dentro de **src/app/\_model/paciente.ts** [45:00]

#### Crear el archivo de lógica para hacer la conexión con el servicio rest **ng g s \_service/paciente –skipTests** [47:27]

#### Implementar el constructor en paciente.service.ts constructor (private http: HttpClient) {}; hacer la importación correspondiente import { environment } from 'src/environments/environment'; [51:00]

#### *Implementar el método* **listar** *en* **paciente.service.ts** return this.http.get<Paciente[]>(‘http:localhost:8080/paciente’); *[52:20]*

#### Invocar al método listar de paciente.service.ts desde paciente.component.ts mediante inyección de dependencias [0:54:35]

#### Importar HttpClientModule en app.module.ts para poder utilizarla en paciente.service.ts[0:59:59]

#### Utilizando programación reactiva Rxjs y manejo de flujos pipe [1:02:00]

#### Establecer la comunicación entre aplicaciones front-end y back-end, implementando la clase src/main/java/com.mitocode.util.CORS.java en la aplicación backend mediapp-backend [1:07:38]

#### Enviar la data obtenida por paciente.component.ts a través de paciente.service.ts a la vista del componente paciente.component.html [1:15:45]

#### Crear la variable de entorno HOST que apunte a la ruta común ‘http://localhost:8080’en enviroments.ts y refactorizar la url en el método listar de la clase PacienteService en paciente.service.ts [1:20:00]

#### Generar un nuevo módulo para almacenar elementos de angular material **ng g m material** esto generara el modulo material.module.ts es como un app.module pero de menor ámbito [1:24:09]

#### Importar el modulo recién creado en app.module.ts [1:24:50]

#### *Copiar el código del import para la tabla con filtro desde la página oficial de angular material* [https://v7.material.angular.io/components/table/api *en material/material.module*](https://v7.material.angular.io/components/table/api%20en%20material/material.module)*.ts [1:23:31]*

#### Copiar el código de la tabla filter desde https://v7.material.angular.io/components/table/api al componente paciente.componente.html y adaptarlo al requerimiento [1:31:10]

#### Refactorizar mover el import MatTableModule a la sección de export en material.module.ts para que pueda ser referenciado en paciente.component.html [1:30:35]

#### Refactorizar paciente.component.ts para pasar el data por la variable dataSource hacia paciente.component.html [1:34:35]

#### Establecer las columnas de la tabla que serán mostradas por medio de la variable displayedColumns

#### Agregar un suscriptor a listar pasándole la data a través del MatTableDataSource (data);

#### Agregar el estilo del componente tabla filter indicado en la página de material en el componente paciente.component.css [1:24:40]

## VIII [Angular CRUD, Uso de Observables y Subjects]

### El archivo ***app-routing.module.ts*** se genera cuando se crea el proyecto, se usa para configurar las rutas de los módulos del proyecto este a su vez queda declarado en el archivo ***app.module.ts*** [0:00:10]

### Implementar la página principal de aplicación ***app.component.html*** definiendo las opciones de navegación [0:01:10]

#### Importar los elementos que se usaran en ***app.component.html*** MatButtonModule, MatSidenavModule, MatIconModule, MatMenuModule, MatToolbarModule, MatDividerModule en *material.module.ts* [03:05]

#### Implementar los estilos para la página principal en ***app.component.css,*** flex: 1 es para hacer los elementos equidistantes [08:21]

#### Toolbar Angular Material para la barra de menú con un botón de icono de menú y close validados con \*ngIf y controlando la barra lateral #sidenav mediante el clic asociado al botón con el evento toggle del botón [0:09:40]

#### Agregar un botón de icono more\_vert ‘tres puntitos verticales’ desencadena un matMenuTrigger que abre el menú asociado el routerLink para navegar a otro componente y cerrar sesión con clic sidenav.opened = false [0:18:36]

#### Concepto de Routing archivo de ruteo app.routing.module.ts y la etiqueta <router-outlet> [0:27:25]

#### Barra lateral mat-sidenav-container con efecto tipo push y las opciones de menú con ma-divider, al hacer clic en una opción de menú ejecuta el sidenav.toggle, routerLink es la opción de navegación hacia un componente, cursor: pointer es para ver el cursor como manita [18:38]

#### Al final del menú se tiene el componente <mat-sidenav-content> que contiene el componente que va a ir cambiando [26:50]

#### Al contrario de las aplicaciones web tradicionales en vez de ir de una página a otra, aquí se debe ir al componente porque tiene su html y css que lo representa. Por lo tanto para poder definir la navegación se configuran las reglas de navegación en ***app.routing.module.ts*** como se muestra a continuación

##### Regla 1. Interceptar la ruta ***path: ‘paciente’, component: PacienteComponent*** cuando el usuario escribe en la url y declarar el componente responsable; Regla 2. Para medico seria ***path: ‘medico’, component: MedicoComponent*** estas reglas sufrirán cambios para hacerlas más dinámicas conforma avance la implementación del proyecto [29:00]

##### Entre las etiquetas <mat-sidenav-content> <router-outlet></router-outlet></mat-sidenav-content> para hacer que las reglas de navegación sean tomadas en cuenta y el componente a mostrar sea el indicado en las reglas de navegación [30:40]

#### Implementar la inserción, modificación y eliminación de paciente en *paciente.component.html* [0:32:05]

#### Agregar el botón icono créate con el routerLink= “/paciente/nuevo” este será un caso de CRUD con navegación en medico será con diálogos modales, registra con combo Box, Registrar E con autocompletes, Registrar W con wizards [34:10]

#### Agregamos la sub ruta paciente/nuevo en el app-routing.module.ts con children: [path: component: paciente-edicion-component.ts]

#### Generar el componente para nuevo paciente ng g c pages/paciente/paciente-edicion –skipTests

#### Insertar una etiqueta <router-outlet> al inicio de paciente.component.html para para permitir la navegación hacia sus hijos es decir elementos de la lista, La etiqueta <router-outlet> se debe colocar en toda página padre para navegar a hijos [40:40]

#### Implementar el css de paciente-edicio.component.css

#### Para trabajar con formularios agregar el ***ReactiveFormModule*** en app.module.ts

#### Implementar el formulario paciente-edicion.component.html, importar las librerías de mat-form-field de angular material en el *material.module.ts* [48:30]

#### Crear el objeto que representa los elementos del formulario de *paciente-edicion.component.html* en el método ***ngOnInit*** de *paciente-edicion.component.ts* [59:30]

#### Agregar los botones para editar el registro seleccionado en paciete-component.html en el campo de acciones [1:04:19]

#### Implementar ***listarPorId*** en paciente.service.ts [1:07:40]

#### Enviar el id del elemento seleccionado al hijo en este caso a paciente-edicion.component.ts mediante una variable en el constructor mediante la clase ActivatedRoute la cual permite saber cuál es la url activa es ese momento [1:09:14]

#### Implementar la suscripción a la variable anterior la cual e un observable para recuperar el id del registro seleccionado en la url dinámica utilizando una variable de apoyo id [1:10:20]

#### Utilizamos una variable de apoyo edicion: boolean para determinar si se trata de nuevo o actualizar registro eso se determina si en la url hay o no un parámetro [1:11:36]

#### Implementar initForm con formulario en blanco o con parámetros dependiendo de la variable booleana [1:12:48]

#### Confirmación de la operación de inserción / modificación [1:15:10]

#### Snackvar para validación del evento insertar / actualizar [1:32:12]

## IX - [SwitchMap, MatSort, MatPaginator, Angular Modal Dialog, Route Children Validación]

### Refactorización de modificar y registrar el proceso paciente-edicion.component.ts [0:02:20]

#### Suscribir observables, más de uno utilizar pipe (switchMap) [0:05:25]

#### Implementando método eliminar paciente [0:14:35]

#### Búsqueda en la tabla [0:20:05]

#### Filtro de la tabla pacientes con @ViewChild(MatSort) sort: MatSort; [0:26:15]

#### Ordenamiento en columnas @ViewChild de la tabla [0:28:08]

#### Implementando paginación en la tabla de datos [0:33:40]

### Implementación *consultar* *modificar* y *eliminar* Medico utilizando dialogo modal [0:43:20]

#### Generación del service Medico [0:45:22]

#### Implementando la ventana modal [0:48:00]

#### Enviando la data a la ventana modal de medico [1:01:37]

#### Implementado la acción cancelar de los botones de la ventana modal [1:05:03]

#### Implementado la acción modificar y registrar con el botón aceptar de los botones de la ventana modal [1:11:20]

### Implementación Examen [1:21:45]

#### Rutas, Routing [1:22:40]

#### Examen model [1:24:12]

#### Examen service ng g s \_service/examen –skipTests [1:24:37]

#### Refactorizando con una clase genérica el CRUD de los services ng g s \_service/generic –skipTests [1:25:09]

#### \_service/generic para refactorizar todos de los Services listar, listar por id, registrar, modificar y eliminar [1:25:00]

### Implementando Especialidad, al editar o nuevo registro se oculta al padre [1:35:00]

#### Ocultar el padre en nuevo o edicion del registro [1:39:50]

#### Generando la carpeta del modelo para Especialidad y su service [1:36:12]

#### Internacionalización de la barra paginador [1:43:40]

## X - Angular Maestro Detalle (Angular Select, Datepicker, List, Accordion, Autocomplete)

### Maestro-Detalle de ‘consulta’ a las tablas **consulta**, **detalle-consulta**, **consulta-examen** [0:0:40]

#### Componente consulta **ng g c pages/consulta --skipTests** [0:00:48]

#### Agregar al routing el componente recién creado consulta [00:57]

#### Crear el modelo **consulta.ts** [1:57]

#### Crear la clase **detalleConulta.ts** [3:50]

#### Crear el servicio **ng g s \_service/consulta --skipTests** [4:25]

#### Método registrar transacción en consulta.service.ts [6:00]

#### Modelo **consultaListaExamenDTO.ts** [07:20]

#### **consultaComponent.html** [08:48]

#### Componente select de material [9:20]

#### **consulta.component.ts** [11:40]

#### Creando el modelo consulta.ts [0:01:57] El objetivo de este componente es poder hacer la inserción en la tabla consulta, detalle-consulta y en la tabla consulta-examen esta ultima de manera opcional.

#### Generando el service ng g c \_service/consulta [0:4:33]

#### Elementos de Angular Material: Select, ComboBox, Datepicker, ModelJs [09:25]

#### Observable a nivel HTML [16:24]

#### Calendar [21:48]

#### Envío de datos al backend [56:10]

#### Para el manejo de fechas momento.js referencia: momentjs.com **npm install moment --save** [1:04:50]

### Registrar E Consulta especialidad elementos Angular Material Autocomplete [1:20:25]

#### ng g c pages/consulta-especial --skipTests [1:24:00]

#### Reglas de navegación [1:24:14]

#### Explicación del funcionamiento del Autocomplete de paciente [1:26:10]

## XI - Angular Stepper, FlexLayout, NgClass

#### Explicación del autocomplete de medico [02:00]

#### Refactory al autocomplete consulta-especialidad.component.ts [02:25]

#### Mezclando ***TwoWayBinding con Form*** [15:05]

#### Registro consulta especial NgModel con formGroup [16:40]

#### Evento del ComboBox @Output selectionChange consulta-especial.component.ts [21:10]

#### Selección de ComboBox anidados [22:10]

## XII - Wizard Registrar ng g c pages/consulta/wizard --skipTests [34:22]

### CRUD Consulta con Wizard [34:25]

### Generar componente ng g c pages/consulta/wizard --skipTests [34:57]

### Registro de wizardComponent en el app-routing.ts [35:00]

### Children o no children en app.routing.component del registro de consulta con Wizard [35:10]

### Implementando wizard.component.ts [37:30]

### Slide-toogle [38:30]

### Slide-toogle y Stepper control [42:30]

### Mat-card [1:05:34]

### Instalación de Angular Flex Layout npm install @angular /flex-layout --save [1:07:50]

#### Mat-card para médicos [1:20:50]

#### Botón para ir al siguiente elemento del layout en el wizard [1:22:50]

#### Paso 2 Selección del consultorio [1:25:35]

### Componente Buscar [30]

#### Generando el componente ng g c pages/buscar –skipTest

#### ConsultaController.java y Buscar.component.ts, @Query para Consulta

#### Consulta de pacientes por dni [8:10]

#### Diferencia entre @Entity y Nombre de la Clase en el model [9:35]

#### Query LPQL en el backend consultaRepo.java [36:00]

#### IConcultaRepo.java [11:06]

#### Implementando buscar.component.ts [37:39]

#### Detalle Búsqueda consulta-examen [1:21:00]

#### Asignación JavaScript con tipo de datos [1:18:00]

#### Reporte PDF ng g c pages/reporte --skipTests [1:24:12]

## XIII Chart.js, PDF-Viewer, JPA Procedures

### Refactoring ConsultaController

### Librería Chat.js en [www.chartjs.org](http://www.chartjs.org) para trabajar con gráficos estadísticos [6:32]

### Chart js Npm install chart.js --save [09:51]

#### JPA store procedure para el grafico estadistico [145:00]

#### ConsultaResumenDTO.java [24:04]

#### Arreglo método map en reporte.component.ts [40:15]

#### Chart.js Bug [50:55]

#### Generar Reporte PDF [54:28]

#### Creando un reporte en Jasper Report [1:00:30]

#### Enviando datos al Jasper [1:18:00]

#### Trabajando con Blob y Base64 [1:31:00]

#### Generando el reporte [1:43:00]

#### Descargar Reporte PDF [1:45:50]

## XIV - Angular Base64, Servicios Paginados, Spring Security Backend, JUnit

#### Guardar archivo Multipart-from-data [02:56]

#### Base 64 Imágenes en Angular [12:00]

#### Leer archivo [25:58]

#### Tratamiento Base64 [26:37]

#### Servicios Pageable [38:15]

#### Paginator [45:21]

#### Spring Security + JWT + OAUTH2

#### SEGURIDAD: 1. ***Dependencias en el POM*** (spring-boot-starter-security, spring-security-jwt, spring-security-oauth2) [1:04:43]

#### SEGURIDAD: 2. ***Parámetros de configuración*** en el application.properties

#### SEGURIDAD: 3. ***Clases de configuración de seguridad en com.mitocode*** (AuthException.java, AuthorizationServer.java, ResourceServerConfig.java, SecurityConfig.java) [1:04:20]

#### JUnit Para generar los usuarios de la app en la tabla usuario [1:15:50]

#### Agregando data a las tablas rol y usuario [1:23:50]

#### Probando la creación del token (con el endpoint que la librería oauth2 provee) y la consulta de pacientes con token, ***mediante postman*** [1:24:30]

#### Flujo de trabajo del uso de los tokens y empresas (amazon cognito) que proporcionan el servicio de seguridad con tokens [1:28:27]

#### Hasta este momento ya se puede hacer uso de la seguridad por tokens al menos desde postman.

## XV - Tokens por Roles, Angular Login, Angular JWT Module, Menu por Roles.

#### Explicación de las clases de seguridad (AuthException.java, AuthorizationServer.java, ResourceServerConfig.java, SecurityConfig.java) [00:24]

#### En las dependencias spring-boot-starter-security protege los roles y los endpoints, spring-security-oauth2 para orquestar el token tiempo de vida del token y spring-security-jwt es para generar el token [02:30]

#### Página para generar cadenas aleatorias <https://www.allkeysgenerator.com/Random/Security-Encryption-Key-Generator.aspx> [06:10]

#### Explicación de los parámetros de seguridad en ***application.properties*** [06:30]

#### Alternativas a la seguridad en la aplicación IdentitiAsAservice Okta, keyCloack, AWS Cognito [09:00]

#### Explicación de la clase SecurityConfig.java ***@Autowired en un método*** [14:00]

#### Explicación de la clase ***ResouceServerConfig.java*** sirve para controlar los endpoints permitidos [30:10]

#### Explicación de la clase ***AuthorizationServer.java***, sirve para accionar toda la seguridad [30:30]

#### ***AuthException.java*** para los mensajes de error de logeo

#### Prueba de generación de tokens desde postman [37:54]

#### ***Tiempo de inactividad en la aplicación*** (idleTime) vs tiempo de vida de la sesión [41:30]

#### ***Seguridad con tokens por Rol a nivel de método*** en los controller para proteger los ***endpoint*** por ejemplo, listar en *MedicoController*, se utiliza la anotación @PreAuthorize("hasAuthority('ADMIN') or hasAuthority('USER')") en su método lista [45:50]

#### Parametrizar el acceso a los endpoints Forma para que pueda accesar un rol en específico mediante de una tabla [50:50]

#### ***Seguridad desde Angular:*** Creando el Login [58:15]

#### Login component, mapeo del login en app-routing.module.ts para que el login sea la primera página en mostrarse [1:00:00]

#### Lógica de inicio de sesión, creando el service [1:09:20]

#### Creando el body de la petición de logeo; Petición Post para obtener el token [1:11:20]

#### Generación de la expresión en base 64 a partir del client id y client secret [1:14:38]

#### Client ID y Client Secret en el environment.ts [1:15:40]

#### Envío del token a todos los endpoints para operarlos, con interceptors http o con librerías [1:21:46]

#### Uso de librerías para enviar el toen a los endpoints, instalando la librerías de jwt , npm install @auth0/angular-jwt --save[1:22:15]

#### Opciones de menú por rol [1:33:30]

#### Modificando el backend para obtener la lista de endpoints para el logeo [1:35:00]

#### Consumiendo los endpoints desde el front [1:40:05]

## XVI - Angular Guards, HttpInterceptor, Themeforest, Multilevel menú

#### Guard.component para Proteger las URL’s del menú por el rol asociado, Opciones de Menu por usuario [00:42]

#### Uso de Gurd para interceptar la navegación en las URL’s de la app [03:15]

#### Dentro de Guards no se debe utiliza un subscribe [15:40]

#### Enfoque de la seguridad de la app [18:54]

#### Componente not-404 para la Validación de URL’s 404 página o recurso no encontrado [21:06]

#### Mapeo de not-404 en el app-routing y las url desconocidas [22:27]

#### Ocultar la barra superior [22:58]

#### Cerrar sesión de la app [24:34]

#### Tokens por base de datos en tablas oauth\_acces\_token y oauth\_refresh\_token [27:23]

#### Librerías para controlar el tiempo de inactividad en la aplicación idle monito npm [36:10]

#### Re direccionamiento al login cuando el token haya expirado con ***shared/server-error-interceptor*** interceptando los errores [40:20]

#### Plantillas de aplicaciones themeforest.net [48:47]

#### Revisión de las plantillas de regalos den el curso [57:57]

## XVII - Envío Correos para Restablecer Contraseña, Validación Forms, DigitalOcean

### Revisión de falla en despliegue de los menús de la app [00:10]

### Envío de correos para restablecer la contraseña, por el backend [02:20]

### Agregar dependencias al pom spring-boot-starter-mail y spring-boot-starter-mail, crear las clases Mail.java y EmailUtil.java [07:30]

### Platilla de mail email-template [08:05]

### Archivo application.yml para indicar los parámetros del email [12:21]

### Esquema de restablecimiento de contraseña [18:33]

### Quartz Calendarización de tareas que se ejecutaran en determinados momentos, análogo al programador de tareas de Windows [57:25]

### Validación de campos en el formulario [58:50]

### Teoría Digital Ocean [1:09:23]

### Uso de MobaXterm (se puede usar winssh o putty) para la transferencia de archivos a la máquina (droplet) virtual en digital Ocean [1:28:27]

### Inicializar la imagen de Ubuntu (distribución de Linux) con: ***sudo apt –get Update*** [1:30:00]

### Instalación de Tomcat y Wilfly, descarga y transferencia a Ubuntu, proporcionar privilegios de ejecución a calina.sh con ***chmod 777 catalina.sh***, levantar servidor con ***sh statup.sh*** [1:30:27]

### Instalación de java desde Ubuntu sudo apt-get install open jdk-8-jdk [1:34:51]

### Verificar la instalación de Tomcat en la ip publica proporcionada por digital Ocean y puerto 8080 [1:38:25]

### Re direccionamiento de la ip con un dominio y enmarar el puerto con Nginx instalándolo ***sudo apt-get install nginx*** [1:41:50]

### Edicion de la tabla de Ruteo de nginx a Tomcat [1:44:51]

### Compilación y generación del archivo war del backend, modificar: pom.xml, MediappBackendApplication.java [1:54:30]

### Modificación del front para despliegue. Copiar el contenido del environment.ts al environment.prod.ts, ejecutar el comando ***ng build --prod*** tomara el environment de producción y otros proceso como ofuscación del código y la minificacion (compresión) del código [2:05:14]

### El comando anterior generara una carpeta dist/mediapp-frontend/ que será lo único que utilizara en el despliegue en el servidor [2:11:05]

### Agregando un puto a la ruta en el tag <base href=”./” para que las rutas delas dependencias sean encontradas [2:13:36]

### Implementando una estrategia de navegación para evitar que se pierdan las rutas de acceso a la app al refrescar la pantalla con F5, porque esperan un servlet que resuelva las rutas [2:15:40]

### Agregando un hashtag ‘#’ el app.module.ts para subsanar la perdida de rutas [2:17:12]

### Modificar la clase LoginController.enviarCorreo.java en la parte del envió del correo poniéndole el símbolo # en la ruta quedando así: [***String url = "http://localhost:4200/#/recuperar/" + token.getToken();*** ]

### ***Agregando el front dentro del war para despliegue en servidores (WebLogic)*** [2:23:05]

### Configuración de Wilfly y Digital Ocean [2:27:36]

## XVIII - Spring Cloud Microservicios y Angular 11 Upgrade

### Teoria de Microservicios [00:00]

### Division del backend en dos proyectos distinto. Proyecto 1Todo los relacionado al CRUD. Preoyecto 2 todo lo ajeno al crud [12:00

### Creacion del proyecto springEureka descrubridor de servicios, explora si hay otros microservicios que se pueden registrar[14:40]

### Revision de la docuentacion de Spring Cloud <https://spring.io/projects/spring-cloud> []

### Configuracion y despliegue de SpringEureka [22:11]

### Configuacion de Eureka Client en los proyectos SpringMicro. Dos formas de hacerlo, la primera agregando dependencias al pom.xml y la segunda forma para el proyecto SpringMicro2 con el springStarterProyect agregar las dependencias faltantes[24:40]

### Registro de los proyectos Spring Micro1 y Spring Micro2 en el discovery server del proyecto SpringEureka [27:09]

### Crear el componente Zuul (gateway) que tomara la petecion (url del cliente) para enviarle la peticion a ueureka para que determine cual de los microservicios puede resolver la peticion [31:18]

### Creacion del proyecto SpringGateaay (Zuul)

### Desde angular el cambio es que apunte al gateway para resuelva internamente la peticion [38:57]

### Para el tema de la seguridad se creara el proyecto SpringAuthServer [39:32]

### Secuencia de ejecusion de proyectos: 1-Eureka, 2-Micro1 y Micro2, 3-AuthServer, 4-Gateway [48:31]

### Pruebas de Seguridad desde el postman pero desde el GateWay [49:32]

### Incorporacion de otras tecnologias al proyecto como node.js [51:43]

### Cambio de Url y puerto en en el environment del frontend para que tome los parametros del proyecto Gateway [53:35]

### Migracion del proyecto frontend a la version 11 de angular [1:01:03]

### Forma de migrar a mas de una version de forma gradual [1:01:30]

### Actualizar la version del Angular core y el Angular CLi ***ng update @angular/core@11 @angular/cli@11*** desde la consola VSCode [10:02:00]

### Como nos sale un error, Ejecutamos el comando: ***npm install @angular/cli –g*** y depues ejecutamos el comando anterior [1:03:25]

### Creamos un repositorio en la PC local y ejetamos un commit por si algo le sucede al proyecto durante la actualizacion ***git init*** [1:04:14]

### Ejecutamos los comandos para agregar todo el proyecto al repositorio: ***git add .*** ***git commit –m “commit”*** [1:8:00]

### Ejecutamos la actualización y de paso algunas librería: ng Update @angular/core @angular/cli @angular/material @angular/cdk [1:13:00]

### Cambios en la version de Angular 11 [1:15:00]

## XIX Evaluación Final

## 

