



TECNOLÓGICO DE ESTUDIOS SUPERIORES DE COACALCO

PROYECTO:

INTEGRACIÓN DEL SISTEMA DE ADMINISTRACIÓN

FEMAEPAS

OPCION DE TITULACION:

INFORME TECNICO

RECIDENCIAS PROFESIONALES

ASESOR INTERNO:

Dr.: OLMOS CASTRO ARMANDO

PARA OBTENER EL TITULO DE:

INGENIERIO EN TECNOLOGIAS DE LA INFORMACION Y

COMUNICACIONES

ALUMNOS:

MAZON ROJAS ESTEBAN MICHELL

MATRICULA:

201720098

ABRIL 2024

Índice

Agradecimientos	10
Resumen	11
Capítulo I: Generalidades del Proyecto	13
1.1 Introducción	13
1.2. Descripción de la Empresa y Organización	14
Misión:	14
Visión:	14
Giro y Actividad	15
Estructura Organizacional	16
Ubicación	17
1.2.1 Descripción del Puesto o Área de Trabajo	18
1.3 Justificación	20
1.4 Problemas a Resolver	21
1.5 Objetivos	22
Objetivo General:	22
Objetivos Específicos:	22
1.6 Cronograma de Actividades	23
Capítulo II Marco Teórico	25
2.1. ¿Qué es un Software Administrativo?	25
2.2. ¿Qué es Angular?	26
2.3. Ng zorro	28
2.4. Que es Node.js	28
2.5. Que es el desarrollo Backend	30
2.6. Que es Front end	31
2.7. Que es Una Base de Datos	32
2.8. Que es un Servidor de Correos	35
2.9 Que es la tecnología rest	38
2.10 . Que es una arquitectura cliente servidor	41
Capítulo III Procedimiento y descripción de las actividades realizadas.	43
3.1. Instalación de angular	43
3.2. Instalación del proyecto	44
3.3. Creación del backend	136
Capítulo IV Resultados	174
Conclusiones	193
Competencias realizadas	194
Fundamentos de la investigación	194
Conceptos básicos de fundamentos de investigación como proceso de construcción social	194
Herramientas de la comunicación oral y escrita en la investigación	194

<i>Estudio del desarrollo de su profesión y su estado actual</i>	195
<i>Proceso de elaboración de una investigación documental</i>	195
Fundamentos de programación	196
<i>Fundamentos de programación orientada a objetos</i>	196
<i>Metodología de solución de problemas</i>	196
<i>Acercamiento a las clases y objetos</i>	197
Programación orientada a objetos.....	197
<i>Entorno de desarrollo</i>	197
<i>Clases y objetos</i>	198
<i>Métodos</i>	198
<i>Herencia y polimorfismo</i>	198
Estructura y organización de datos	199
<i>Fundamentos de estructura de datos</i>	199
<i>Estructuras lineales</i>	200
<i>Estructuras no lineales</i>	200
<i>Métodos de ordenamiento y búsqueda</i>	200
<i>Archivos</i>	201
FUNDAMENTOS DE BASE DE DATOS	201
<i>Introducción a las bases de datos</i>	201
<i>Diseño de Bases de Datos con el modelo E-R</i>	201
<i>Modelo relacional</i>	202
<i>Normalización de bases de datos</i>	202
<i>Álgebra relacional</i>	202
<i>Introducción al lenguaje SQL</i>	203
Taller de Base de datos	203
<i>El SGBD y el Lenguaje de Definición de Datos</i>	203
<i>Lenguaje de Manipulación de Datos Competencias</i>	204
<i>Administración de base de datos</i>	204
<i>Tecnologías de conectividad a base de datos</i>	204
Base de datos Distribuidas	205
<i>Fundamentos de bases de datos distribuidas</i>	205
<i>Diseño de bases de datos distribuidas</i>	205
<i>Procesamiento de consultas distribuidas</i>	205
Ingeniería de software.....	206
<i>Introducción a la Ingeniería de Software y Sistemas de Información</i>	206
<i>Modelos de la Ingeniería de Software</i>	206
<i>Análisis del Proyecto de Software</i>	207
<i>Análisis y Modelado del Proyecto de Software</i>	207
<i>Calidad de Software</i>	208
Taller de investigación I	208
<i>Tipos de investigación</i>	208
<i>Estructura del protocolo de Investigación</i>	208
<i>Comunicación del protocolo de investigación</i>	209
Taller de investigación II.....	209

<i>Evaluación y complementación del protocolo de investigación</i>	209
<i>Desarrollo de la metodología del proyecto de investigación</i>	210
<i>Presentación del Informe de investigación.</i>	210
Programación web	210
<i>HTML, XML y CSS</i>	210
<i>programación de lado del cliente</i>	211
<i>Programación del lado del servidor</i>	211
<i>Cómputo en la nube y servicios</i>	212
Bibliografía	212

Índice de Figuras

Figura 1. “Estructura Organizacional FEMAEPAS Durante la estancia”. (Datos Obtenidos de: https://www.femaepas.com)	16
Figura 2. “Referencia de Ubicación”. (Datos Obtenidos de: Federación Mundial Académica y Empresarial de las Áreas y Profesionales de la Salud - Google Maps)	17
Figura 3 “Figura representa la arquitectura de base de datos obtenidos de ; (entorno cliente servidor, s.f.)	42
Figura 4 “comando npm install” (Datos Obtenidos en el Software CMD).	43
Figura 5: “mensaje de instalación al momento de instalar el proyecto”(Datos Obtenidos en el Software CMD).	43
Figura 6. “Versión Actual”. (Datos Obtenidos en el Software CMD).	44
Figura 8 “creación del proyecto”(Datos Obtenidos en el Software CMD).	45
Figura 7 “compilamos el código con ng serve”(Datos Obtenidos en el Software CMD).	45
Figura 9 “muestra cómo se llevara la página de angular y como todavía no hay código te muestra la página de angular””(Datos Obtenidos en el browser localhost angular host4200).	46
Figura 10 “lista de elementos que se crearon”(Datos Obtenidos en el Software visual basic framework angular 12).	47
Figura 11 “muestra el código principal”(Datos Obtenidos en el Software visual basic framework angular 12).	47
Figura 12 “se muestra como se configura un nuevo componente” (Datos Obtenidos en el Software visual basic terminal cmd de visual basic framework angular 12).	48
Figura 13 “el contenido de los módulos y componentes que se usaran”(Datos Obtenidos en el Software visual basic framework angular 12).	48
Figura 14 “se muestra la parte de los componentes que se crearon y el login por default aparece primero” (Datos Obtenidos en el Software visual basic framework angular 12).	49
Figura 15 “partes del código de appcomponent” (Datos Obtenidos en el Software visual basic framework angular 12).	50
Figura 16 “librerías del appcomponent”(Datos Obtenidos en el Software visual basic framework angular 12).	50
Figura 17 “variable de los módulos” (Datos Obtenidos en el Software visual basic framework angular 12).	51
Figura 18 “rutas dentro de los datos” (Datos Obtenidos en el Software visual basic framework angular 12).	52
Figura 19 “librería de ng zorro”(Datos Obtenidos en el Software visual basic framework angular 12).	52
Figura 20 “se exportan las variables de ng-zorro” (Datos Obtenidos en el Software visual basic framework angular 12).	53
Figura 21 “ejemplo de como crear un componente de login”(Datos Obtenidos en el Software visual basic terminal cmd framework angular 12).	53
Figura 22 “creación de los componentes”(Datos Obtenidos en el Software visual basic framework angular 12).	54
Figura 23 “Componentes del login” (Datos Obtenidos en el Software visual basic framework angular 12).	54

Figura 24 “implementación de HTML”(Datos Obtenidos en el Software visual basic framework angular 12).	55
Figura 25 “HTML de login en la parte de validacion de el password” (Datos Obtenidos en el Software visual basic framework angular 12).	55
Figura 26 “importacion de librerias y de variables utilizadas en el proyecto” (Datos Obtenidos en el Software visual basic framework angular 12).	56
Figura 27 “creación de modulo y respuesta de los datos” (Datos Obtenidos en el Software visual basic framework angular 12).	57
Figura 28 “Funcion donde se guardaran los datos la validacion de los campos” (Datos Obtenidos en el Software visual basic framework angular 12).	57
Figura 29 “Impelementacion de la variable del constructor” (Datos Obtenidos en el Software visual basic framework angular 12).	58
Figura 30 “el oinit se usara para la validacion de los campos y la funcion login se mandara a hacer para logear a la persona” (Datos Obtenidos en el Software visual basic framework angular 12).	59
Figura 31 “servicio del login” (Datos Obtenidos en el Software visual basic framework angular 12)	60
Figura 32 “interface del modulo de miembros” (Datos Obtenidos en el Software visual basic framework angular 12).	61
Figura 33 “respuesta de datos” (Datos Obtenidos en el Software visual basic framework angular 12).	62
Figura 34 “HTML de registro”(Datos Obtenidos en el Software visual basic framework angular 12).	63
Figura 35 “Código del HTML” (Datos Obtenidos en el Software visual basic framework angular 12).	64
Figura 36 “Código HTML” (Datos Obtenidos en el Software visual basic framework angular 12).	65
Figura 37 “Implementación del nz- select” (Datos Obtenidos en el Software visual basic framework angular 12).	66
Figura 38 “Implementación del botón” (Datos Obtenidos en el Software visual basic framework angular 12).	66
Figura 39 “Plantilla del drawer” (Datos Obtenidos en el Software visual basic framework angular 12).	67
Figura 40 “modulo de entrada de datos de respuesta y de arreglo” (Datos Obtenidos en el Software visual basic framework angular 12).	68
Figura 41 “servicio de miembros” (Datos Obtenidos en el Software visual basic framework angular 12).	69
Figura 42 “librerias del servicios” (Datos Obtenidos en el Software visual basic framework angular 12).	70
Figura 43 “Construtor del servicio y funcion get all” (Datos Obtenidos en el Software visual basic framework angular 12).	70
Figura 44 “funciones de editar y agregar” (Datos Obtenidos en el Software visual basic framework angular 12).	71
Figura 45 “librerías de registro”(Datos Obtenidos en el Software visual basic framework angular 12).	72
Figura 46 “variables para ingresar en el ts” (Datos Obtenidos en el Software visual basic framework angular 12).	72
Figura 47 “objeto miembro” (Datos Obtenidos en el Software visual basic framework angular 12).	73
Figura 48 “objeto correo” (Datos Obtenidos en el Software visual basic framework angular 12).	73
Figura 49 “arreglo de membresías” (Datos Obtenidos en el Software visual basic framework angular 12).	74
Figura 50 “variables” (Datos Obtenidos en el Software visual basic framework angular 12).	74
Figura 51 “constructor” (Datos Obtenidos en el Software visual basic framework angular 12).	75
Figura 52 “inicializaciones en las variables” (Datos Obtenidos en el Software visual basic framework angular 12).	75
Figura 53 “funciones del modal” (Datos Obtenidos en el Software visual basic framework angular 12).	76
Figura 54 “función muestra” (Datos Obtenidos en el Software visual basic framework angular 12).	77
Figura 55 “función de arranque de la página web” (Datos Obtenidos en el Software visual basic framework angular 12).	78
Figura 56 “función del registro” (Datos Obtenidos en el Software visual basic framework angular 12).	79
Figura 57 “validación de la función de registro” (Datos Obtenidos en el Software visual basic framework angular 12).	80
Figura 58 “función de abrir” (Datos Obtenidos en el Software visual basic framework angular 12).	81
Figura 59 “enviar mensaje” (Datos Obtenidos en el Software visual basic framework angular 12).	81

Figura 60 “implementación de servicio de correo” (Datos Obtenidos en el Software visual basic framework angular 12).	82
Figura 61 “la plantilla de correo” (Datos Obtenidos en el Software visual basic framework angular 12).	83
Figura 62 “HTML del monitor de miembros”(Datos Obtenidos en el Software visual basic framework angular 12).	84
Figura 63 “drawer de registro” (Datos Obtenidos en el Software visual basic framework angular 12).	85
Figura 64 “botón de datos” (Datos Obtenidos en el Software visual basic framework angular 12).	85
Figura 65 “drawer del correo” (Datos Obtenidos en el Software visual basic framework angular 12).	86
Figura 66 “librerías del componente de miembros ts” (Datos Obtenidos en el Software visual basic framework angular 12).	87
Figura 67 “creacion del modulo” (Datos Obtenidos en el Software visual basic framework angular 12).	87
Figura 68 “arreglo y constructor de miembros” (Datos Obtenidos en el Software visual basic framework angular 12).	88
Figura 69 “variables y el arreglo de correo” (Datos Obtenidos en el Software visual basic framework angular 12).	89
Figura 70 “función oninit” (Datos Obtenidos en el Software visual basic framework angular 12).	90
Figura 71 “función open” (Datos Obtenidos en el Software visual basic framework angular 12).	90
Figura 72 “configuraciones de la funcion de envio de mensajes de correo en miembros” (Datos Obtenidos en el Software visual basic framework angular 12).	91
Figura 73 “función de editar” (Datos Obtenidos en el Software visual basic framework angular 12).	92
Figura 74 “HTML monitor usuario” (Datos Obtenidos en el Software visual basic framework angular 12).	92
Figura 75 “monitor de usuario” (Datos Obtenidos en el Software visual basic framework angular 12).	93
Figura 76 “modulo de usuario” (Datos Obtenidos en el Software visual basic framework angular 12).	94
Figura 77 “modulo de departamento” (Datos Obtenidos en el Software visual basic framework angular 12).	95
Figura 78 “modulo de roles” (Datos Obtenidos en el Software visual basic framework angular 12).	95
Figura 79 “servicio de departamento” (Datos Obtenidos en el Software visual basic framework angular 12).	96
Figura 80 “servicio de roles” (Datos Obtenidos en el Software visual basic framework angular 12).	97
Figura 81 “servicio de usuarios” (Datos Obtenidos en el Software visual basic framework angular 12).	98
Figura 82 “librerías para el componente ts” (Datos Obtenidos en el Software visual basic framework angular 12).	99
Figura 83 “variables de usuarios” (Datos Obtenidos en el Software visual basic framework angular 12).	100
Figura 84 “variables de envio de correos” (Datos Obtenidos en el Software visual basic framework angular 12).	100
Figura 85 “arreglos de edición” (Datos Obtenidos en el Software visual basic framework angular 12).	101
Figura 86 variables en constructor	102
Figura 87 “creación de funciones en el oninit” (Datos Obtenidos en el Software visual basic framework angular 12).	103
Figura 88 “función de agregar y editar” (Datos Obtenidos en el Software visual basic framework angular 12).	104
Figura 89 “función de actualizar” (Datos Obtenidos en el Software visual basic framework angular 12).	105
Figura 90 “funciones de actualización” (Datos Obtenidos en el Software visual basic framework angular 12).	105
Figura 91 “funciones del envío de correos” (Datos Obtenidos en el Software visual basic framework angular 12).	106
Figura 92 “HTML patrocinadores” (Datos Obtenidos en el Software visual basic framework angular 12).	107
Figura 93 “domicilio html de patrocinadores” (Datos Obtenidos en el Software visual basic framework angular 12).	108
Figura 94 “modulo de las variables de lo que tiene contactos” (Datos Obtenidos en el Software visual basic framework angular 12).	109
Figura 95 “modulo de contactos” (Datos Obtenidos en el Software visual basic framework angular 12).	110
Figura 96 “ts de patrocinadores (Datos Obtenidos en el Software visual basic framework angular 12).	111
Figura 97 “librerías interfaces de correo” (Datos Obtenidos en el Software visual basic framework angular 12).	112
Figura 98 “variable de patrocinadores (Datos Obtenidos en el Software visual basic framework angular 12).	

	113
Figura 99 “creación de arreglos del patrocinadores” (Datos Obtenidos en el Software visual basic framework angular 12).	114
Figura 100 “constructor” (Datos Obtenidos en el Software visual basic framework angular 12).	115
Figura 101 “oninit donde se obtendran los datos de la base de datos y se mostraran en la tabla” (Datos Obtenidos en el Software visual basic framework angular 12).	116
Figura 102 “funciones de agregación” (Datos Obtenidos en el Software visual basic framework angular 12).	117
Figura 103 “funciones de patrocinio” (Datos Obtenidos en el Software visual basic framework angular 12).	118
Figura 104 “funciones de contacto y domicilio” (Datos Obtenidos en el Software visual basic framework angular 12).	119
Figura 105 “funciones de edición y la de correo” (Datos Obtenidos en el Software visual basic framework angular 12).	120
Figura 106 “correo individual html” (Datos Obtenidos en el Software visual basic framework angular 12).	121
Figura 107 “librerías e interfaces” (Datos Obtenidos en el Software visual basic framework angular 12).	122
Figura 108 “variables” (Datos Obtenidos en el Software visual basic framework angular 12).	122
Figura 109 “constructor de correo individual” (Datos Obtenidos en el Software visual basic framework angular 12).	123
Figura 110 “oninit en correo individual” (Datos Obtenidos en el Software visual basic framework angular 12).	123
Figura 111 “correo masivo” (Datos Obtenidos en el Software visual basic framework angular 12).	124
Figura 112 “codigo de correo masivo ts2(Datos Obtenidos en el Software visual basic framework angular 12).	125
Figura 113 “codigo del ts submitrom” (Datos Obtenidos en el Software visual basic framework angular 12).	126
Figura 114 “captura de la función enviar” (Datos Obtenidos en el Software visual basic framework angular 12).	127
Figure 115 “es la implementación de el modo de arranque para que la aplicación con el comando de ng - serve para poder observar las interfaces” (Datos Obtenidos en el Software visual basic terminal cmd de visual basic framework angular 12).	128
Figura 116 “interfaz de login” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	128
Figura 117 “interfaz de registro de miembros” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	129
Figura 118 “Interfaz del monitor de miembros” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	129
Figura 119 “interfaz de monitor de usuario” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	130
Figura 120 “drawers para la alta de usuarios” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	130
Figura 121 “Interfaz del monitor de patrocinador” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	131
Figura 122 Interfaz de alta de usuarios(Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	131
Figura 123 “Intefaz de patrocinadores con la vista de las tablas de contactos y de datos” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	132
Figura 124 interfaz de la edicion de contacto(Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	133
Figura 125 “Interfaz de domicilio” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	133
Figura 126 “Interfaz de correo masivo” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).	134
Figura 127 “muestra de prueba del correo” (Datos Obtenidos en el Software visual basic terminal cmd de visual basic framework angular 12).	135
Figura 128 “info de pack json” (Datos Obtenidos por el autor en el software nodejs. backend.)	136

Figura 129 “typorm” (Datos Obtenidos por el autor en el software nodejs. backend.)	137
Figura 130 “rutas” (Datos Obtenidos por el autor en el software nodejs. backend.)	138
Figura 131 “ruta login” (Datos Obtenidos por el autor en el software nodejs. Backend)	139
Figura 132 “ruta correo” (Datos Obtenidos por el autor en el software nodejs. backend.)	139
Figura 133 “ruta usuario” (Datos Obtenidos por el autor en el software nodejs. backend.)	140
Figura 134 ruta roles	141
Figura 135 “ruta departamento” (Datos Obtenidos por el autor en el software nodejs. backend.)	142
Figura 136 “ruta miembros” (Datos Obtenidos por el autor en el software nodejs. backend.)	143
Figura 137 “ruta clave referido” (Datos Obtenidos por el autor en el software nodejs. backend.)	144
Figura 138 “ruta giro” (Datos Obtenidos por el autor en el software nodejs. backend.)	144
Figura 139 “ruta patrocinio” (Datos Obtenidos por el autor en el software nodejs. backend.)	145
Figura 140 “creacion de los entytis” (Datos Obtenidos por el autor en el software nodejs. backend.)	146
Figura 141 “funciones de encriptado y de verificacion de contraseña” (Datos Obtenidos por el autor en el software nodejs. backend.)	146
Figura 142 “correo controllador y la libreria de respuesta y requerimeintos” (Datos Obtenidos por el autor en el software nodejs. backend.)	146
Figura 143 “creacion del transporte” (Datos Obtenidos por el autor en el software nodejs. backend.)	147
Figura 144 “validacion del transporte” (Datos Obtenidos por el autor en el software nodejs. backend.)	148
Figura 145 “controlador de departamento sus librerias y su variable estatica” (Datos Obtenidos por el autor en el software nodejs. backend.)	148
Figura 146 “función getall” (Datos Obtenidos por el autor en el software nodejs. backend.)	149
Figura 147 “funciones del getbyid y la funcion new” (Datos Obtenidos por el autor en el software nodejs. backend.)	150
Figura 148 “funciones de roles controllers” (Datos Obtenidos por el autor en el software nodejs. backend.)	151
Figura 149 “funciones de clave referido controller” (Datos Obtenidos por el autor en el software nodejs. backend.)	152
Figura 150 “funciones de girocontroller” (Datos Obtenidos por el autor en el software nodejs. backend.)	153
Figura 151 “funciones de recuperación de datos” (Datos Obtenidos por el autor en el software nodejs. backend.)	154
Figura 152 “función getbycuenta” (Datos Obtenidos por el autor en el software nodejs. backend.)	155
Figura 153 “función new de usuario” (Datos Obtenidos por el autor en el software nodejs. backend.)	156
Figura 154 “función editar datos” (Datos Obtenidos por el autor en el software nodejs. backend.)	157
Figura 155 “funciones de eliminar de usuario y la parte de guardar los datos que se editaron” (Datos Obtenidos por el autor en el software nodejs. backend.)	158
Figura 156 “funcion para obtener datos” (Datos Obtenidos por el autor en el software nodejs. backend.)	159
Figura 157 “función de buscar id” (Datos Obtenidos por el autor en el software nodejs. backend.)	160
Figura 158 “funcion de obtener datos por email” (Datos Obtenidos por el autor en el software nodejs. backend.)	160
Figura 159 “inicio de la función miembro” (Datos Obtenidos por el autor en el software nodejs. backend.)	161
Figura 160 “variable del repositorio donde se guarda miembros” (Datos Obtenidos por el autor en el software nodejs. backend.)	161
Figura 161 “función editar” (Datos Obtenidos por el autor en el software nodejs. backend.)	162
Figura 162 “función eliminar eliminar” (Datos Obtenidos por el autor en el software nodejs. backend.)	163
Figura 163 “librería de patrocinio” (Datos Obtenidos por el autor en el software nodejs. backend.)	163
Figura 164 “función gett all” (Datos Obtenidos por el autor en el software nodejs. backend.)	164
Figura 165 “funciones de busquedas para buscar domicilio contacto y la id del patrocinio” (Datos Obtenidos por el autor en el software nodejs. backend.)	165
Figura 166 “funcion nuev” (Datos Obtenidos por el autor en el software nodejs. backend.)	166
Figura 167 “contacto nuevos” (Datos Obtenidos por el autor en el software nodejs. backend.)	166
Figura 168 creación del patrocinio de domicilio(Datos Obtenidos por el autor en el software nodejs. Backend).	167
Figura 169 “editar patrocinadores ,contacto y domicilio” (Datos Obtenidos por el autor en el software	

nodejs. backend.)	168
Figura 170. “Función de Domicilio”. (Datos Obtenidos por el autor en el software Visual Stdudio Code).	169
Figura 171 “catch de patrocinio” (Datos Obtenidos por el autor en el software nodejs. backend.)	170
Figura 172 modo de arranque del servidor Datos Obtenidos en el Software visual basic termial cmd de visual basic framework angular 12).	170
Figura 173. “Prueba insomnia con el registro de un usuario lremos a la parte de backend y se debe demostrar”.(Datos Obtenidos por el autor en el software Insomnia).	171
Figura 174. “mensaje del bakend cuando ingresa un usuario”.(Datos Obtenidos por el autor en el software nodejs. backend.	172
Figura 175 “entorno de prueba” “. (Datos Obtenidos por el autor en el software Insomnia).	173
Figura 176: ”mensaje de error en el sistema al no introducir bien la contraseña y la cuenta” “. (Datos Obtenidos por el autor en el browser mozillafierox angular localhost 4200).	174
Figura 177 “El mensaje que se ve en el backend” “. (Datos Obtenidos por el autor en el software nodejs. backend.	175
Figura 178 cuando el resultado es correcto”. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	175
Figura 179 “muestra los datos ingresados del usuario” Datos Obtenidos por el autor en el software nodejs. Backend respuesta de datos.	176
Figura 180 “registro de miembros”. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	177
Figura 181 “muestra de la tabla con datos” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	178
Figura 182 “edición” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	178
Figura 183 actualización de la tabla. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	179
Figura 184 “alta de usuario” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	180
Figura 185 “tabla llena de usuarios” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	181
Figura 186 “envío de correo” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	182
Figura 187 “alta de patrocinado” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	183
Figura 188 “tabla de patrocinadores” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	183
Figura 189 “interfaz de edición” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	184
Figura 190 “edición de domicilio”. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	185
Figura 191 “nuevo patrocinador contacto o domicilio dentro de la edición” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	186
Figura 192 “actualización del patrocinador con sus nuevos contactos y domicilio” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	187
Figura 193 “envió de correo” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	188
Figura 194 “envío individual” “. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)	189
Figura 195 “masivo bienvenida” “. (Datos Obtenidos por el autor en el browser de gmail)	190
Figura 196 “patrocinadores moyco” “. (Datos Obtenidos por el autor en el browser de gmail)	191
Figura 197 “individual” “. (Datos Obtenidos por el autor en el browser de gmail)	191
Figura 198 “registro de un miembro” “. (Datos Obtenidos por el autor en el browser de gmail)	192

Índice de Tablas

Tabla 1 “Planeación de actividades semanales de los meses de oct a feb”. (Datos Obtenidos de: Michell Mazón Rojas)	23
Tabla 2. “Actividades de Creación de Proyectos y Pruebas de Diciembre a Marzo”. (Datos Obtenidos por el Autor)	24

Agradecimientos

Gracias:

Señoras y Señores Docentes, les comparto mi agradecimiento, a través de este documento como mi único medio de expresar el éxito logrado junto a las personas en las cuales me dieron la confianza de quien soy y de mis capacidades porque gracias a ello tuve una oportunidad de aprendizaje

Por grandes razones, agradezco primero a dios, quien me dio la oportunidad de la vida, y la salud, así como a mi mamá, que a pesar de todas las trabas estuvo conmigo en los malos y buenos momentos en los cuales me apoyo en todo lo indispensable, a mi abuela que me apoyan en lo que otras personas jamás me ayudarían y por ultimo a esas personas que ya no están conmigo que dejaron un vacío pero gracias a eso tuve que luchar mucho para para alcanzar el éxito y honrar su memoria de las esas personas

Agradezco al Tecnológico de Estudios Superiores de Coacalco, que, con sus Docentes, lograron pulir mis módicas facultades y lograr el desarrollo en la Ingeniería en tecnologías de la información y comunicación.

Y agradezco a quienes no creyeron en que logre mi cometido, después su actitud fue el lastre que elevó mi fuerza de trabajo.

Resumen

En el presenta trabajo, se reportan avances del desarrollo de un sistema en ambiente Web enfocado a la administración de la información que necesita la página de FEMAEPAS, en la cual se utiliza para promocionar los servicios, productos de los patrocinadores y donde se registran las personas que quieren ser miembros de FEMAEPAS, por lo tanto el sistema a desarrollar debe contar con la gestión de los miembros, la gestión de los patrocinadores, el envío masivo e individual de correos a los miembros, la gestión de la seguridad que contempla los usuarios y los roles que van a tener acceso al sistema de administración.

La tecnología que se va a ocupar en el BackEnd (en el servidor) node.js como servidor de aplicaciones y servicios REST, como base de datos se ocupa MySQL, en la parte del FrontEnd (en los Cliente) se utiliza el framework de angular con apoyo en la parte presentación se ocupa ng-zorro-antd.

Palabras Clave:



Sistemas web



Nodejs



Consumir datos



Framework



Proceso cliente servidor

Abstract

In the present work progress is reported in the development of a system in a Web environment focused on the administration of the information that the FEMAEPAS page needs, in which it is used to promote the services, products of the sponsors and where people register. who wish to be members of FEMAEPAS, therefore the system to be developed must have the management of affiliates, the management of sponsors, the mass and individual sending of emails to affiliates, the security management that includes users and roles that will have access to the administration system.

The technology that will be used in the BackEnd (on the server) node.js as an application server and REST services, MySQL is used as a database, in the FrontEnd part (in the Clients) the angular framework is used with support in the presentation part, ng-fox-antd is taken care of.

Keywords:

 Web systems
 Nodejs
 Framework

 Consume data
 client server process

Capítulo I: Generalidades del Proyecto

1.1 Introducción

Citado de Argote Sevillano, 9-may-2019 el crecimiento de las nuevas tecnologías es un hecho y podríamos afirmar que estos avances han mejorado nuestra calidad de vida. En particular las Tecnologías de la Información y Comunicación (Tics) actualmente llegan prácticamente a todos los ámbitos, como, por ejemplo, en la educación, ingeniería, psicología, administración, química, física, por mencionar algunos, debido a que han permitido que se desarrollen herramientas, simuladores, programas, instrumentos, sistemas de información, aplicaciones, etc., para automatizar funciones, proceso y diferentes actividades que brindan un beneficio. En el marco de la investigación científica en donde se requiere de la captura y procesamiento de grandes cantidades de información, y que, además, los recursos humanos y el tiempo necesario para realizar estas actividades dependen del tamaño de la muestra y del tipo de instrumentos utilizados.

Las Tics han ayudado a simplificar dichas actividades, reduciendo el tiempo para la aplicación de los cuestionarios y procesando resultados confiables para grupos numerosos de personas, evitando así, los errores humanos que se pueden presentar al momento de transcribir los datos. Aunado a lo anterior, las Tics ofrecen otros beneficios como: facilidad y rapidez en el cálculo de la puntuación de cada escala, ramificación automática de los ítems, flexibilidad en los cambios de los elementos de un cuestionario disminución de los costos que genera la producción, calificación y aplicación reducción de respuestas omitidas eliminación de errores que se cometen en la evaluación manual aplicación de cuestionarios a personas que viven en cualquier parte del mundo y conjuntamente permiten el acceso a cuestionario grandes y diversos. Por tanto, se han desarrollado diferentes aplicaciones y sistemas que brindan diversas funcionalidades para dar respuesta a un número ilimitado de casos particulares que requieren gestionar información, reducir la incidencia de errores humanos en su captura, recibir de manera instantánea las respuestas y distribuir los resultados en el menor tiempo posible. Por ejemplo, en la siguiente tabla se describen las características de algunos sistemas y aplicaciones enfocados a la gestión de cuestionarios,

1.2. Descripción de la Empresa y Organización

La Federación Mundial Académica y Empresarial de las Áreas y Profesionales de la salud (FEMAEPAS) es una empresa que cuyo objetivo ser el vínculo entre los agremiados de las áreas de la salud con los servicios que prestan el área empresarial, para obtener beneficios a nuestros agremiados, como descuentos o precios preferenciales y servicios especiales. capacitación a los agremiados en temas empresariales (finanzas, contabilidad, administración, entre otros.) para ser empresarios exitosos

Nuestra infraestructura cuenta con tecnología de vanguardia y calificada, en la que se incluyen sistemas capaces de generar las condiciones normativas y operativas necesarias para la administración.

Misión:

Transformar la visión de los profesionales en la salud en grandes empresarios consolidados dentro del sector de ventas de salud

Visión:

Ser el vínculo entre profesionales de la salud y el entorno empresarial nacional más importante de México e internacional

Valores:



Calidad



Progresión



Conciencia cívica



Conciencia social



Conciencia ambiental

Giro y Actividad

Giro: Una federación es una agrupación formal e institucionalizada que acoge diversas entidades sociales autónomas en torno a un interés o elemento común. La palabra federación proviene del latín federation, que quiere decir ‘unión’. Por lo tanto, federación hace alusión a la acción y efecto de unir.

En la cual la agrupación se encarga de impulsar las áreas de la salud con las áreas empresariales en el cual el impacto que se busca es unir esas dos áreas y buscar beneficios con ello

Actividad:

Como Empresa: Es una empresa dedicada que cuyo objetivo ser el vínculo entre los agremiados de las áreas de la salud con los servicios que prestan el área empresarial, para obtener beneficios a nuestros agremiados, como descuentos o precios preferenciales y servicios especiales capacitación a los agremiados en temas empresariales (finanzas, contabilidad, administración, etc.) para ser empresarios exitosos

Durante el Proyecto: En la página oficial de FEMAEPAS en la cual se utiliza para promocionar los servicios, productos de los patrocinadores y donde se registran las personas que quieren ser miembros de FEMAEPAS, por lo tanto, el sistema a desarrollar debe contar con la gestión de los miembros, la gestión de los patrocinadores, el envío masivo e individual de correos a los miembros, la gestión de la seguridad que contempla los usuarios y los roles que van a tener acceso al sistema de administración.

Estructura Organizacional

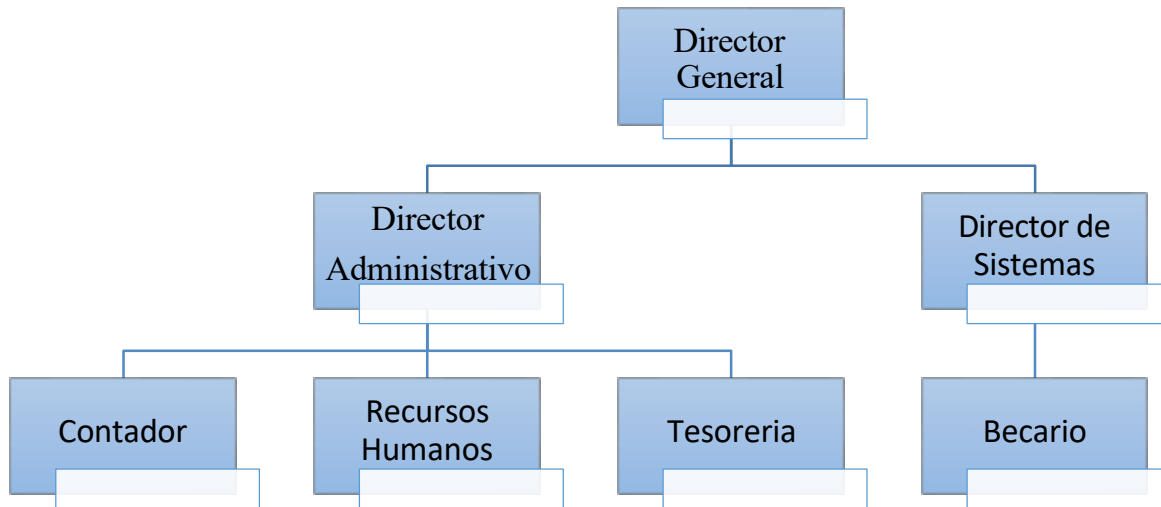


Figura 1. “Estructura Organizacional FEMAEPAS Durante la estancia”.
(Datos Obtenidos de: <https://www.femaepas.com>)

Ubicación

Gustavo Baz Prada #73, Col. Bosques de Echegaray
Naucalpan de Juárez, Estado de México, CP.53310

Horarios de Servicio

Lunes a viernes: 8:30 horas - 18:00 horas

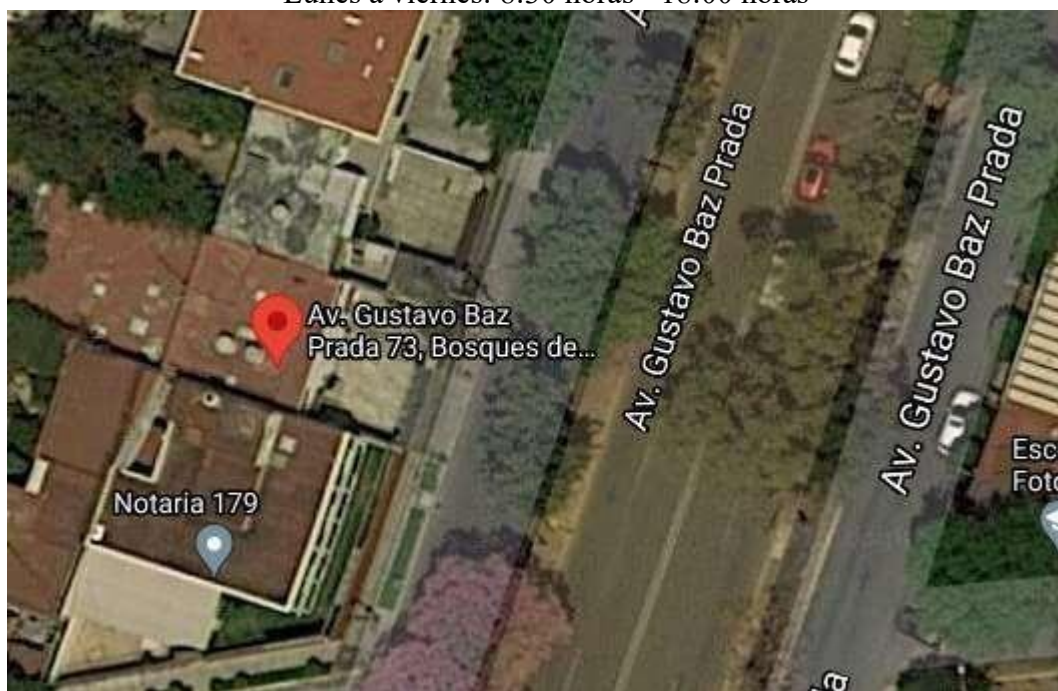


Figura 2. “Referencia de Ubicación”. (Datos Obtenidos de: Federación Mundial Académica y Empresarial de las Áreas y Profesionales de la Salud - Google Maps)

Objetivos de la Empresa:

- ✚ En busca de procura mantener un balance entre las necesidades de las partes interesadas del negocio: clientes, colaboradores, accionistas, proveedores y la sociedad. Este equilibrio ha surgido y se ha fortalecido gracias a nuestro propósito.
- ✚ Fomentar en nuestra gente la pasión por contribuir con un mejor entorno, una mejor sociedad y un México mejor.

1.2.1 Descripción del Puesto o Área de Trabajo

Puesto desarrollador de software: Desarrollar los pilares de los sistemas operativos que son creados por los Programadores y de probar el código de nuevos programas para garantizar su eficiencia. Asimismo, realizan pruebas de calidad en nuevos proyectos antes de su lanzamiento. Están involucrados en todo el proceso relacionado a la creación y diseño de nuevos sistemas, partiendo de la planificación inicial, estableciendo parámetros, diseño, escritura, código, encriptar y probar. Cada miembro del equipo de Desarrolladores de Sistemas asume una de las fases que será supervisada por el Gerente de Proyecto.

Según la descripción de desarrollo de software, (s.f.), la Función principal de los desarrolladores de software es:

- Crear y desarrollar nuevos programas o sistemas:
 - Investigar las necesidades de los usuarios.
 - Diseñar y elaborar nuevos programas.
 - Probar los nuevos programas.
- Evaluar sistemas nuevos y existentes:
 - Diseñar planes de prueba para los programas nuevos desarrollados.
- Realizar pruebas para la medición de calidad en los sistemas desarrollados.
 - Detectar errores en los programas.
 - Corregir los errores presentes en los programas.
 - Mejorar programas existentes:
- Analizar los requerimientos y sugerencias de los usuarios.
 - Crear soluciones para fallas existentes.
 - Implementar las soluciones.
- Realizar el mantenimiento correspondiente en los sistemas existentes, realizando el monitoreo y corrección de los defectos detectados, así como elaborar el código en lenguajes especializados (HTML, PHP, XML) para nuevos programas:
 - Ejecutar el código para medir su eficiencia.
 - Reescribir el código para detectar errores.
 - Realizar pruebas hasta garantizar que el programa esté libre de errores.
- Elaborar manuales operativos y especificaciones técnicas de los sistemas.
- Trabajar en conjunto con el resto del equipo, es decir, con Gerentes de Proyecto, Diseñadores Gráficos, otros Desarrolladores, Administradores de Bases de Datos y Reunirse con los clientes y Gerentes de Proyecto para diseñar y desarrollar nuevos programas.
 - Establecer parámetros y diseñar la arquitectura de nuevos programas.
 - Diseñar, escribir, leer, probar y corregir el código de nuevos programas.
- Realizar pruebas de medición de calidad y detectar errores en el desarrollo del programa.
 - Preparar la documentación necesaria para programas nuevos o actualizados.

Asesor Externo:

El encargado del área es Sergio Quintana Gutiérrez, **Gerente Administrativo y Director de Sistemas.**

Sus funciones dentro de la empresa son las siguientes:

Gerente Administrativo: según la descripción de gerente administrativo, (s.f.), sus funciones son las siguientes :

Gestiona las Operaciones Administrativas Diarias. Forma parte del equipo de altos ejecutivos que depende directamente del director ejecutivo y su trabajo será un factor crítico en el rendimiento general. Sabe cómo mantener las cosas en orden y añadir una visión estratégica a las actividades diarias, queremos conocerte. Si, además, eres un líder natural y muy eficiente, eres nuestro candidato ideal.

Director de Sistemas de Información: En la parte de descripción de director de sistemas, 2021 se indica que Es considerado parte del equipo de administración de la compañía. Con ese propósito, se reúne con el Directorio, debe considerar los pedidos de otros ejecutivos e informarles acerca de sus actividades.

Habilidades: Debe poseer una experiencia técnica sólida en TI y una visión estratégica para la innovación y el cambio, junto con habilidades de administración.

Es más, el DSI debe ser metódico y riguroso en su trabajo. Debe poseer habilidades interpersonales para comunicarse con su equipo y debe tener la capacidad de negociar. Por último, debe tener buen dominio de inglés, independientemente del lugar donde viva, para contactarse con socios comerciales de otros países.

1.3 Justificación

En el presente trabajo que se requiere presentar en del Tecnológico de estudios superiores de Coacalco para culminar el ciclo formativo de grado superior de Tecnologías de la información y comunicación con especialización en tecnologías móviles con el motivo de cubrir 500 horas de residencias profesionales me estoy integrando al siguiente proyecto:

desarrollo de la administración de información femaepas.com.

El siguiente proyecto se realiza con la intención de administrar la información registrada de los agremiados en la página actual de femaepas.com, así como la información publicada de los patrocinadores en la página de femaepas.com.

Con el objetivo de optimización de envíos del correo en los cuales se ingresara el miembro al cual se le enviara cierto mensaje para no perder tiempo se decide hacer más eficaz este tipo de envío de correo al igual que el proceso de validación de vigencia se realizara con el afán de administrar los agremiados que cuentan con su membresía vigente y gozar de los beneficios que tienen su membresía así como darle de alta a los usuarios que podrán administrar en la página web y así con ello hacer uso de la gestión de los roles priorizan ciertas funciones de edición y envío de correos a los agremiados otro punto que se va a tratar es la auditoria de la información la en la cual se recolectara basándonos en las necesidades del usuarios después se ingresara con un entorno de pruebas el cual tiene como necesidad a cubrir un ambiente similar al que se usara para la producción del sistema así como la entrada de servidores virtuales dentro de las áreas de backend donde estará el servicio proporcionado para el respaldo de la base de datos e interacción de los usuarios hacia la página web

1.4 Problemas a Resolver

En la federación mundial académica y empresarial de las áreas de la salud mencionaron que no se cuenta con un sistema de administración de la página de femae-pas.com dentro de la página web se dan de alta los registros y los beneficios que se encuentran en los tipos de membresía al igual que la integración de los convenios en el sistema que se llevara a cabo es llevar un control de los Agremiados dados de alta en la página ya que en los agremiados tenemos que llevar la información de los pagos y de los membrecías. Al igual que no se cuenta con envíos de correos electrónicos donde se hace llegar beneficios y ofertas de la federación o mandarle información valiosa como su contraseña o un status de pagos al igual que no se cuenta con una sección de alta de patrocinadores en la cual es importante guardar los datos que son las vigencias. Relacionadas al status de pago otra de las cosas que necesitamos integrar al proyecto es un control de roles al usuario en el cual permita navegar a ciertas páginas del sistema y con ellas darles permisos de modificación de los datos

1.5 Objetivos

Objetivo General:

Crear un sistema administrativo que permita la gestión de la información que se genera en la página de femaepas.com.

Objetivos Específicos:

- Gestionar el desarrollo de los componentes de usuario, agremiados, administración de patrocinadores
 - Analizar las altas dadas de la página como al igual las bajas que se desarrollan.
- Verificar vigencias, registro de pagos y los beneficios de cada agremiado que se dieron de alta
 - Corregir datos los datos generales o actualizar
 - Consolidar el envío de correos masivo o individual
- Gestionar el entorno de pruebas para la herramienta de insomnia para validación servicios REST
 - Analizar el permiso de los roles dentro de la página administrativa

1.6 Cronograma de Actividades

Actividades	Oct Sem 1	Oct. Sem 2	Oct Sem 3	Oct Sem 4	Nov Sem 1	Nov Sem 2	Nov Sem 3	Nov Sem 4	Dic Sem 1	Dic Sem 2	Dic Sem 3	Ene Sem 1	Ene Sem 2	Ene Sem 3	Ene- Sem 4	Feb Sem 1
Levantamiento de requerimientos																
Análisis de los requerimientos																
Diseño del prototipo																
Presentación de requerimientos																
Desarrollo del prototipo																
Presentación a asesores																
Diseño de planificación																

Tabla 1 “Planeación de actividades semanales de los meses de oct a feb”. (Datos Obtenidos de: Michell Mazón Rojas).

Actividades	Dic Sem 1	Dic Sem 2	Dic Sem 3	Ene Sem 1	Ene Sem 2	Ene Sem 3	Ene- Sem 4	Feb Sem 1	Feb Sem 2	Feb Sem 3	Feb Sem 4	Mar Sem 1	Mar Sem 2	Mar Sem 3	Mar- Sem 4	Abril Sem 1
Creación de proyecto																
Creación de la base de datos en mysql																
Creación del entorno web																
Revisión de asesores																
Creación del entorno ba- ckend																
Creación de entorno de pruebas																
Pruebas de corrección de datos y actualización																
Prueba de envíos de co- reos masivo e indivi- dual																
Pruebas de alta de usua- rios y patrocinadores y miembros																
Prueba de gestión de ro- les																
Prueba de final de efi- cacia																
documentación																
Conclusión.																
Entrega final.																

Tabla 2. “Actividades de Creación de Proyectos y Pruebas de Diciembre a Marzo”. (Datos Obte-
nidos por el Autor).

Capítulo II Marco Teórico

2.1. ¿Qué es un Software Administrativo?

Según lo consultado en el libro relacionado al Software erp s.f, la definicion que otorga es un programa o sistema informático utilizado a nivel empresarial. En general, se desarrollan para **registrar, organizar, controlar y detallar toda la información** que generan los departamentos de una organización. Es decir, se trata de una solución integral

utilizada con frecuencia para:

- Acelerar la gestión y el almacenamiento de datos
- Agilizar las tareas administrativas y de gestión
 - Optimizar procesos
- Reducir los costos provenientes del flujo de trabajo.

Este tipo de herramientas digitales cuentan con un conjunto de aplicaciones para gestionar los procesos organizacionales.

Estos procesos están relacionados con las áreas financieras, contables, inventarios, recursos humanos, procesos de fabricación, atención al cliente, cadena de suministros, entre otras.

Para ser más específicos, este tipo de *software* permite optimizar y agilizar las tareas relacionadas con:

- Facturación
- Cuentas por cobrar
 - Pagos
- Control de compras y gastos
- Registro, control y pago de personal
 - Registros contables y más

2.2. ¿Qué es Angular?

Angular es una plataforma de desarrollo, construida sobre Mecanografiado. Como plataforma, Angular incluye:

Dentro de la definición de Angular (s.f.), un marco basado en componentes para crear aplicaciones web escalables

Una colección de bibliotecas bien integradas que cubren una amplia variedad de funciones, incluido el enrutamiento, la gestión de formularios, la comunicación cliente-servidor y más

Un conjunto de herramientas de desarrollo para ayudarlo a desarrollar, compilar, probar y actualizar su código

2.2. Características:

Según el trabajo de investigación de (azutare) y su desarrollo web agil en angular tenemos este tipo de características:

Plataforma Cruzada:

Aplicaciones web progresivas

Utilice las capacidades de la plataforma web moderna para ofrecer experiencias similares a las de una aplicación. Instalación de alto rendimiento, fuera de línea y paso cero.

Nativo

Cree aplicaciones móviles nativas con estrategias de Córdova, Ionice o NativeScript.

Escritorio

Cree aplicaciones instaladas en el escritorio en Mac, Windows y Linux utilizan los mismos métodos de Angular que aprendió para la web, además de la capacidad de acceder

a las *API nativas del sistema operativo*.

Velocidad y Rendimiento:

Código de GENERACION

Angular convierte sus plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy, brindándole todos los beneficios del código escrito a mano con la productividad de un marco.

Universal

Sirva la primera vista de su aplicación en Node.js®, .NET, PHP y otros servidores para una representación casi instantánea solo en HTML y CSS. También allana el camino para los sitios que optimizan para SEO.

División de código

Las aplicaciones angulares se cargan rápidamente con el nuevo enrutador de componentes, que ofrece división de código automática para que los usuarios solo carguen el código necesario para representar la vista que solicitan.

Productividad:

Plantillas

Cree rápidamente vistas de interfaz de usuario con una sintaxis de plantilla sencilla y potente.

CLI angular

Herramientas de línea de comandos: comience a construir rápidamente, agregue componentes y pruebas, luego implemente instantáneamente.

IDE

Obtenga finalización de código inteligente, errores instantáneos y otros comentarios

en editores e IDE populares.

Historia de Desarrollo Completa:

Pruebas

Con Karma para pruebas unitarias, puedes saber si has roto cosas cada vez que guardas.

Animación

Cree coreografías complejas y de alto rendimiento y líneas de tiempo de animación con muy poco código a través de la API intuitiva de Angular.

Accesibilidad

Cree aplicaciones accesibles con componentes habilitados para ARIA, guías para desarrolladores e infraestructura de prueba integrada cada año.

2.3. Ng zorro

La definición de (ng-zorro, s.f.) se basa en que es una biblioteca de componentes de interfaz de usuario angular de clase empresarial basada en Ant Design, todos los componentes son de código abierto y de uso gratuito bajo la licencia MIT.

Características:

- Un lenguaje de diseño de interfaz de usuario de clase empresarial para aplicaciones angulares.
 - Más de 60 componentes angulares de alta calidad listos para usar.
 - Escrito en TypeScript con tipos definidos completos.
 - Admite el modo OnPush, alto rendimiento.
 - Potente personalización del tema en cada detalle.
 - Soporte de internacionalización para decenas de idiomas

2.4. Que es Node.js

Según los datos obtenidos por (node.js, 2022) Node.js, es un entorno en tiempo de

ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript.

Node.js es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tienes que preocuparte con el bloqueo de procesos, pues no hay bloqueos.

2.4.1. ¿Cómo funciona Node.js?

dentro de la investigación (puciarelli, 2018) instalación Arquitectura node npm se indica que El diseño de Node.js

está inspirado en sistemas como el Event Machine de Ruby o el Twisted de Python. Sin embargo, Node.js presenta un bucle de eventos como una construcción en tiempo de ejecución en lugar de una biblioteca. Este bucle de eventos es invisible para el usuario.

Otra característica especial de Node.js es que está diseñado para simplificar la comunicación. No tiene subprocesos, pero te permite aprovechar múltiples núcleos en su entorno y compartir sockets entre procesos

2.4.2. Características principales de Node.js

Resumiendo, Node.js se ha hecho popular en los últimos años gracias a las siguientes según lo consultado por (Sosa Quispe, 2019) características son las siguientes :

- Velocidad. Node.js está construido sobre el motor de JavaScript V8 de Google Chrome, por eso su biblioteca es muy rápida en la ejecución de código.
- Sin búfer. Las aplicaciones de Node.js generan los datos en trozos (chunks), nunca los almacenan en búfer.
- Asíncrono y controlado por eventos. Como hemos dicho anteriormente, las APIs de la biblioteca de Node.js son asíncronas, sin bloqueo. Un servidor basado en Node.js no espera que una API devuelva datos. El servidor pasa a la siguiente API después de llamarla, y un mecanismo de notificación de eventos ayuda al servidor a obtener una respuesta de la llamada a la API anterior.
- Un subproceso escalable. Node.js utiliza un modelo de un solo subproceso con bucle

de eventos. Gracias al mecanismo de eventos, el servidor responde sin bloqueos, como hemos dicho. Esto hace que el servidor sea altamente escalable comparado con los servidores tradicionales como el Servidor HTTP de Apache.

2.4.3. ¿Para qué sirve Node.js?

(puciarelli, 2018)según su arquitectura Puedes utilizar Node.js para diferentes tipos de aplicaciones. Los siguientes son algunos de los ejemplos:

- Aplicaciones de transmisión de datos (streaming)
- Aplicaciones intensivas de datos en tiempo real
 - Aplicaciones vinculadas a E/S
 - Aplicaciones basadas en JSON: API
 - Aplicaciones de página única

2.5. Que es el Desarrollo Backend

El Backend según (Argote Sevillano, 9-may-2019) según las características de la programación reactiva es la parte trasera de cualquier página web.

Se trata de todo el conjunto del desarrollo que se encarga de que una página funcione y de que lo haga como lo hace, pero que al mismo tiempo es totalmente invisible para el usuario, que solo ve lo visual y gráfico.

Su contrapartida es el Front end, que es la mitad con la que interactúa cada visitante en una web. Ambas, juntas dan forma a todo el conjunto que construye una página web en la actualidad y lo que permite que, además de la usabilidad, también pueda hablarse de rendimiento y de comunicación en su interior.

Podemos encontrar muchos tipos de Backends diferentes en la actualidad. Por ejemplo, cualquier página hecha con WordPress, que además lo utilice como CMS, tiene en él a gran parte de su backend, ya que es lo que se encarga del lado administrativo, de permitir que se elaboren contenidos y, al mismo tiempo, que se gestionen usuarios y se lleven a cabo otras tareas de carácter interno.

Aquí encontramos lenguajes de desarrollo bastante diferentes a los anteriores, como pueden ser PHP, Java o .Net. En el Back end se llevan a cabo tareas como la lógica

de la web, las conexiones con bases de datos en el servidor, el uso de librerías o incluso las operaciones llevadas a cabo por las distintas funciones disponibles.

2.5.1. Características del Backend

- Conexión con base de datos.
- Simplificar el proceso de desarrollo.
- Uso de librerías del servidor web.
- Combinar información, transformarla y devolvérsela al usuario.

2.5.2. Para que Sirve

El Backend sirve para que tu página web funcione a la perfección a través de elementos que van desde la generación de contenidos hasta el tiempo de carga.

Este último, dicho sea de paso, es un factor muy importante para posicionar tu sitio en los motores de búsqueda, ya que, a mayor tiempo de carga, más usuarios abandonarán tu página web. Todas las acciones aplicadas en el Backend están relacionadas enteramente a la experiencia del usuario, optimizan los procesos para que la navegación en tu sitio sea lo suficientemente óptima como para captar cada vez a más visitantes.

Sería algo más o menos así: tu sitio web necesita de diseñadores que aseguren una gran interfaz (Frontend), pero también requiere de personas que desarrollen la parte analítica (Backend). Ahora que ya sabes para qué sirve el Backend, ya podemos explicarte mejor cuáles son las diferencias exactas entre ambos conceptos.

2.6. Que es Front end

El Front end es la parte de una web que conecta e interactúa con los usuarios que la visitan. Es la parte visible, la que muestra el diseño, los contenidos y la que permite a los visitantes navegar por las diferentes páginas mientras lo deseen.

Es una de las dos mitades en las que se divide la estructura de cualquier página web.

Junto a esta se encuentra el Back end, que es el polo completamente opuesto, la capa que accede a datos y software en general para su comunicación. Ambas se reúnen en

cualquier site que visite una persona y son las que, trabajan, hacen que funcione en todo momento tal y como lo hace.

Esta parte es la que engloba y muestra todo el trabajo de diseño web y, por lo general, reúne en su interior hasta 3 lenguajes de programación diferentes: HTML, CSS y JavaScript. Cada uno orientado a determinados fines en concreto, se suman para conseguir el resultado final que aparece por la pantalla de cada usuario que entra en una web, sea cual sea.

Se podría decir que el Front end es como la máscara de una página, pero en realidad va más allá de eso. Es lo que debe mostrar la identidad corporativa de una compañía dentro de Internet, la parte que puede atraer o alejar al consumidor cuando este accede a la página.

De ahí que sea tan importante y requiera de tanto esfuerzo

6.1. Para qué Sirve el Front End

(Haro, 2019) de acuerdo con la información del front- end Sirve para que el portal funcione como lo hace, que permita desde publicar contenidos hasta habilitar un botón para que el usuario se registre.

En función de cómo esté construido, puede hacer que una web sea más rápida o no, que sea más completa o menos, que el usuario quede más o menos satisfecho. El Back end es una parte fundamental de cualquier web, aunque debe estar siempre junto con el Front end.

Debido a esto, es algo que influye incluso a ojos de Google y el posicionamiento web. Tiempos de carga elevados no son buenos para el buscador, y mucho menos la presencia de código inútil o que realiza operaciones fallidas. Cuidar esta parte trasera de una web es crucial para el crecimiento de una empresa en internet.

2.7. Que Es Una Base de Datos

ORACLE, s.f. define que una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático.

Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.

Los datos de los tipos más comunes de bases de datos en funcionamiento actualmente se suelen utilizar como estructuras de filas y columnas en una serie de tablas para aumentar la eficacia del procesamiento y la consulta de datos. Así, se puede acceder, gestionar, modificar, actualizar, controlar y organizar fácilmente los datos. La mayoría de las bases de datos utilizan un lenguaje de consulta estructurada (SQL) para escribir y consultar datos.

Las bases de datos según Mendoza (2018), se hicieron predominantes en la década de 1980. Los elementos de una base de datos relacional se organizan como un conjunto de tablas con columnas y filas. La tecnología de bases de datos relacionales proporciona la forma más eficiente y flexible de acceder a información estructurada.

Bases de datos orientadas a objetos. La información de una base de datos orientada a objetos se representa en forma de objetos, como en la programación orientada a objetos.

Bases de datos distribuidas. Una base de datos distribuida consta de dos o más archivos que se encuentran en sitios diferentes. La base de datos puede almacenarse en varios ordenadores, ubicarse en la misma ubicación física o repartirse en diferentes redes.

Almacenes de datos. Un repositorio central de datos, una data warehouse es un tipo de base de datos diseñado específicamente para consultas y análisis rápidos.

Bases de datos NoSQL. Una base de datos NoSQL, o base de datos no relacional, permite almacenar y manipular datos no estructurados y semiestructurados (a diferencia de una base de datos relacional, que define cómo se deben componer todos los datos insertados en la base de datos). Las bases de datos NoSQL se hicieron populares a medida que las aplicaciones web se volvían más comunes y complejas.

Bases de datos orientadas a grafos. Una base de datos orientada a grafos almacena

datos relacionados con entidades y las relaciones entre entidades.

Bases de datos OLTP. Una base de datos OLTP es una base de datos rápida y analítica diseñada para que muchos usuarios realicen un gran número de transacciones.

Estos son solo algunos de las varias docenas de tipos de bases de datos que se utilizan hoy en día. Otras bases de datos menos comunes se adaptan a funciones científicas, financieras o de otro tipo muy específicas. Además de los diferentes tipos de bases de datos, los cambios en los enfoques de desarrollo tecnológico y los avances considerables, como la nube y la automatización, están impulsan a las bases de datos en direcciones completamente nuevas. Algunas de las bases de datos más recientes incluyen

Bases de datos de código abierto. Un sistema de base de datos de código abierto es aquel cuyo código fuente es de código abierto; tales bases de datos pueden ser bases de datos SQL o NoSQL.

Bases de datos en la nube. Una base de datos en la nube es una recopilación de datos, estructurados o no estructurados, que reside en una plataforma de cloud computing privada, pública o híbrida. Existen dos tipos de modelos de bases de datos en la nube: tradicional y base de datos como servicio (DBaaS). Con DBaaS, un proveedor de servicios realiza las tareas administrativas y el mantenimiento.

Base de datos multimodelo. Las bases de datos multimodelo combinan distintos tipos de modelos de bases de datos en un único servidor integrado. Esto significa que pueden incorporar diferentes tipos de datos.

Bases de datos de documentos/JSON. Diseñadas para almacenar, recuperar y gestionar información orientada a los documentos, las bases de datos de documentos son una forma moderna de almacenar los datos en formato JSON en lugar de en filas y columnas.

Bases de datos de autogestión. El tipo de base de datos más nuevo e innovador, las bases de datos de autogestión (también conocidas como bases de datos autónomas) están basadas en la nube y utilizan el machine learning para automatizar el ajuste de la base de datos, la seguridad, las copias de seguridad, las actualizaciones y otras tareas de gestión

rutinarias que tradicionalmente realizan los administradores de bases de datos

Que es un Servidor de Correos

Un servidor de correo electrónico o servidor de email para Tejedor (s.f), es aquel que gestiona el envío, la recepción y el almacenamiento de los correos electrónicos.

Existen grandes servicios gratuitos de email como Outlook.com, que no debemos confundir con el cliente de escritorio Outlook, o Gmail que proporcionan una interfaz web para que los usuarios puedan redactar y leer emails. Estas aplicaciones se comunican directamente con los servidores de correo asociados para ofrecer un punto de acceso centralizado con un excelente rendimiento. Obviamente, estos servicios, salvo que contratemos servicios adicionales a estas empresas, no permiten que tengamos cuentas diferentes a las que terminan en @outlook.com o @gmail.com.

Cuando disponemos de un servidor de email o de una cuenta de hosting, podemos configurar estos servidores para que funcionen con nuestro nombre de dominio o el de nuestra empresa. Algunos ejemplos de estos servidores son Exim, Dovecot, Microsoft Exchange Server y Sendmail, aunque la lista es muy extensa. Es muy habitual que junto con estos servidores también se incluya algún servicio de webmail para que los usuarios puedan enviar, recibir y gestionar correos electrónicos. De este modo, no es necesario utilizar aplicaciones externas, teniendo un punto de acceso perfectamente centralizado y configurado para funcionar con el servidor. Algunos de estos lectores son, por ejemplo, RounCube, Horde, Squirrelmail o Zimbra.

Sin embargo, muchas personas optan por utilizar aplicaciones de gestión de correo electrónico de escritorio como Microsoft Outlook o Mozilla Thunderbird, algo especialmente útil cuando se trabaja con muchas cuentas de email a la vez, aunque esto es algo que también se puede configurar en algunos webmails de los más famosos, como las cuentas gratuitas de Gmail. A estas aplicaciones se les suele denominar como clientes de email.

Componentes de un Servidor de Correos

Teniendo todo esto en cuenta, es fácil darnos cuenta de que existen varios componentes claves cuando hablamos de un servidor de correo electrónico.

- **Servidor SMTP:** es el encargado de realizar el envío y transmisión de nuestros emails desde nuestro servidor de correo hacia el destino.
- **Servidor POP:** es quien recibe los mensajes en un equipo local mediante el protocolo POP, que almacena en el equipo los mensajes sin dejar copia en el servidor (por defecto).
- **Servidor IMAP:** actúa como anfitrión de un servidor de correo, el cual obtiene una copia del correo que hay actualmente en el servidor de correo.
- **Cliente de Correo local:** Mozilla Thunderbird, Microsoft Outlook, Opera Mail, Evolution y otros son el ejemplo de clientes de correo que corren en nuestros equipos locales, y que pueden recibir el correo tanto por POP como IMAP.
- **Cliente de Correo web:** estos son los llamados Webmail, es decir, software de cliente de correo que corre remotamente en el servidor, como ya mencionamos en ejemplos con Roundcube, Squirrelmail, y los clásicos Hotmail, Gmail, Yahoo mail, etc. Estos clientes utilizan el protocolo IMAP para mostrar los mensajes.

Protocolos de Correo

El correo electrónico no podría enviarse o recibirse de forma correcta sin ser por la gran ayuda de los llamados protocolos de correo.

Veamos ahora cuales son los diferentes tipos de protocolos de correo, sus funciones y características.

SMTP es quizás el protocolo de correo más conocido. Sus siglas significan Simple Mail Transfer Protocol, en español significa Protocolo Simple de Transporte de Correo. Es un protocolo usado para enviar correo desde un servidor a otro destino.

El envío de un correo a través de SMTP consta de tres pasos:

- Cliente de correo redacta el mensaje y lo envía.
- El mensaje llega al servidor SMTP, el cual acepta y envía el mensaje.
- El mensaje es finalmente entregado y recepcionado por un servidor de correo entrante, como POP3 O IMAP.

El SMTP es un protocolo que se basa en el modelo cliente servidor, y su función principal entonces es la de permitir que un cliente pueda interactuar con un servidor, abriendo las posibilidades de que un usuario pueda enviar correo a múltiples destinatarios.

SMTP se compone de tres comandos básicos que definen su funcionamiento:

- **MAIL:** es el usado para el establecimiento de la dirección del remitente que envía los mensajes.

- RCPT: es el encargado de establecer el destinatario del mensaje.
- DATA: cumple la función de enviar el contenido escrito en el cuerpo del mensaje.

MAP significa Internet Message Access Protocol, en español significaría Protocolo de Acceso a mensajes de Internet, y se encarga de poder recibir copia de los mensajes almacenados en el servidor, pero desde un cliente de correo.

Al momento de usar IMAP para recibir nuestro correo, los mismos seguirán estando siempre alojados en el servidor de correo en Internet. Es decir, IMAP entonces viene más que a ser un servidor, en realidad, a ser un lector de correo, ya que muestra copias del correo almacenado en el disco duro del server.

Aparte de esto, también permite manipular el contenido de una casilla de correo tal como si se hiciese directo desde el server, es decir:

- Borrar emails
- Responder email
- Crear, borrar y renombrar directorios

El uso de IMAP se ha popularizado con la llegada de los Smartphones en la década del 2010, donde se puede mantener un mismo acceso a la cuenta de correo desde diferentes dispositivos como móvil inteligente, tablet, smart TV, smartwatch y por supuesto las computadoras de escritorio/notebooks.

Este protocolo como ya dijimos antes no descarga los mensajes al cliente de correo, sino que se descarga la información de las cabeceras únicamente, y luego cuando se abre el mensaje ahí sí recién descarga el componente del cuerpo o body del email.

POP

POP, o POP3, es otro famoso protocolo de correo. Sus siglas vienen del inglés Post Office Protocol, que en español significa Protocolo de Oficina de Correo. Es un protocolo que se utiliza para recibir mensajes de correo en un cliente de email.

El actual protocolo es la versión 3 de POP, ya que las versiones 1 y 2 están en el olvido hoy en día.

El funcionamiento para descargar un correo con POP3 es muy sencillo:

- Desde un cliente de correo electrónico local como Thunderbird o Outlook, configuramos una casilla de correo.
 - Le indicamos que debe descargar el correo mediante el protocolo POP.
 - Ingresamos los datos del servidor POP3 que tengamos.
- Al verificar si hay nuevo correo en el servidor, el cliente se conecta al servidor POP y descarga los emails a nuestra PC o dispositivo móvil.
 - Al descargarlos al dispositivo, los emails se borran del servidor de correos remoto.
- Debido a que borra los emails al descargarlos y únicamente permanecen en un lugar específico, esto hace que no se suela utilizar con frecuencia, y que en su lugar sea IMAP el que reine en dispositivos móviles, laptops y equipos de escritorio pues permite una sincronizada entre dispositivos que no está presente en POP.

2.9 Que es la tecnología rest

(desarrolloweb.com, s.f.) en la información correspondiente dentro de la página web la definición de REST deriva de "REpresentational State Transfer", que traducido vendría a ser “transferencia de representación de estado”, lo que tampoco aclara mucho, pero contiene la clave de lo que significa.

Porque la clave de REST es que un servicio REST no tiene estado (es stateless), lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos. Esto es, que no se puede llamar a un servicio REST y pasarle unos datos (p. ej. un usuario y una contraseña) y esperar que “nos recuerde” en la siguiente petición.

De ahí el nombre: el estado lo mantiene el cliente y por lo tanto es el cliente quien debe pasar el estado en cada llamada. Si quiero que un servicio REST me recuerde, debo pasarle quien soy en cada llamada. Eso puede ser un usuario y una contraseña, un token o cualquier otro tipo de credenciales, pero debo pasarlas en cada llamada. Y lo mismo aplica para el resto de información.

El no tener estado es una desventaja clara: tener que pasar el estado en cada llamada es, como mínimo, tedioso, pero la contrapartida es clara: escalabilidad. Para mantener un estado se requiere algún sitio (generalmente memoria) donde guardar todos los estados de todos los clientes. A más clientes, más memoria, hasta que al final podemos llegar a no poder admitir más clientes, no por falta de CPU, sino de memoria. Es algo parecido a lo

que ocurre con la web tradicional (que también es stateless). Sea cual sea la tecnología con la que desarrolles para web, seguro que conoces que puedes crear lo que se llama una “sesión”. Los datos de la sesión se mantienen entre peticiones y son por cada usuario. El clásico ejemplo de sesión puede ser un carrito de la compra, entre las distintas peticiones web, la información del carrito de compra se mantiene (¡ojo! hay otras alternativas para implementar carritos de compra).

Por norma general, la sesión se almacena en la memoria RAM del servidor, por lo que es fácil quedarnos sin memoria si introducimos demasiados datos en sesión (o tenemos muchos usuarios simultáneos). Por supuesto, podemos utilizar varios servidores, pero como bien saben los desarrolladores web, mover la sesión contenida en memoria entre servidores es generalmente imposible o altamente ineficiente, lo que penaliza el balanceo de carga, lo que a su vez redundará en menor escalabilidad y menor tolerancia a fallos. Hay soluciones intermedias, tales como guardar la sesión en base de datos, pero su impacto en el rendimiento suele ser muy grande. De hecho, el consejo principal para desarrollar aplicaciones web altamente escalables es: no usar la sesión.

Así pues, tenemos que el ser stateless es la característica principal de REST, aunque por supuesto no la única. Así, cualquier servicio REST (si quiere ser merecedor de ese nombre) debería no tener estado, pero no cualquier servicio sin estado es REST. Hay otros factores, pero vamos a destacar el que los ingleses llaman “uniform interface” y es lo que diferencia un servicio web clásico (orientado a RPC) de un servicio REST.

SOAP es el acrónimo de “Simple Object Access Protocol” y es el protocolo que se oculta tras la tecnología que comúnmente denominamos “Web Services” o servicios web. SOAP es un protocolo extraordinariamente complejo pensado para dar soluciones a casi cualquier necesidad en lo que a comunicaciones se refiere, incluyendo aspectos avanzados de seguridad, transaccionalidad, mensajería asegurada y demás. Cuando salió SOAP se vivió una época dorada de los servicios web. Aunque las primeras implementaciones eran lo que se llamaban WS1.0 y no soportaban casi ningún escenario avanzado, todo el mundo pagaba el precio de usar SOAP, ya que parecía claro que era el estándar que dominaría el futuro. Con el tiempo salieron las especificaciones WS-* que daban soluciones avanzadas, pero a la vez que crecían las capacidades de SOAP, crecía su complejidad. Al final, los

servicios web SOAP terminan siendo un monstruo con muchas capacidades pero que en la mayoría de los casos no necesitamos.

Por su parte REST es simple. REST no quiere dar soluciones para todo y por lo tanto no pagamos con una demasiada complejidad una potencia que quizá no vamos a necesitar.

2.9.2 Uniform interface

Una diferencia fundamental entre un servicio web clásico (SOAP) y un servicio REST es que el primero está orientado a RPC, es decir, a invocar métodos sobre un servicio remoto, mientras que el segundo está orientado a recursos. Es decir, operamos sobre recursos, no sobre servicios.

En una API REST la idea de “servicio” como tal desaparece. Lo que tenemos son recursos, accesibles por identificadores (URIs). Sobre esos recursos podemos realizar acciones, generalmente diferenciadas a través de verbos HTTP distintos.

Así, en un servicio web clásico (SOAP) tendríamos un servicio llamado BeerService que tendría un método llamado GetAll que me devolvería todas las cervezas. La idea, independientemente de la tecnología usada para consumir el servicio web, es que se llama a un método (GetAll) de un servicio remoto (BeerService). Del mismo modo para obtener una cerveza en concreto llamaríamos al método GetById () pasándole el id de la cerveza. De ahí que se diga que están orientados a RPC (Remote Procedure Call – Llamada a método remoto).

Por su parte en un servicio REST la propia idea de servicio se desvanece. En su lugar nos queda la idea de un “recurso”, llamémosle “Colección de cervezas” que tiene una URI que lo identifica, p. ej. /Beers. Así, si invoco dicha URI debo obtener una representación de dicho recurso, es decir, debo obtener el listado de todas las cervezas.

Para obtener datos de una cerveza, habrá otro recurso (cerveza) con una URI asociada. Además, entre los recursos hay relaciones: está claro que una cerveza forma parte

de la “Colección de cervezas” así que parece lógico que a partir de la colección entera (/Beers) pueda acceder a uno de sus elementos con una URI tipo /Beers/123, siendo "123" el ID de la cerveza.

Volvamos a nuestro servicio SOAP: para dar de alta una cerveza llamaríamos a un método "AddBeer" de nuestro "BeerService", y le pasaríamos la cerveza a añadir. En cambio, en nuestra API REST añadir una cerveza consiste en usar la URI de dicha cerveza, (p. ej. /Beers/456 si estamos añadiendo la cerveza con ID 456) usan POST o PUT como verbo HTTP y colocando los datos de la cerveza en el cuerpo de la petición. La URI para acceder a la cerveza con ID 456 es siempre /Beers/456 y es el verbo HTTP (GET, POST, PUT, DELETE,...) el que indica cual es la operación que deseamos hacer con esa cerveza.

2.9.3 Métodos del HTTP en REST

En un API REST usamos los métodos del HTTP, existentes desde siempre en el protocolo, aunque hasta ahora infrutilizados en los sitios web clásicos (aquellos basados en contenido), para indicar el tipo de operación que vamos a realizar sobre los recursos que nos ofrecen los servicios web.

Estos métodos también los conocemos como los verbos del HTTP y son los siguientes:

- GET: Para operaciones de consulta de datos. Puedes consultar un listado de cervezas o consultar una cerveza determinada. Ambas operaciones serían GET, solo que la consulta al listado se realizaría sobre el recurso /beer y la consulta a una cerveza sobre el recurso de una cerveza particular /beer/33 o algo parecido.
 - POST: Operaciones de inserción
 - PUT: Para realizar modificaciones
- PATCH: Para realizar modificaciones parciales en un recurso
 - DELETE: Para eliminar un recurso dado.

2.10. *Que es una arquitectura cliente servidor*

Los servicios son utilizados por programas denominados programas clientes que con base a la información realizada por janes en el (entorno cliente servidor, s.f.)

ejecutan en equipos clientes. Por eso se utiliza el término "cliente" (cliente FTP, cliente de correo electrónico, etc.) cuando un programa que se ha diseñado para ejecutarse en un equipo cliente, capaz de procesar los datos recibidos de un servidor (en el caso del cliente FTP se trata de archivos, mientras que para el cliente de correo electrónico se trata de correo electrónico).

Un modelo cliente/servidor se recomienda, en particular, para redes que requieran un alto grado de fiabilidad. Entre las principales ventajas se puede mencionar: recursos centralizados, el servidor es el centro de la red y puede administrar los recursos que son comunes a todos los usuarios, por ejemplo, una base de datos centralizada se utilizaría para evitar problemas provocados por datos contradictorios y redundantes; seguridad mejorada, ya que la cantidad de puntos de entrada que permite el acceso a los datos no es importante; administración al nivel del servidor, los clientes no juegan un papel importante en este modelo, requieren menos administración; red escalable, gracias a esta arquitectura, es posible quitar o agregar clientes sin afectar el funcionamiento de la red y sin la necesidad de realizar mayores modificaciones.

Desventajas del modelo cliente/servidor

La arquitectura cliente/servidor también tiene las siguientes desventajas: costo elevado, debido a la complejidad técnica del servidor; un eslabón débil, el servidor es el único eslabón débil en la red de cliente/servidor, debido a que toda la red está construida en torno a él. Afortunadamente, el servidor es altamente tolerante a los fallos (principalmente gracias al sistema RAID).

Funcionamiento del sistema cliente/servidor

Un sistema cliente/servidor funciona tal como se detalla en el siguiente diagrama:

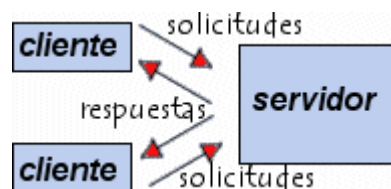


Figura 3 "Figura representa la arquitectura de base de datos obtenidos de;

(entorno cliente servidor, s.f.)

Capitulo III. Procedimiento y Descripción de las Actividades Realizadas

3.1. Instalación de Angular

En esta parte se posicionará dentro de cmd y se ingresará el comando npm install donde se hará referencia a la herramienta de interfaz de angular



Figura 4 “comando npm install” (Datos Obtenidos en el Software CMD).

Se indicarán los paquetes que se instalaron y algunos que se modificaron dentro de las versiones antiguas

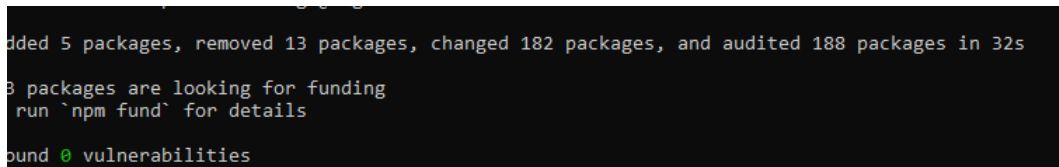


Figura 5: “Mensaje de Instalación al Momento de Instalar el Proyecto”(Datos Obtenidos en el Software CMD).

Se ingresa el comando ng versión para observar la versión en este caso la versión es angular 13 anteriormente era angular 12.0.0 la que se tenía en el área de desarrollo

```
Angular:
...

Package                                Version
-----
@angular-devkit/architect              0.1302.2 (cli-only)
@angular-devkit/core                   13.2.2 (cli-only)
@angular-devkit/schematics             13.2.2 (cli-only)
@schematics/angular                   13.2.2 (cli-only)

C:\Desarrollo> ng new siafe
```

Figura 6. “Versión Actual”. (Datos Obtenidos en el Software CMD).

3.2. Instalación del proyecto

Se ingresa el comando para el proyecto de administración del sitio web de femae-pas.com

Siguiente paso es escribir el comando `ng serve` sirve para que nos muestre que ingresara terminal de nuestro browser el cual se encontrara con la dirección `localhost420` que es propia de angular

```

C:\Desarrollo> ng new siafe
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE siafe/angular.json (3033 bytes)
CREATE siafe/package.json (1068 bytes)
CREATE siafe/README.md (1051 bytes)
CREATE siafe/tsconfig.json (863 bytes)
CREATE siafe/.editorconfig (274 bytes)
CREATE siafe/.gitignore (548 bytes)
CREATE siafe/.browserslistrc (600 bytes)
CREATE siafe/karma.conf.js (1422 bytes)
CREATE siafe/tsconfig.app.json (287 bytes)
CREATE siafe/tsconfig.spec.json (333 bytes)
CREATE siafe/.vscode/extensions.json (130 bytes)
CREATE siafe/.vscode/launch.json (474 bytes)
CREATE siafe/.vscode/tasks.json (938 bytes)
CREATE siafe/src/favicon.ico (948 bytes)
CREATE siafe/src/index.html (291 bytes)
CREATE siafe/src/main.ts (372 bytes)
CREATE siafe/src/polyfills.ts (2338 bytes)
CREATE siafe/src/styles.css (80 bytes)
CREATE siafe/src/test.ts (745 bytes)
CREATE siafe/src/assets/.gitkeep (0 bytes)
CREATE siafe/src/environments/environment.prod.ts (51 bytes)
CREATE siafe/src/environments/environment.ts (658 bytes)
CREATE siafe/src/app/app-routing.module.ts (245 bytes)
CREATE siafe/src/app/app.module.ts (393 bytes)

```

Figura 8 “creación del proyecto”(Datos Obtenidos en el Software CMD).

```

C:\Desarrollo\siafe>ng serve
✓ Browser application bundle generation complete.

Initial Chunk Files | Names          | Raw Size
├── vendor.js       | vendor         | 1.97 MB
├── polyfills.js    | polyfills     | 299.95 kB
├── styles.css, styles.js | styles       | 173.21 kB
├── main.js         | main          | 53.56 kB
├── runtime.js      | runtime       | 6.51 kB
└── Initial Total   |               | 2.49 MB

Build at: 2022-02-07T16:28:28.581Z - Hash: 6b6c3156cdf1862 - Time: 13438ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.

```

Figura 7 “compilamos el código con ng serve”(Datos Obtenidos en el Software CMD).

Ya cuando se termina de compilar te aparece ya levantado un servidor de aplicaciónes

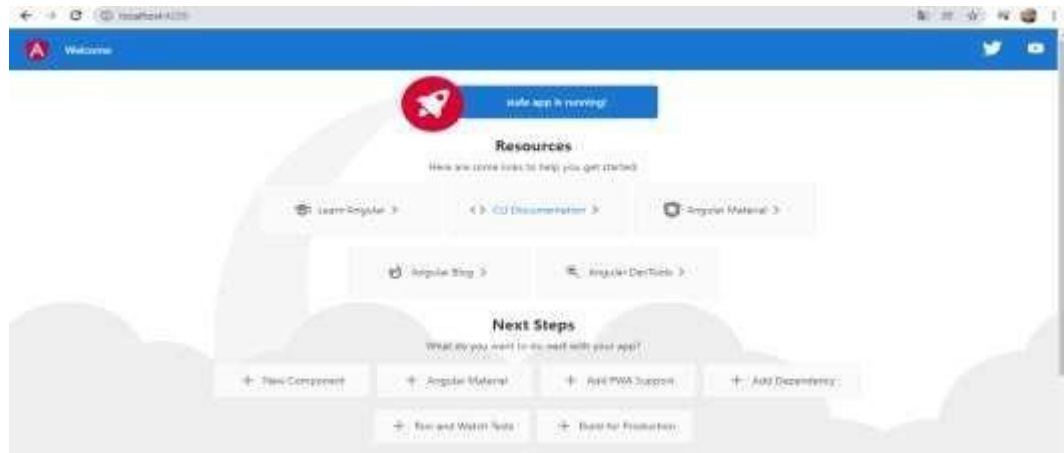


Figura 9 “muestra cómo se llevara la página de angular y como todavía no hay código te muestra la página de angular” ”(Datos Obtenidos en el browser localhost angular host4200).

Abriremos el proyecto en visual studio code

En el cual se usará para poder desarrollar el sistema en la primera parte observaremos las partes que se divide el sistema en la primera que configuraremos es el archivo de índice

En el cual se usará para poder desarrollar el sistema en la primera parte observaremos las partes que se divide el sistema en la primera que configuraremos es el archivo de índice

El índice es la presentación de datos así que lo único que se cambia es el app root

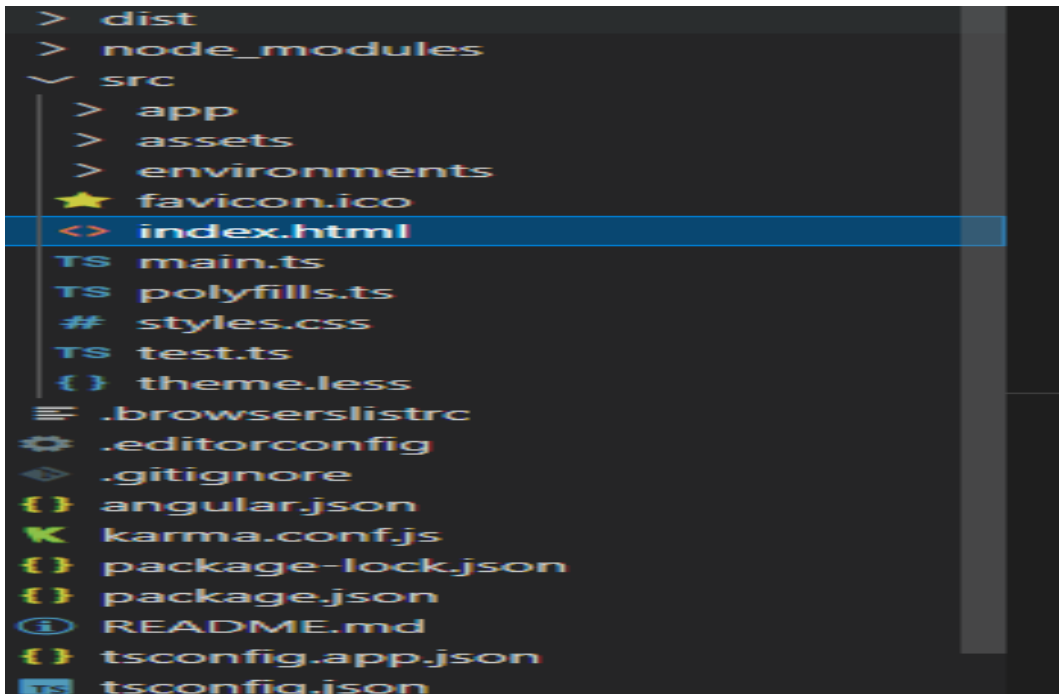


Figura 10 “lista de elementos que se crearon”(Datos Obtenidos en el Software visual basic framework angular 12).

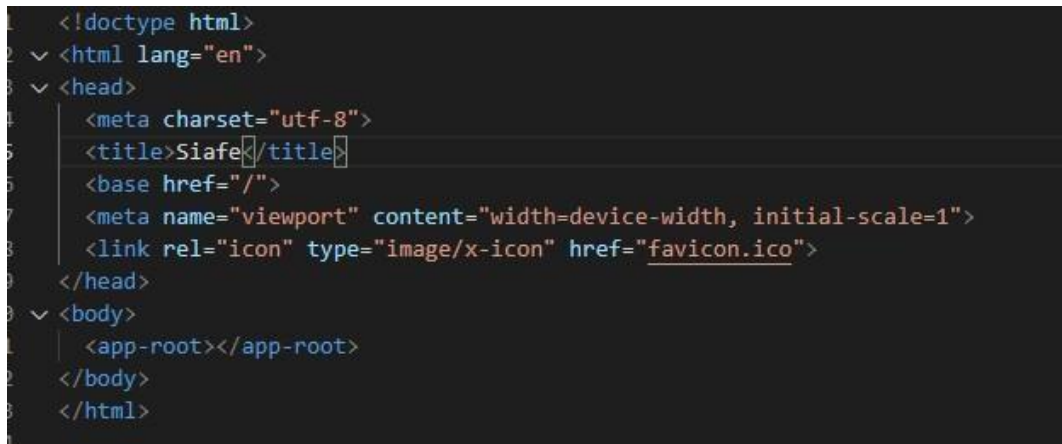


Figura 11 “muestra el código principal”(Datos Obtenidos en el Software visual basic framework angular 12).

Crearemos nuestros componetes de esta manera

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\desarrollo\admins> ng g p /login

```

Figura 12 “se muestra como se configura un nuevo componente”
(Datos Obtenidos en el Software visual basic termial cmd de visual
basic framework angular 12).

Después nos iremos al apartado de app donde tendremos guardado lo necesario para el desarrollo de aplicaciones lo primero a configurar son los componentes del sistema administrativo

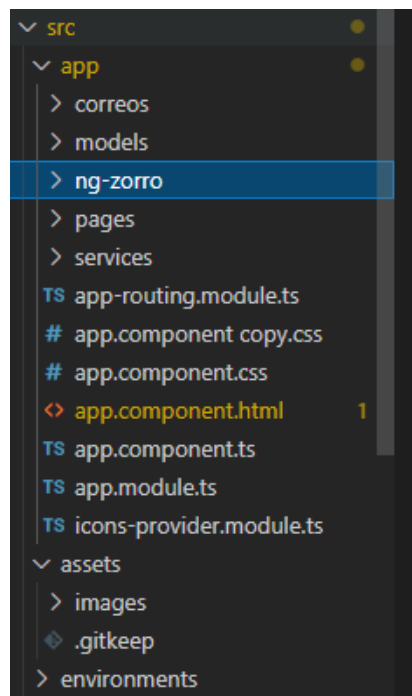
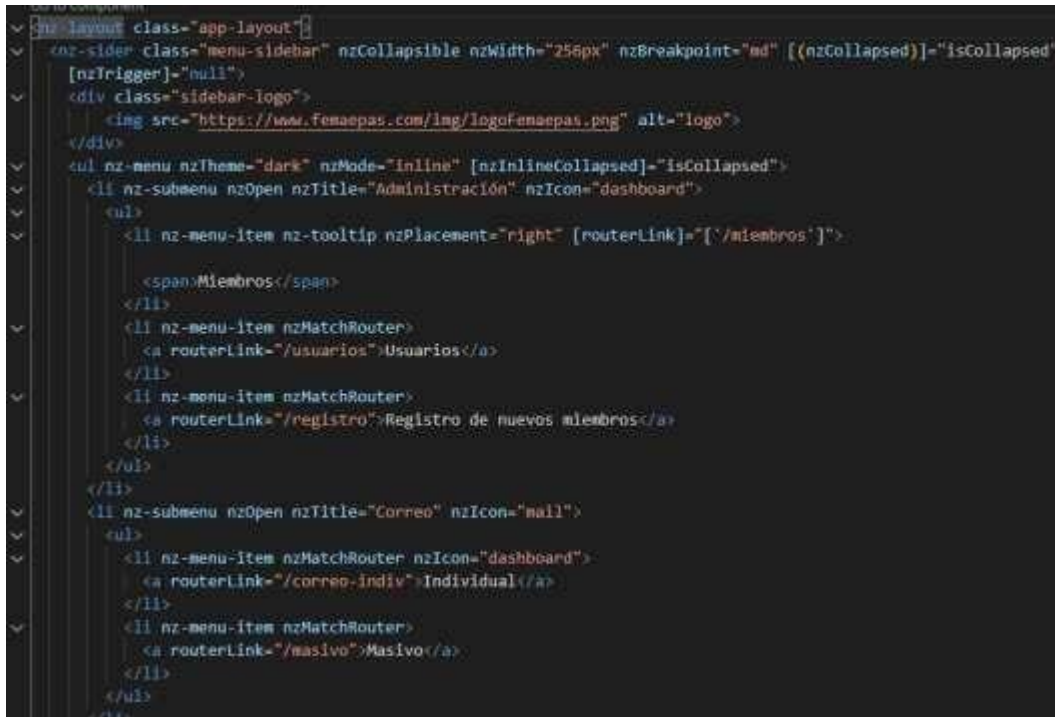


Figura 13 “el contenido de los modulos y componentes que se usaran”(Da-
tos Obtenidos en el Software visual basic framework angular 12).

Empezaremos configurar los módulos del proyecto y componentes empezaremos con el
HTML

Bueno en este HTML se usa para poder configurar la estructura del proyecto en la que sea en una sola pantalla aquí en la imagen puede

observar que hay un menú donde se divide en dos tipos administración y correos

A screenshot of an IDE showing Angular 12 HTML code for a sidebar menu. The code is in Spanish and defines a sidebar layout with a logo and a menu. The menu is divided into two sections: 'Administración' (Administration) and 'Correo' (Mail). The 'Administración' section contains three items: 'Miembros' (Members), 'Usuarios' (Users), and 'Registro de nuevos miembros' (Register new members). The 'Correo' section contains two items: 'Individual' and 'Masivo' (Bulk). The code uses the `nz` prefix for Angular 12's `ng-zorro-antd` components.

```
<div class="app-layout">
  <div class="menu-sidebar" nzCollapsible nzWidth="256px" nzBreakpoint="md" [(nzCollapsed)]="isCollapsed"
    [nzTrigger]="null">
    <div class="sidebar-logo">
      
    </div>
    <ul nz-menu nzTheme="dark" nzMode="inline" [nzInlineCollapsed]="isCollapsed">
      <li nz-submenu nzOpen nzTitle="Administración" nzIcon="dashboard">
        <ul>
          <li nz-menu-item nz-tooltip nzPlacement="right" [routerLink]="['/miembros']">
            <span>Miembros</span>
          </li>
          <li nz-menu-item nzMatchRouter>
            <a routerLink="/usuarios">Usuarios</a>
          </li>
          <li nz-menu-item nzMatchRouter>
            <a routerLink="/registro">Registro de nuevos miembros</a>
          </li>
        </ul>
      </li>
      <li nz-submenu nzOpen nzTitle="Correo" nzIcon="mail">
        <ul>
          <li nz-menu-item nzMatchRouter nzIcon="dashboard">
            <a routerLink="/correo-indiv">Individual</a>
          </li>
          <li nz-menu-item nzMatchRouter>
            <a routerLink="/masivo">Masivo</a>
          </li>
        </ul>
      </li>
    </ul>
  </div>
</div>
```

Figura 14 “se muestra la parte de los componentes que se crearon y el login por default aparece primero” (Datos Obtenidos en el Software visual basic framework angular 12).

Aquí nos indica que el contenido de la página se mostrara la izquierda y que se importan las clases css para darle formato ala pagina

```

</ul>
<nz-sider>
<nz-layout>
<nz-header>
  <div class="app-header">
    <span class="header-trigger" (click)="isCollapsed = !isCollapsed">
    </span>
  </div>
</nz-header>
<nz-content>
  <div class="inner-content">
    <router-outlet></router-outlet>
  </div>
</nz-content>
</nz-layout>
</nz-layout>

```

Figura 15 “partes del código de appcomponent” (Datos Obtenidos en el Software visual basic framework angular 12).

Iremos a ingresar las librerías de ciertos módulos que se usaran en el desarrollo

```

> app > ts app.module.ts > ...
1  import { CommonModule, registerLocaleData } from '@angular/common'; 51.9K (gzipped: 16.7K)
2  import { FormsModule, ReactiveFormsModule } from '@angular/forms'; 64.1K (gzipped: 14.1K)
3
4  import { AppComponent } from './app.component';
5  import { AppRoutingModule } from './app-routing.module';
6  import { BrowserAnimationsModule } from '@angular/platform-browser/animations'; 80.4K (gzipped: 25.8K)
7  import { BrowserModule } from '@angular/platform-browser'; 20.3K (gzipped: 5.6K)
8  import { CorreoIndivComponent } from './pages/correo-indiv/correo-indiv.component';
9  import { CorreoMasivComponent } from './pages/correo-masiv/correo-masiv.component';
10 import { HttpClientModule } from '@angular/common/http'; 81K (gzipped: 24.3K)
11 import { IconsProviderModule } from './icons-provider.module';
12 import { LoginComponent } from './pages/login/login.component';
13 import { MiembrosComponent } from './pages/miembros/miembros.component';
14 import { NgModule } from '@angular/core'; 110.4K (gzipped: 35K)
15 import { NgZorroModule } from './ng-zorro/ng-zorro.module';
16 import { NzLayoutModule } from 'ng-zorro-antd/layout';
17 import { NzMenuModule } from 'ng-zorro-antd/menu';
18 import { NzModalModule } from 'ng-zorro-antd/modal'; 372.4K (gzipped: 95.3K)
19 import { RegistroComponent } from './pages/registro/registro.component';
20 import { UsuariosComponent } from './pages/usuarios/usuarios.component';
21 import en from '@angular/common/locales/en'; 1.9K (gzipped: 997)

```

Figura 16 “librerías del appcomponent” (Datos Obtenidos en el Software visual basic framework angular 12).

Y en esta parte se crearán las variables que se utilizaran

```

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    MiembrosComponent,
    RegistroComponent,
    CorreoIndivComponent,
    CorreoMasivComponent,
    UsuariosComponent,
  ],
  imports: [
    CommonModule,
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    BrowserAnimationsModule,
    NgZorroModule,
    IconsProviderModule,
    NzLayoutModule,
    NzMenuModule,
    NzModalModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Figura 17 “variable de los modulos” (*Datos Obtenidos en el Software visual basic framework angular 12*).

En esta parte se introducirá la parte de los componentes en donde irán al momento de darle click al menú si no encuentra ninguna ruta se ira a por default al login

```
import { RouterModule, Routes } from '@angular/router'; 94.8K (gzipped: 24.3K)
import { CorreoIndivComponent } from './pages/correo-indiv/correo-indiv.component';
import { CorreoMasivComponent } from './pages/correo-masiv/correo-masiv.component';
import { LoginComponent } from './pages/login/login.component';
import { MiembrosComponent } from './pages/miembros/miembros.component';
import { NgModule } from '@angular/core'; 110.4K (gzipped: 35K)
import { RegistroComponent } from './pages/registro/registro.component';
import { UsuariosComponent } from './pages/usuarios/usuarios.component';

const routes: Routes = [
  { path: '', pathMatch: 'full', redirectTo: '/login' },
  {path: 'login', component: LoginComponent},
  {path: 'miembros', component: MiembrosComponent},
  {path: 'masivo', component: CorreoMasivComponent},
  {path: 'registro', component: RegistroComponent},
  {path: 'correo-indiv',component:CorreoIndivComponent},
  {path:'usuarios',component:UsuariosComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Figura 18 “rutas dentro de los datos” (Datos Obtenidos en el Software visual basic framework angular 12).

Y también se ingresarán los módulos de zorro

```
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { NgAffixModule } from 'ng-servo-antd/affix';
import { NGAlertModule } from 'ng-servo-antd/alert';
import { NGAnterModule } from 'ng-servo-antd/anchor';
import { NzAutoCompleteModule } from 'ng-servo-antd/auto-complete';
import { NzAvatarModule } from 'ng-servo-antd/avatar';
import { NzBackTopModule } from 'ng-servo-antd/back-top';
import { NzBadgeModule } from 'ng-servo-antd/badge';
import { NzBreadcrumbModule } from 'ng-servo-antd/breadcrumb';
import { NzButtonModule } from 'ng-servo-antd/button';
import { NZCalendarModule } from 'ng-servo-antd/calendar';
import { NzCardModule } from 'ng-servo-antd/card';
import { NzCarouselModule } from 'ng-servo-antd/carousel';
import { NzCascaderModule } from 'ng-servo-antd/cascader';
import { NzCheckboxModule } from 'ng-servo-antd/checkbox';
import { NzCollapseModule } from 'ng-servo-antd/collapse';
import { NzCommentModule } from 'ng-servo-antd/comment';
import { NzDateTimeModule } from 'ng-servo-antd/core/no-animation';
import { NzDateTimeModule } from 'ng-servo-antd/core/no-transition'; // 779 (skipped: 451)
import { NzCoreModule } from 'ng-servo-antd/core/wave';
import { NzDatePickerModule } from 'ng-servo-antd/date-picker';
import { NzDescriptionsModule } from 'ng-servo-antd/descriptions';
import { NzDividerModule } from 'ng-servo-antd/divider';
import { NzDrawerModule } from 'ng-servo-antd/drawer';
import { NzDropdownModule } from 'ng-servo-antd/dropdown';
import { NzEmptyModule } from 'ng-servo-antd/empty';
import { NzFormModule } from 'ng-servo-antd/form';
import { NzGridModule } from 'ng-servo-antd/grid';
import { NzIconModule } from 'ng-servo-antd/icon';
import { NzListModule } from 'ng-servo-antd/list';
import { NzModalModule } from 'ng-servo-antd/modal';
```

Figura 19 “libreria de ng zoro”(Datos Obtenidos en el Software visual basic framework angular 12).

```

2  @NgModule({
3    exports: [
4      CommonModule,
5      NzAffixModule,
6      NzAlertModule,
7      NzAnchorModule,
8      NzAutocompleteModule,
9      NzAvatarModule,
10     NzBackTopModule,
11     NzBadgeModule,
12     NzButtonModule,
13     NzBreadCrumbModule,
14     NzCalendarModule,
15     NzCardModule,
16     NzCarouselModule,
17     NzCascaderModule,
18     NzCheckboxModule,
19     NzCollapseModule,
20     NzCommentModule,
21     NzDatePickerModule,
22     NzDescriptionsModule,
23     NzDividerModule,
24     NzDrawerModule,
25     NzDropDownModule,
26     NzEmptyModule,
27     NzFormModule,
28     NzGridModule,
29     NzI18nModule,
30     NzIconModule,
31     NzImageModule,
32     NzInputModule,

```

Figura 20 “se exportan las variables de ng-zorro” (Datos Obtenidos en el Software visual basic framework angular 12).

```

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\desarrollo\admin> ng g p /login

```

Figura 21 “ejemplo de como crear un componente de login”(Datos Obtenidos en el Software visual basic terminal cmd framework angular 12).

Configuraremos el área de páginas llamada pages la primera que configuraremos la de login

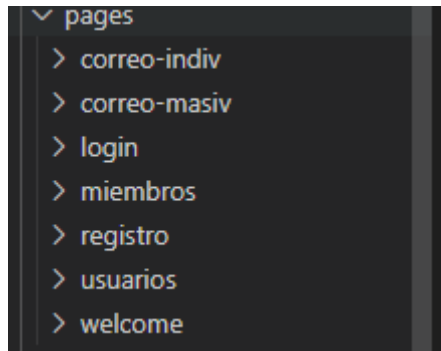


Figura 22 “creación de los componentes”(Datos Obtenidos en el Software visual basic framework angular 12).

Angular se maneja en componentes css, HTML, desarrollo de funcionalidades en typescript

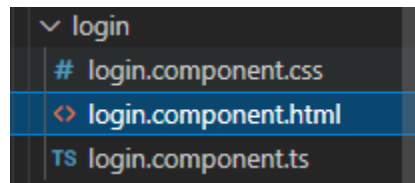


Figura 23 “Componentes del login” (Datos Obtenidos en el Software visual basic framework angular 12).

Creamos el código en HTML donde la primera columna se divide en la imagen representativa de nosotros el inicio de sesión el botón donde tiene un validate from que lo usaremos como formulario de validación el botón

Los campos de donde se ingresarán los comandos

```


Figura 24 “implementación de HTML”(Datos Obtenidos en el Software visual basic framework angular 12).



En esta parte tendremos los comandos donde se ingresará el usuario la contraseña y se verá encriptada para no mostrarla



```

16 [(ngModel)]="dataLogin.cuenta" />
17 </nz-input-group>
18 </nz-form-control>
19 </nz-form-item>
20 <nz-form-item>
21 <nz-form-control nzErrorTip="Ingresa la Contraseña!">
22 <nz-input-group nzPrefixIcon="lock">
23 <input type="password" nz-input formControlName="password" placeholder="Contraseña"
24 [(ngModel)]="dataLogin.password" />
25 </nz-input-group>
26 </nz-form-control>
27 </nz-form-item>
28
29 <nz-form-item>
30 <nz-form-control>
31 <button nz-button class="login-form-button" [nzType]="primary"
32 [nzLoading]="isLoading" >{{ isLoading ? 'Espere': 'Ingresar' }} </button>
33 </nz-form-control>
34 </nz-form-item>
35 </form>
36
37 Version 1.0
38 </div>
39 </div>
40 </div>

```



Figura 25 “HTML de login en la parte de validacion de el password”(Datos Obtenidos en el Software visual basic framework angular 12).



Ahora iremos al js



Donde estarán los importados los componentes necesarios para darle la funcionalidad al objeto en cuestión



El isvisible nos mostrara cuando se actualiza la página él nos muestra el contenido de las paginas



55


```

Size decimos que es de tipo botón nos mostrara el tamaño que se le pondrá al botón

La siguiente parte es el tiempo de espera para la aplicación

Y el validate Form es de FormGroup una herramienta de zorro que nos permite validar los campos vacio

```
app / pages / login / logincomponent.ts LoginComponent
import { Component, OnInit } from '@angular/core'; 110.7K (gzipped: 35.1K)
import { FormBuilder, FormGroup, Validators } from '@angular/forms'; 64.4K (gzipped: 14.2K)

import { LoginI } from '../models/login';
import { LoginService } from '../services/login.service';
import { NzButtonSize } from 'ng-zorro-antd/button'; 165.5K (gzipped: 47.8K)
import { NzMessageService } from 'ng-zorro-antd/message'; 199.5K (gzipped: 52.4K)
import { Router } from '@angular/router'; 94.6K (gzipped: 24.2K)

Complexity is 4 Everything is cool!
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  isVisible = false;
  size: NzButtonSize = 'large';
  validateForm!: FormGroup;
  isAwaiting: boolean;
  loginFailed: boolean;
  datalogin : LoginI;
```

Figura 26 “importacion de librerias y de variables utilizadas en el proyecto” (Datos Obtenidos en el Software visual basic framework angular 12).

La data login es nuestro objeto donde tendremos guardado el arreglo

Y él se ubica en los modules donde se guardan los datos de cuenta y password del usuario la otra es para conseguir su estatus si es activo o inactivo y la validación de la respuesta

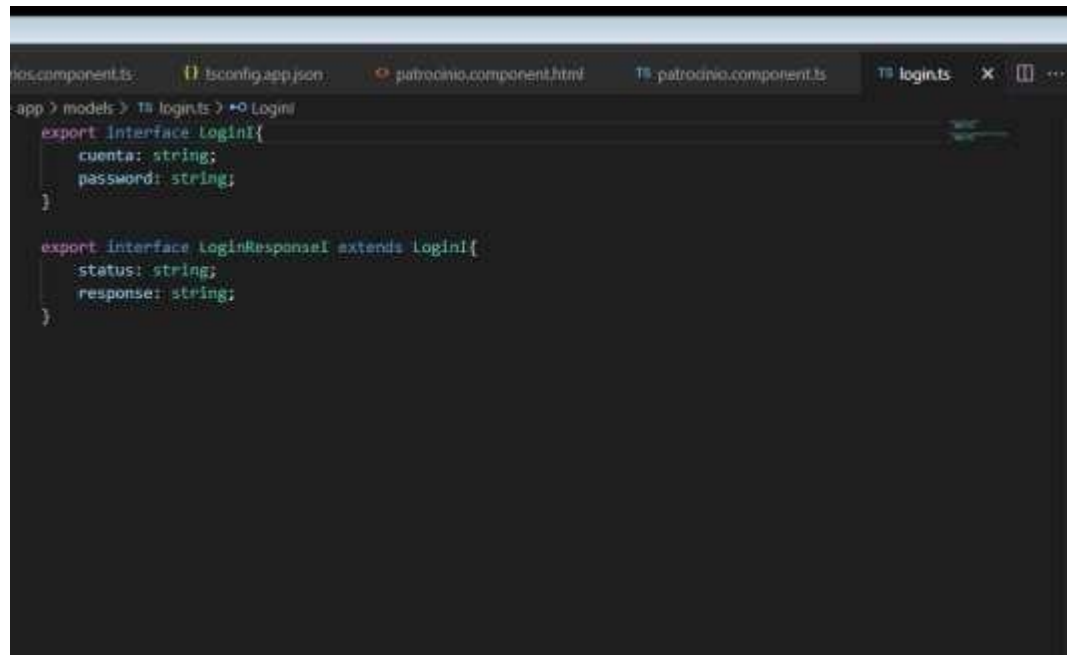


Figura 27 “creación de modulo y respuesta de los datos” (Datos Obtenidos en el Software visual basic framework angular 12).

Esta función es relacionada por el botón de ingresar, pero en esta parte se valida que los campos estén llenos



Figura 28 “Funcion donde se guardaran los datos la validacion de los campos” (Datos Obtenidos en el Software visual basic framework angular 12).

Esta parte es la constructora en ese usaran las inicializaciones de funciones como lo servicios donde se llaman funciones de backend

Después de llamar los servicios inicializado el objeto llama su contenido del arreglo

```

}
constructor(
  private fb: FormBuilder,
  private router: Router,
  private loginsvr: LoginService,
  private message: NzMessageService
) {
  this.isAwaiting = false;
  this.loginFailed = false;
  this.dataLogin = {
    cuenta: '',
    password: ''
  }
}

```

Figura 29 “Implementación de la variable del constructor” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta parte ingresaremos en el oninit los campos que se llegan a validar en algunos casos igual se usa para mandar a llamar los datos de una tabla ya que el oninit es la primera función que se inicia en el sistema

En la función del login se valida que la función de la búsqueda sea igual a los campos del module el suscribe es un observable que se puede mandar a llamarla base de datos

Después se hace una comparación donde sea diferente mande error y si no que mande mensaje de listo

Y por ultima creamos un mensaje de tipo string crean el tipo de mensaje

```

    ngOnInit(): void {

        this.validateForm = this.fb.group({
            userName: [null, [Validators.required]],
            password: [null, [Validators.required]],
        });
    }

    Complexity is 4 Everything is cool!
    login(): void {

        Complexity is 3 Everything is cool!
        this.loginsvr.loginByEmail(this.dataLogin).subscribe((res) => {
            if(res==null){
                this.creamensaje('error');
            }else{
                this.creamensaje('success');
            }
        });

        creamensaje(type: string):void {
            this.message.create(type, `This is a message of ${type}`);
        }
    }

```

Figura 30 “el oinit se usara para la validacion de los campos y la funcion login se mandara a hacer para logear a la persona” (Datos Obtenidos en el Software visual basic framework angular 12).

Ahora crearemos la función del loginemail en el código
 En esta parte se ingresarán las importaciones de los modelos el observable y el ambiente donde se vincula al backend en la función se indicará que los datos ingresados vienen de la interface login y el observable es loginresponse donde se introdujo la respuesta
 Y se ingresa un return donde se regresarán datos mediante la urlbase de login

```

app / services / loginServices / ...
import { LoginI, LoginResponseI } from '../models/login';
import { Observable, from } from 'rxjs';

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { JwtHelperService } from '@auth0/angular-jwt';
import { environment } from 'src/environments/environment';

const helper = new JwtHelperService();

@Injectable({
  providedIn: 'root'
})
export class LoginService {

  constructor(
    private http: HttpClient
  ) { }

  loginByEmail(form: LoginI): Observable<LoginResponseI> {
    console.log("datos", form);
    return this.http.post<LoginResponseI>(`${environment.Urlbase}/login`, form);
  }
}

```

Figura 31 “servicio del login” (Datos Obtenidos en el Software visual basic framework angular 12)

Generación del componente de miembros

Después se ingresará el formato de miembros se compondrá por un componente de registro

Y del y uno donde contendrá el miembro de la tabla

Se empezará con el componente registrar

Cual los ingresaremos las interfaces en la cual se indicarán como se viajarán los datos en el arreglo

En esta parte se ingresarán interfaces que nos ayudarán para el resguardo de datos una la usaremos para la captura y la otra para traer los datos del backend

```
export type Roles = 'miembro' | 'admin';

export interface Miembro {
  nombre: string;
  apellidop: string;
  apellidom: string;
  telefono: string;
  email: string;
  estado: string;
  profesion: string;
  escolaridad: string;
  especialidad: string;
  cedula: string;
  claverreferido: string;
  password: string;
  membresia: number;
  tipomembresia: string;
  role: string;
  fechareg: Date;
}
```

Figura 32 “interface del modulo de miembros” (Datos Obtenidos en el Software visual basic framework angular 12).

Este se usará para extraer el id y después desplegarlo en la tabla correspondiente

```

export interface MiembroResponse extends Miembro {
  id: number;
  message: string;
  nombre: string;
  apellidop: string;
  apellidom: string;
  telefono: string;
  email: string;
  estado: string;
  profesion: string;
  escolaridad: string;
  especialidad: string;
  cedula: string;
  claverreferido: string;
  password: string;
  membresia: number;
  tipomembresia: string;
  role: Roles;
  fechareg: Date;
}

```

Figura 33 “respuesta de datos” (Datos Obtenidos en el Software visual basic framework angular 12).

En este paso se ingresará el HTML de registro donde irán guardados los datos. Esta parte de código se dividirá por reglones en el cual la primera ira un titulo. Después un renglón donde se pondrán la opción de la membresía donde la membresía puede ser dorada plateada o Premium después ingresaremos el botón de registro de datos y algunos colores de fondo.

```

<div class="main-registro">
<div style="font-size: 20px; text-align: center; height: 0.5px;">
  <span>Datos necesarios para su registro </span>
</div>

<div nz-row >
  <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="6" nzXl="6" nzXxl="6" style="padding-top: 50px;">
    <nz-radio-group [(ngModel)]="radioValue" (ngModelChange)="open()">
      <label nz-radio nzValue="0">Silver</label>
      <label nz-radio nzValue="1">Gold</label>
      <label nz-radio nzValue="2">platinum</label>
    </nz-radio-group>
    <img nz-image class="imagen" *ngIf="invisible"
      [nzSrc]="membresia" alt="" />
    </div>

  <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="16" nzXl="16" nzXxl="16" style="padding-top: 50px;">
    <form nz-form [formGroup]="validateForm" (ngSubmit)="registrar()">
      <hr style="height:5px; background-color : rgba(103, 155, 233, 0.753)">
      <br/>
    </form>
  </div>
</div>

```

Figura 34 “HTML de registro”(Datos Obtenidos en el Software visual basic framework angular 12).

Después irán las divisiones de las columnas y que renglones se acomodara todo el primero reglón son para datos personales y los otros son para el nivel de estudios que manejan

```

<img nz-image class="imagen" *ngIf="invisible"
[nzSrc]="membresia" alt="" />
</div>

<div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="16" nzXl="16" nzxxl="16" style="padding-top: 56
<form nz-form [formGroup]="validateForm" (ngSubmit)="registrar()">
  <hr style="height:5px; background-color : rgba(103, 155, 233, 0.753)">
  <br/>
  <div nz-row [nzGutter]="24">
    <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzxxl="7">
      <!-- [nzSpan]="6" -->
      <nz-form-item>
        <nz-form-control nzErrorTip="Ingresa Nombre!">
          <input nz-input placeholder="Nombre(s)" type="text"
            formControlName="nombre" [(ngModel)]= "DataRegistro.nombre" >
        </nz-form-control>
      </nz-form-item>
    </div>
    <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzxxl="7" >
      <!-- [nzSpan]="5" -->
      <nz-form-item>
        <nz-form-control>
          <input nz-input placeholder="Apellido paterno" type="text"
            formControlName="apellidoop" [(ngModel)]= "DataRegistro.apellidoop" >
          </nz-form-control>
        </nz-form-item>
      </div>
    <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzxxl="7" >
      <!-- [nzSpan]="5" -->
      <nz-form-item>
        <nz-form-control nzErrorTip="Ingresa Apellido Materno!">
          <input nz-input placeholder="Apellido materno" type="text"
            formControlName="apellidom" [(ngModel)]= "DataRegistro.apellidom" >
          </nz-form-control>
        </nz-form-item>
      </div>
    </div>
    <div nz-row [nzGutter]="24">
      <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzxxl="7">
        <!-- [nzSpan]="7" -->
        <nz-form-item>
          <nz-form-control nzErrorTip="Ingresa Telefono!">
            <input nz-input placeholder="Telefono" type="text"
              formControlName="telefono" [(ngModel)]= "DataRegistro.telefono" >
            </nz-form-control>
          </nz-form-item>
        </div>

```

Figura 35 “Código del HTML” (Datos Obtenidos en el Software
visual basic framework angular 12).

```

pages > registro > registro.component.html > div.main-registro

<!-- [nzSpan]="5" -->
<nz-form-item>
  <nz-form-control nzErrorTip="Ingresa Nivel de Escolaridad!">
    <nz-select nzShowSearch nzAllowClear
      nzPlaceholder="Nivel de escolaridad"
      FormControlName="escolaridad" [(ngModel)]="DataRegistro.escolaridad" >
      <nz-option nzLabel="Primaria" nzValue="Primaria"></nz-option>
      <nz-option nzLabel="Secundaria"
        nzValue="Secundaria"></nz-option>
      <nz-option nzLabel="Preparatoria"
        nzValue="Preparatoria"></nz-option>
      <nz-option nzLabel="Universidad"
        nzValue="Universidad"></nz-option>
      <nz-option nzLabel="Maestría" nzValue="Maestría"></nz-option>
      <nz-option nzLabel="Posgrado" nzValue="Posgrado"></nz-option>
    </nz-select>
  </nz-form-control>
</nz-form-item>
</div>
<div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzXl="7">
  <!-- [nzSpan]="7" -->
  <nz-form-item>
    <nz-form-control nzErrorTip="Ingresa Especialidad!">
      <input nz-input placeholder="Área de especialidad"
        FormControlName="especialidad" [(ngModel)]="DataRegistro.especialidad"
        type="text">
    </nz-form-control>
  </nz-form-item>
</div>
<div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzXl="7">
  <!-- [nzSpan]="7" -->
  <nz-form-item>
    <nz-form-control nzErrorTip="Ingresa Cédula Profesional!">
      <input nz-input placeholder="Cédula Profesional"
        FormControlName="cedula" [(ngModel)]="DataRegistro.cedula"
        type="text">
    </nz-form-control>
  </nz-form-item>
</div>

```

Figura 36 “Código HTML” (Datos Obtenidos en el Software visual
basic framework angular 12).

Después ingresaremos un desplegable para referidos o sea personas que les vendieron las membrecías estos estarán guardados en la base de datos en la cual tiene su id su nombre y el dato

```

    nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="7" nzXl="7" nzxXl="7">
    z-form-item>
    <nz-form-control nzErrorTip="Ingresa Nombre del referido!">
    <nz-select nzShowSearch nzAllowClear
    nzPlaceholder="Referido por ..."
    formControlName="clavereferido" [(ngModel)]= "DataRegistro.clavereferido" >
    <nz-option [nzValue]="item.clave" *ngFor="let item of listOFDRUMCLAVEREF"
    [nzLabel]="item.nombre" </nz-option>

```

Figura 37 “Implementación del nz- select” (Datos Obtenidos en el Software visual basic framework angular 12).

Y después se generará el botón donde si hay un error y se mostrara una alerta de espera si no se ira a la función registro

```

    </nz-form-item>
  </div>
</div>
<hr style="height:5px; background-color : rgba(103, 155, 233, 0.753)">
<br/>
<div nz-row>
  <div nz-col nzXs="5" nzSm="5" nzMd="5" nzLg="5" nzXl="5" nzxXl="5">

  </div>
</div>
<div nz-row >
  <div nz-col nzXs="24" nzSm="24" nzMd="24" nzLg="5" nzXl="5" nzxXl="5"nzOffset="6">
    <nz-form-item>
      <button nz-button class="form-button" [nzType]='primary'
      [nzLoading]="isAwaiting">{{ isAwaiting ? 'Espere': 'Registrar'
      }}</button>
    </nz-form-item>
  </div>
</div>
</form>
</div>
</div>

```

Figura 38 “Implementación del botón” (Datos Obtenidos en el Software visual basic framework angular 12).

Una vez todo hecho el proceso se ingresará una segunda pantalla donde se mandará la

bienvenida y los datos necesarios

```
/div>
<nz-modal
  [(nzVisible)]= "miembroVisible" nzTitle="Inscripción lista!! " nzCentered [nzFooter]="modalB
  <ng-container *nzModalContent>
    <div style="text-align: center">
      <h1>Bienvenido(a)</h1>
      <p>{{DataRegistro.nombre}} {{DataRegistro.apellido}} {{DataRegistro.apellido}}</p>
      <span> MEMBRESÍA {{DataRegistro.tipomembresia}} No. {{numMembresia}}</span>
    </div>
    <br>
    <div>
      <tr>
        <td class="pin">
          <p>En nombre de FEMAEPAS le doy la bienvenida oficialmente como asociado a nuestra Fe
            con gran orgullo y entusiasmo, me permito animarle a que acceda a nuestro portal (w
            y redes sociales, para que conozca los beneficios que le ofrecemos con su Membresía
          </p>
        </td>
      </tr>
    </div>
  </ng-container>
  <ng-template #modalBotones>
    <button nz-button nzType="primary" (click)="miembroOk()" >Cerrar</button>
  </ng-template>
</nz-modal>
<nz-modal [(nzVisible)]= "modalVisible" nzTitle="TÉRMINOS Y CONDICIONES" [nzFooter]="modalFooter"
  <ng-container *nzModalContent>
    <h3>ÁMBITO DE APLICACIÓN</h3>
    <tr>
      <td class="pin">
        <p>Bienvenido al sitio web www.femaepas.com y sitios de internet relacionados,
          te rogamos que leas detenidamente los presentes términos y condiciones antes
          de utilizar el sitio web, ya que al entrar a nuestro sitio confirmas tu entendimiento
        </p>
      </td>
    </tr>
    <tr>
      <td class="pin">
        <p>Debido al contenido y la finalidad del sitio web, las personas que quieran beneficia
          de sus servicios deben de suscribirse, lo cual se realiza a través del llenado del fo
          de registro y siguiendo los pasos que con posterioridad se les comunicará por medio
          electrónico para dichos efectos, la condición de suscriptor supone la aceptación de
          términos y condiciones de uso.
        </p>
      </td>
    </tr>
  </ng-container>
```

Figura 39 “Plantilla del drawer” (Datos Obtenidos en el Software visual basic framework angular 12).

Ahora crearemos el ts de registro y sus servicios al igual que haremos el modulo para los referidos

```
export interface Clavereferido {  
  clave: string;  
  nombre:string;  
  id_status: number;  
  fecha_reg: string;  
}  
  
export interface clavreferidoResponse extends Clavereferido {  
  id: number;  
  message: string;  
  clave: string;  
  nombre:string;  
  id_status: number;  
  fecha_reg: string;  
}
```

Figura 40 “modulo de entrada de datos de respuesta y de arreglo”
(Datos Obtenidos en el Software visual basic framework angular
12).

El servicio de clave referidos se importa el httpclient donde lo usaremos para que pueda hacer comunicación con el backend

Después obtenemos dos funciones una de consulta donde getall se ara como consulta en los datos mediante la urrl del id en esta parte se ira ala interface response ya que cuando se da de alta una persona referida guarda en la base de datos y buscaremos por el id dentro de localhost del backend y la ruta referida con el método de búsqueda get

En la función registro agregaremos el objeto del clave referido de la interface, pero se observará como respuesta el response el cual se usará para que el registro tenga su propia id

Se usará para dos componentes el registro y el de consulta de la tabla en el cual

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';
import { environment } from 'src/environments/environment';
import { Clavreferido, clavreferidoResponse } from '../models/clavreferido';

@Injectable({
  providedIn: 'root'
})
export class ClavreferidoService {

  constructor(
    private http: HttpClient
  ) {}

  getAll(): Observable<clavreferidoResponse[]> {
    return this.http.get<clavreferidoResponse[]>(`${environment.Urlbase}/referido/all`);
  }

  registrar(regData: Clavreferido): Observable<clavreferidoResponse | void> {
    console.log('agregar', regData);
    return this.http.post<clavreferidoResponse>(`${environment.Urlbase}/referido`, regData);
  }
}
```

Figura 41 “servicio de miembros” (Datos Obtenidos en el Software visual basic framework angular 12).

Después pasaremos a construir el proceso del servicio de registro en este servicio se comparte con el componente de miembros bueno las funciones que usaremos son la de registrar en esta parte del código usaremos los datos de miembro y el http cliente e importar el ambiente

```

admin > src > app > services > TS miembro.service.ts > ...
{ BehaviorSubject, Observable, throwError } from 'rxjs';
import { Miembro, MiembroResponse } from '../models/miembro';
import { catchError, map } from 'rxjs/operators';

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { JwtHelperService } from '@auth0/angular-jwt';
import { Router } from '@angular/router';
import { environment } from 'src/environments/environment';

const helper = new JwtHelperService();

@Injectable({
  providedIn: 'root'
})
export class MiembroService {
  [x: string]: any;

```

Figura 42 “librerías del servicios” (Datos Obtenidos en el Software visual basic framework angular 12).

Después de exportar la clase haremos su constructor

Donde usaremos el httpclient que es el que permitirá la comunicación entre la base de datos y la aplicación y el root crearemos las funciones siguientes el get all será el que se llamará para poder ver el dato del miembro en la base de datos e imprimir la consulta en una tabla

```

✓ constructor() {
  private http: HttpClient,
  private router: Router {
    //this.checkToken();
  }
}

✓ getAll(): Observable<MiembroResponse[]> {
  return this.http.get<MiembroResponse[]>(`${environment.Urlbase}/miembros/all`);
}

```

Figura 43 “Constructor del servicio y función get all” (Datos Obtenidos en el Software visual basic framework angular 12).

En la parte de registro se usa la función registro ya que le mandaremos la interfaz de

miembros que se encuentra en el módulo y nos regresara los datos de la interface response ya que se necesitan para poder hacer el post en el cual se consigue mediante una url que te permite interactuar con el backend

Ahora tendremos la función de edición donde usaremos el match que se usa para poder partir de la búsqueda del id y la edición de nuevos campos

```
✓ registrar(regData: Miembro): Observable<MiembroResponse | void>={  
  console.log('DataRegistro', regData);  
  return this.http.post<MiembroResponse>(`${environment.Urlbase}/miembros`, regData);  
}  
  
✓ Editar(meditData: MiembroResponse): Observable<MiembroResponse | void>={  
  console.log('edita', meditData);  
  return this.http.patch<MiembroResponse>(`${environment.Urlbase}/miembros/  
    ${meditData.id}`, meditData);  
}  
}
```

Figura 44 “funciones de editar y agregar” (Datos Obtenidos en el Software visual basic framework angular 12).

Ahora iremos a la configuración ts de registrar el ts es de typescript ahí se realizarán las funciones

Esta parte de las imágenes es para activar las rutas de angular importar las funcionalidades de los correos o los miembros después los servicios de miembros y clave referidos

```

import { ActivatedRoute, Router } from '@angular/router';
import { Component, OnDestroy, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { ClavereferidoService } from 'src/app/services/claveref.service';
import { CorreoService } from 'src/app/services/correo.service';

import { Miembro } from 'src/app/models/miembro';
import { MiembroService } from 'src/app/services/miembro.service';
import { Miembrosemail } from '.../correos/miembross';
import { NzButtonSize } from 'ng-zorro-antd/button';

import { NzModalService } from 'ng-zorro-antd/modal';
import { Subscription } from 'rxjs';
import { clavereferidoResponse } from 'src/app/models/clavereferido';

```

Figura 45 “librerías de registro” (Datos Obtenidos en el Software visual basic framework angular 12).

Aquí vamos a crear variables de tipo universal ósea que se usaran en varias partes el código algunas son booleanas ya que solo tiene valores de falso y verdadero y el fromvalidate es el que nos ayuda a validar si no tenemos campos vacíos

```

terminos = false;
modalVisible = false;
miembroVisible = false;
isfirst: boolean;
isAwaiting: boolean;
radioValue = '0';
invisible = true;
validateForm!: FormGroup;

```

Figura 46 “variables para ingresar en el ts” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta parte se creará nuestro arreglo donde usaremos la interface miembro para interactuar en la aplicación

```

DataRegistro: Miembro = {
  nombre: '',
  apellidop: '',
  apellidom: '',
  telefono: '',
  email: '',
  estado: '',
  profesion: '',
  escolaridad: '',
  especialidad: '',
  cedula: '',
  claverreferido: '',
  password: '',
  membresia: 0,
  tipomembresia: '',
  role: 'miembro',
  fechareg: new Date()
};

```

Figura 47 “objeto miembro” (Datos Obtenidos en el Software visual
basic framework angular 12).

Se ingresa un arreglo para poder hacer el envío de datos para el correo

```

dataCorreo = {
  asunto: '',
  email: '',
  mensaje: '',
  nombre: '',
}

```

Figura 48 “objeto correo” (Datos Obtenidos en el Software visual
basic framework angular 12).

Se creará un valor membresía de tipo any y también un arreglo de imagen donde se usará
para mostrar el tipo de membresía en la aplicación web

```

membresia: any;
arrayimg = [
  {
    img: './assets/images/MMSilver.png',
    tipo: 'Silver',
    costo: 0
  },
  {
    img: './assets/images/MMGold.png',
    tipo: 'Gold',
    costo: 5750
  },
  {
    img: './assets/images/MMPlatinum.png',
    tipo: 'Platinum',
    costo: 7950
  }
];

```

Figura 49 “arreglo de membresías” (Datos Obtenidos en el Software visual basic framework angular 12).

Después usaremos el tipo de botón sea largo

Y otros datos referentes ala membresía

Ya después de tener eso crear un arreglo donde traiga la clave referida

```

size: NzButtonSize = 'large';
private subscription: Subscription = new Subscription()

foliomembresia = '';
ciclo = 0;
numMembresia = 0;
listOfDRUMCLAVEREF: clavreferidoResponse[] = [];

```

Figura 50 “variables” (Datos Obtenidos en el Software visual basic framework angular 12).

Crearemos el constructor donde pondremos el servicio del validate para que pueda vincular el servicio ahora activaremos el router que este nos ayudara a ir a la ruta donde se encontrara el componente de miembros después usaremos el modale donde hace el servicio proveniente de zorro después crearemos el servicio de correo y los smp para poder mandar correos

```

constructor(
  private fb: FormBuilder,
  private route: ActivatedRoute,
  private router: Router,
  private miembroSvc: MiembroService,

  private modal: NzModalService,
  public correoSrv: CorreoService,
  public vienb: Miembrosemail,
  private clavferSvc: ClavereferidoService
) {

```

Figura 51 “constructor” (Datos Obtenidos en el Software visual basic framework angular 12).

Crearemos la variable de espera donde está en un valor false

Y otro valor donde se activa en el constructor

en membresía será igual al arreglo de la imagen y también inicializaremos el termino con valor falso

```

    {
      this.isAwaiting = false;
      this.isfirst = true;
      this.membresia = this.arrayimg[0].img;
      this.terminos = false;
    }

```

Figura 52 “inicializaciones en las variables” (Datos Obtenidos en el Software visual basic framework angular 12).

En la primera función muestra un modal externo donde se ingresará la vista del modal en el HTML de registro para que muestre un mensaje de bienvenida después es un botón para aceptar los términos y condiciones

Después el naavigate se manda al componente miembros donde se verá el registro de la tabla

Y después la función para cerrar el termino y condiciones

```
showModal(): void {  
  this.modalVisible = true;  
}  
  
terminosOk(): void {  
  this.terminos = true;  
  this.modalVisible = false;  
}  
  
miembroOk(): void {  
  this.miembroVisible = false;  
  this.router.navigate(['miembros']);  
}  
  
terminosCancel(): void {  
  this.terminos = false;  
  this.modalVisible = false;  
}
```

*Figura 53 “funciones del modal” (Datos Obtenidos en el Software visual
basic framework angular 12).*

Después se hace la función donde se muestra que se aceptaron los términos y condiciones
y si no pasa eso un mensaje donde tienes que aceptar los términos y condiciones

```

muestra(): void {
  if (!this.isfirst) {
    if (this.terminos) {
      this.modal.success({
        nzTitle: '¡Felicidades!',
        nzContent: '¡Haz aceptado los Términos y Condiciones!'
      });
    } else {
      this.modal.warning({
        nzTitle: '¡Espera!',
        nzContent: 'Todavía no aceptas los términos y condiciones'
      });
    }
  } else {
    this.isfirst = false;
  }
}

```

Figura 54 “función muestra” (*Datos Obtenidos en el Software visual basic framework angular 12*).

En la función onint se mandará que el valor de membresía sea igual ala membresía

El valor de radio será igual al tipo de dato

En esta parte se ingresarán los campos que se van a validar y si van a ser requeridos después iremos a buscar el servicio de clave referido para obtener los valores del nombre de la persona

```

ngOnInit(): void {

  this.membresia = this.arrayimg[0].img;
  this.DataRegistro.tipomembresia = this.radioValue;
  this.DataRegistro.password = '12345678';

  this.validateForm = this.fb.group({
    nombre: [null, [Validators.required]],
    apellidop: [null, [Validators.required]],
    apellidom: [null, [Validators.required]],
    telefono: [null, [Validators.required]],
    email: [null, [Validators.required]],
    estado: [null, [Validators.required]],
    profesion: [null, [Validators.required]],
    escolaridad: [null, [Validators.required]],
    especialidad: [null, [Validators.required]],
    cedula: [null, [Validators.required]],
    claverferido: [null, [Validators.required]],
    // termino: [null, [Validators.required]],
  });
  this.clavferSvc.getAll().subscribe((claverferido) => {
    console.log(claverferido);
    this.listOfDRUMCLAVeref = claverferido
  });
}

```

Figura 55 “función de arranque de la página web” (Datos Obtenidos en el Software visual basic framework angular 12).

El primer for es para validar que ningún campo este vacío y el otro para verificar que se ingresaron los datos

Después del primer if que si es verdadero valla a la función de muestra

If si ya se llenaron los campos y los términos ya fueron aceptados se hará el ingreso de datos

```

registrar(): void {
  this.terminos = true;
  for (const i in this.validateForm.controls) {
    this.validateForm.controls[i].markAsDirty();
    this.validateForm.controls[i].updateValueAndValidity();
  }
  for (const key in this.validateForm.controls) {
    this.validateForm.controls[key].markAsDirty();
  }
  if (!this.terminos) {
    this.muestra();
  }
  if (this.validateForm.valid && this.terminos) {

```

Figura 56 “función del registro” (Datos Obtenidos en el Software visual
basic framework angular 12).

Se harán que la fecha por default sea ingresada automáticamente después que se ingrese nueva fecha después pasamos los valores del radio botón a string y que el tipo de membresía sea igual al arreglo de la imagen

Después se ingresará un tiempo para poder actualizar la tabla con un tiempo de 4 milisegundos

Membresía será igual a 1 eso indica que se muestra que sea igual a 1

Después agregamos la función de la suscripción después ingresaremos un servicio de registrar donde fromvalue ósea el this data registro que es un objeto se usara en el observable para mandar la respuesta donde si el número de membresía sea igual a la respuesta y regresara a la función de envió con el tiempo de 10 milisegundos

```

    if (this.validateForm.valid && this.terminos) {

        this.DataRegistro.fechareg = new Date();
        this.DataRegistro.password = '12345678';
        this.ivisible = false;
        let i = parseInt(this.radioValue);
        this.DataRegistro.tipomembresia = this.arrayimg[i].tipo;
        setTimeout(() => {
            this.ivisible = true;
        }, 4);

        this.DataRegistro.membresia = 1;
        const formValue = this.DataRegistro;
        const that = this;
        this.subscription.add(
            this.miembroSvc.registrar(formValue).subscribe((res) => {
                if (res) {
                    setTimeout(function () {
                        that.numMembresia = res.membresia;

                        that.envio();
                    }, 1800);
                }
            })
        );
    }
}

```

Figura 57 “validación de la función de registro” (Datos Obtenidos en el Software visual basic framework angular 12).

Usaremos la función de abrir que esto es cuando se da abre el HTML

Después en la función de destroy después de agregar se manda un nuevo formato

```

open(): void {
  this.invisible = false;
  let i = parseInt(this.radioValue);
  this.membresia = this.arrayimg[i].img;
  setTimeout(() => {
    this.invisible = true;
  }, 4);
}

ngOnDestroy(): void {
  this.subscription.unsubscribe();
}

envio() {

```

Figura 58 “función de abrir” (Datos Obtenidos en el Software visual
basic framework angular 12).

En el la funcion de envio se mandara el mensaje de bienvenida y datos y despues usaremos el servicio de correo para poderlo mandarlo

```

envio() {
  this.dataCorreo.nombre = this.DataRegistro.nombre;
  this.dataCorreo.email = this.DataRegistro.email;
  this.dataCorreo.asunto = 'asunto';
  this.dataCorreo.mensaje = this.vienb.moymes(this.DataRegistro);
  this.correoSrv.sendMessage(this.dataCorreo).subscribe(() => {
    console.log("Formulario de contacto", "Mensaje enviado correctamente", 'success');
  });
}

```

Figura 59 “enviar mensaje” (Datos Obtenidos en el Software visual
basic framework angular 12).

También usaremos

Plantillas para los correos de bienvenida y el de miembros donde ira quien se registra y todo eso

Y su servicio

```
4 import { Injectable } from '@angular/core';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class CorreoService {
10
11   constructor(private http: HttpClient) { }
12
13   sendMessage(body: any) {
14     console.log('contenido', body);
15     return this.http.post(`${environment.Urlcorreo}/correo`, body);
16   }
17 }
```

Figura 60 “implementación de servicio de correo” (Datos Obtenidos en el Software visual basic framework angular 12).


```

<p class="font">Monitor de Miembros</p>
<nz-col [nzSpan]="12">
  | [nz-statistic [nzValue]="(numrag | number)!" [nzTitle]="'#miembros'<img alt="miembros icon" data-bbox="715 105 735 125"/>"</nz-statistic>
</nz-col>
<nz-table #fixedTable [nzData]="listOfData" [nzScroll]="{ x: '1550px', y: '240px' }">
  <thead>
    <tr class="font">
      <th nzLeft nzWidth="150px">Nombres</th>
      <th nzWidth="120px">Teléfono</th>
      <th nzWidth="200px">E-mail</th>
      <th nzWidth="120px">Estado</th>
      <th nzWidth="150px">Profesión</th>
      <th nzWidth="100px">Escolaridad</th>
      <th nzWidth="110px">Especialidad</th>
      <th>Cédula</th>
      <th>Clave Referido</th>
      <th nzWidth="100px">Membresía</th>
      <th>Tipo</th>
      <th colspan="2">Fecha de registro</th>
      <th nzRight nzWidth="60px">Acciones</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let data of fixedTable.data" class="font">
      <td nzLeft nzWidth="150px">{{ data.nombre }}
        <div>
          {{ data.apellido1 }}
          {{ data.apellido2 }}
        </div>
      <td nzWidth="120px">{{ data.telefono }}</td>
      <td nzWidth="200px">{{ data.email }}</td>
      <td nzWidth="120px">{{ data.estado }}</td>
      <td nzWidth="150px">{{ data.profesion }}</td>
      <td nzWidth="100px">{{ data.escolaridad }}</td>
      <td nzWidth="110px">{{ data.especialidad }}</td>
      <td>{{ data.cedula }}</td>
      <td>{{ data.clavereferido }}</td>
      <td nzWidth="100px">{{ data.membresia }}</td>
      <td>{{ data.tipomembresia }}</td>
      <td colspan="2">{{ data.fechareg }}</td>
      <td nzRight nzWidth="60px">
        <div>
          <i>nz-icon nzType="edit" nzTheme="outline" (click)="open(data)" style="font-size: 20px;"</i>
          <i>nz-icon nzType="mail" nzTheme="outline" (click)="opencorreo(data)" style="margin-left: 20px; font-size: 20px;"</i>
          <i>nz-icon nzType="delete" nzTheme="outline" style="margin-left: 20px; font-size: 20px;"</i>
        </div>
      </td>
    </tr>
  </tbody>
</nz-table>

```

Figura 62 “HTML del monitor de miembros”(Datos Obtenidos en el Software visual basic framework angular 12).

```

<nz-drawer [nzBodyStyle]="{ overflow: 'auto' }" [nzMaskClosable]="false" [nzWidth]="720"
[nzVisible]="visible" nzTitle="Datos del miembro" [nzFooter]="footerTpl" (nzOnClose)="close()"
<form nz-form *nzDrawerContent>
  <div nz-row [nzGutter]="8">
    <div nz-col nzSpan="10">
      <nz-form-item>
        <nz-form-label>Nombre(s)</nz-form-label>
        <nz-form-control>
          <input nz-input placeholder="Nombre" [(ngModel)]="dataDrawer.nombre" name="
        </nz-form-control>
      </nz-form-item>
    </div>

    <div nz-col nzSpan="10">
      <nz-form-item>
        <nz-form-label>Apellido paterno</nz-form-label>
        <nz-form-control>
          <input nz-input placeholder="paterno" [(ngModel)]="dataDrawer.apellidop" na
        </nz-form-control>
      </nz-form-item>
    </div>
  </div>

  <div nz-row>
    <div nz-col nzSpan="10">
      <nz-form-item>
        <nz-form-label>Apellido materno</nz-form-label>
        <nz-form-control>
          <input nz-input placeholder="materno" [(ngModel)]="dataDrawer.apellidom" n
        </nz-form-control>
      </nz-form-item>
    </div>
    <div nz-col nzSpan="10">
      <nz-form-item>
        <nz-form-label>Teléfono</nz-form-label>
        <nz-form-control>
          <input nz-input placeholder="telefono" [(ngModel)]="dataDrawer.telefono" na
        </nz-form-control>
      </nz-form-item>
    </div>
  </div>

```

Figura 63 “drawer de registro” (Datos Obtenidos en el Software visual basic framework angular 12).

Después le crearemos los botones con las funciones de editar y cerrar

```

<ng-template #footerTpl>
  <div style="float: right">
    <button nz-button style="margin-right: 8px;" (click)="close()">Cancelar</button>
    <button nz-button nzType="primary" (click)="editar()">Guardar cambios</button>
  </div>
</ng-template>

```

Figura 64 “botón de datos” (Datos Obtenidos en el Software visual basic framework angular 12).

Después debemos crear un drawer de correo después ingresaremos el contenido donde mandamos a llamar los valores de datadrawer que es un arreglo y las opciones de los correos después se enviarán los sistemas de contactform donde buscaremos el arreglo el asunto y mensaje

```

<nz-drawer [nzBodyStyle]="{ overflow: 'auto' }" [nzMaskClosable]="false" [nzWidth]="720"
[nzVisible]="visiblecorreo" nzTitle="Correo" [nzFooter]="footercorreo"
(nzOnClose)="closecorreo()">
  <ng-container *nzDrawerContent>
    <div>
      <p>Nombre: {{ dataDrawer.nombre }} {{ dataDrawer.apellido }}
        {{ dataDrawer.apellido }}</p>
      <p>Correo: {{ dataDrawer.email }}</p>
    </div>

    <div nz-col nzSpan="20">
      <label>Asunto</label>
      <input type="text" class="form-control" [(ngModel)]="asunto" name="asunto" />
    </div>

    <div nz-col nzSpan="20">
      <label>Mensaje</label>
      <textarea type="text" class="form-control" [(ngModel)]="mensaje" name="mensaje">

      </textarea>
    </div>

    <br><br>
    <nz-radio-group [(ngModel)]="mailType" name="radio">
      <label nz-radio nzValue="0">Bienvenida</label>
      <label nz-radio nzValue="1">Moyco</label>
      <label nz-radio nzValue="2">Webinar</label>
    </nz-radio-group>
  </ng-container>

  <ng-template #footercorreo>
    <div style="float: right">
      <button nz-button style="margin-right: 8px;" (click)="closecorreo()">Cancelar
      </button>
      <button nz-button nzType="primary" (click)="contactForm(dataDrawer, asunto, mensaje)"
      >Enviar</button>
    </div>
  </ng-template>

```

Figura 65 “drawer del correo” (Datos Obtenidos en el Software visual basic framework angular 12).

Ingresaremos las importaciones de los módulos y los servicios de correos y las plantillas de correos

```
import { Component, OnInit } from '@angular/core';
import { Miembro, MiembroResponse } from 'src/app/models/miembro';
import { Bienvenidamail } from 'src/app/correos/bienvenida';
import { CorreoService } from 'src/app/services/correo.service';
import { MiembroService } from 'src/app/services/miembro.service';
import { Miembrosemail } from 'src/app/correos/miembross';
import { Moycomail } from 'src/app/correos/moyco';
```

Figura 66 “librerías del componente de miembros ts” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta parte crearemos la interface de los correos después ingresaremos los datos de email mensaje y el nombre

```
interface ItemData {
  asunto: string;
  email: string;
  mensaje: string;
  nombre: string;
}

@Component({
  selector: 'app-miembros',
  templateUrl: './miembros.component.html',
  styleUrls: ['./miembros.component.css']
})
export class MiembrosComponent implements OnInit {
```

Figura 67 “creacion del modulo” (Datos Obtenidos en el Software visual basic framework angular 12).

En el primer arreglo de datos el lisofdata es donde se esta guardano los arreglos

Los visible son de tipo booleano ya que nos mostrara si se enciende o se apaga la funcion o

los drawers

Después los arreglos de edición y del correo el asunto y mensaje de los datos y el tipo de mail donde será un arreglo de email número de reg este es en todos los elementos de la aplicación funciona para ver que miembro se dio de alta

Y el constructor se crea el servicio e correo y las plantillas

```
listOfData: MiembroResponse[] = [];  
visible: boolean;  
visiblecorreo: boolean;  
dataDrawer: MiembroResponse;  
dataCorreo: ItemData;  
asunto= '';  
mensaje: any;  
mailType= '0';  
router: any;  
numrag:number;  
  
constructor (  
  private miembroSvr: MiembroService,  
  public correoSrv: CorreoService,  
  private bienm: BienvenidaMail,  
  private moycm: Moycomail  
) {  
  this.visible = false;
```

Figura 68 “arreglo y constructor de miembros” (Datos Obtenidos en el Software visual basic framework angular 12).

Inicializaremos las vistas con un valor falso

Usaremos un dataDrawer que un arreglo de datos donde se usará para la edición de miembros

Después el arreglo de correo

```

71
this.visible= false;
this.visiblecorreo= false;
this.dataDrawer = {
  id: 0,
  message: '',
  nombre: '',
  apellidop: '',
  apellidom: '',
  telefono: '',
  email: '',
  estado: '',
  profesion: '',
  escolaridad: '',
  especialidad: '',
  cedula: '',
  claverreferido: '',
  password: '',
  membresia: 0,
  tipomembresia: '',
  role: 'miembro',
  fechareg: new Date(),
}
this.dataCorreo = {
  asunto: '',
  email: '',
  mensaje: '',
  nombre: '',
}
this.numrag=0;
}

```

Figura 69 “variables y el arreglo de correo” (Datos Obtenidos en el Software visual basic framework angular 12).

La inicialización de datos en el número de registro

En la parte del onint se mostrará los datos de obtención del miembro y se plasmaran en la tabla de datos y del número de registros sea igual al dato de la tabla y a su longitud del objeto

```

ngOnInit(): void {
  console.log('ejecutando');
  this.miembroSvr.getAll().subscribe((miembros)=>{
    console.log('miembros', miembros);
    this.listOfData = miembros
    this.numrag=this.listOfData.length;
  });
}

```

Figura 70 “función oninit” (Datos Obtenidos en el Software visual basic framework angular 12).

Al momento de hacer la función edit. se mandará a llamar la función abre y buscará los datos de la tabla y los datos guardados en el arreglo para mostrarlos

```

open(data: any): void {
  this.visible = true;
  this.dataDrawer=data;
  console.log('datos que se tienen que ver', this.dataDrawer)
}

```

Figura 71 “función open” (Datos Obtenidos en el Software visual basic framework angular 12).

Aquí se usarán las funciones del correo desde abrir el correo que llevara los datos del drawer al de los datos de asunto y datos de mensaje el otro es el close del drawer de edición

El botón del drawer con las opciones de mandar el correo y la implementación del servicio

```

opencorreo(data: any): void {
  this.visiblecorreo = true;
  this.dataDrawer=data;
  this.asunto='';
  this.mensaje='';
  console.log('correo', this.dataDrawer)
}

close(): void {
  this.visible = false;
}

closecorreo(): void {
  this.visiblecorreo = false;
}

contactForm(miembro: MiembroResponse, asunto: string, mensaje: any) {

  let idFolio = parseInt(this.mailType);
  switch (idFolio) {
    case 0:
      this.dataCorreo.mensaje=this.bienm.moymes(miembro);
      break;
    case 1:
      this.dataCorreo.mensaje=this.moym.moymes();
      break;
    default:
      this.dataCorreo.mensaje=mensaje;
  }
  // contactForm(form)
  this.dataCorreo.nombre=miembro.nombre;
  this.dataCorreo.email=miembro.email;
  this.dataCorreo.asunto=asunto;
  this.correoSrv.sendMessage(this.dataCorreo).subscribe(() => {
    console.log("Formulario de contacto", "Mensaje enviado correctamente", 'success');
  } );
  this.visiblecorreo = false;
}

```

Figura 72 “configuraciones de la funcion de envio de mensajes de correo en miembros” *(Datos Obtenidos en el Software visual basic framework angular 12).*

En esta parte de edición usaremos el servicio de miembros y la función servicio el rol será miembros

```

editar(){
  console.log('edit',this.dataDrawer);
  this.dataDrawer.fechareg = new Date();
  this.dataDrawer.role= 'miembro'
  this.miembroSvr.Editar(this.dataDrawer).subscribe((resp) => {
    console.log("Datos",resp);
  }));
  this.visible = false;
}
}

```

Figura 73 “función de editar” (Datos Obtenidos en el Software visual basic framework angular 12).

En el componente de usuario se creará casi lo mismo, pero en HTML lo que cambia es lo siguiente que tiene un botón el cual se usara para el drawe de agregar más personas las cuales

Y eso no se hizo en miembros porque miembros viene registrado de la página web femae-pas

```

<nz-row>
  <div>
    <p style="margin-left: 450px; font-size: large;">Monitor de usuarios</p>
  </div>
  <nz-col [nzSpan]="12">
    <nz-statistic [nzValue]="(numrog | number)!" [nzTitle]="'#usuarios'"></nz-statistic>
  </nz-col>
  <div style="margin-left: 350px;">
    <button nz-button nzType="primary" [nzSize]="size" nzShape="circle" (click)="open()"><i nz-i
      nzType="plus"></i></button>
  </div>
</nz-row>

```

Figura 74 “HTML monitor usuario” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta parte se creará la tabla con los datos siguientes y también se crea un drawer de agregación para el usuario y al igual que los dos drawers de correo y de editar ya antes vistos

En el componente miembros

```

<div style="margin-left: 350px;">
  <button nz-button nzType="primary" [nzSize]="size" nzShape="circle" (click)="open()"><i nz-ic
    nzType="plus"></i></button>
</div>
</nz-row>
</nz-table>
<nz-table #basicTable [nzData]="listOfData" *ngIf="vista" [nzScroll]="{ x: '450px', y: '340px' }">
  <thead>
    <tr>
      <th>Nombre</th>
      <th>correo</th>
      <th>departamento</th>
      <th>rol</th>
      <th>cuenta</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let data of basicTable.data ; index as i" >
      <td>{{ data.nombre }} {{ data.ap_paterno }} {{ data.ap_materno }}</td>
      <td>{{ data.correo }}</td>
      <td>{{ data.departamento }}</td>
      <td>{{ data.rol }}</td>
      <td>{{ data.cuenta }}</td>
      <td>
        <i nz-icon nzType="edit" nzTheme="outline" (click)="clickedit(data)" style="font-size:
        <i nz-icon nzType="mail" nzTheme="outline" (click)="opencorreo (data)" style="margin-1
        <i nz-icon nzType="delete" nzTheme="outline" style="margin-left: 2vh;font-size: 20px;
      </td>
    </tr>
  </tbody>
</nz-table>
<nz-drawer [nzBodyStyle]="{ overflow: 'auto' }" [nzMaskClosable]="false" [nzWidth]="720" [nzVisible
  nzTitle="Agregar Usuario" [nzFooter]="afooterIpl" (nzOnClose)="close()">
  <form nz-form *nzDrawerContent [formGroup]="validateForm">
    <div nz-row [nzGutter]="8">
      <div nz-col nzSpan="8">

```

Figura 75 “monitor de usuario” (Datos Obtenidos en el Software visual
basic framework angular 12).

Ahora iremos a los módulos y al servicio

Se crearán los módulos de usuario, departamento y rol

```
✓ export interface Usuario {  
  nombre: string;  
  ap_paterno: string;  
  ap_materno: string;  
  correo: string;  
  id_departamento: number;  
  id_rol: number;  
  cuenta: string;  
  password: string;  
  id_status: number;  
  fecha_reg: string;  
}  
  
✓ export interface UsuarioResponse extends Usuario {  
  id: number;  
  message: string;  
  nombre: string;  
  ap_paterno: string;  
  ap_materno: string;  
  correo: string;  
  id_departamento: number;  
  departamento: string;  
  rol: string;  
  id_rol: number;  
  cuenta: string;  
  password: string;  
  id_status: number;  
  fecha_reg: string;  
}
```

Figura 76 “modulo de usuario” (Datos Obtenidos en el Software visual
basic framework angular 12).

Después crearemos sus módulos de departamento y de rol

```

export interface Departamneto {
  departamento: string;
  id_status: number;
  fecha_reg: string;
}

export interface DepartamentoResponse extends Departamneto {
  id: number;
  message: string;
  departamento: string;
  id_status: number;
  fecha_reg: string;
}

```

Figura 77 “modulo de departamento” (Datos Obtenidos en el Software visual basic framework angular 12).

```

export interface Rol {
  rol: string;
  id_status: number;
  fecha_reg: string;
}

export interface RolResponse extends Rol {
  id: number;
  message: string;
  rol: string;
  id_status: number;
  fecha_reg: string;
}

```

Figura 78 “modulo de roles” (Datos Obtenidos en el Software visual basic framework angular 12).

Se crearán sus servicios donde tendrán la única función del servicio del departamento y el rol será el geta ll para poder mostrarlo en el drawer al momento de la captura

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';
import { environment } from 'src/environments/environment';
import { DepartamentoResponse, Departamneto } from '../models/departamento';

@Injectable({
  providedIn: 'root'
})
export class DepartamentoService {

  constructor(
    private http: HttpClient
  ) { }

  getAll(): Observable<DepartamentoResponse[]> {
    return this.http.get<DepartamentoResponse[]>(`${environment.Urlbase}/depto/all`);
  }
}
```

Figura 79 “servicio de departamento” (*Datos Obtenidos en el Software visual basic framework angular 12*).

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';
import { environment } from 'src/environments/environment';
import { Rol, RolResponse } from '../models/rol';

@Injectable({
  providedIn: 'root'
})
export class RolService {

  constructor(
    private http: HttpClient
  ) { }

  getAll(): Observable<RolResponse[]> {
    return this.http.get<RolResponse[]>(`${environment.Urlbase}/rol/all`);
  }
}

```

Figura 80 “servicio de roles” (Datos Obtenidos en el Software visual
basic framework angular 12).

Y después aremos el servicio de usuario las funciones del servicio serán parecidas ya que se usarán las mismas de editar y registrar y obtener los datos solo cambia la url del ambiente de datos

```

1 import { Usuario, UsuarioResponse } from '../models/usuario';
2
3 import { HttpClient } from '@angular/common/http';
4 import { Injectable } from '@angular/core';
5 import { Observable } from 'rxjs';
6 import { environment } from 'src/environments/environment';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class UsuarioService {
12
13   constructor(
14     private http: HttpClient
15   ) { }
16
17   getAll(): Observable<UsuarioResponse[]> {
18     return this.http.get<UsuarioResponse[]>(`${environment.Urlbase}/usu/all`);
19   }
20
21   registrar(regData: Usuario): Observable<Usuario> {
22     console.log('agregar', regData);
23     return this.http.post<Usuario>(`${environment.Urlbase}/usu`, regData);
24   }
25
26   Editar(editData: UsuarioResponse): Observable<UsuarioResponse | void> {
27     console.log('edita', editData);
28     return this.http.patch<UsuarioResponse>(`${environment.Urlbase}/usu/${editData.id}`, editData);
29   }
30 }

```

Figura 81 “servicio de usuarios” (Datos Obtenidos en el Software visual basic framework angular 12).

En el componente ts importamos las librerías a usar

Creamos la interfaz de correo

```

1  import * as moment from 'moment';
2
3  import { Component, OnInit } from '@angular/core';
4  import { FormBuilder, FormGroup, Validators } from '@angular/forms';
5  import { Usuario, UsuarioResponse } from 'src/app/models/usuario';
6
7  import { Bienvenidamail } from 'src/app/correos/bienvenida';
8  import { CorreoService } from 'src/app/services/correo.service';
9  import { DepartamentoResponse } from 'src/app/models/departamento';
10 import { DepartamentoService } from 'src/app/services/departamento.service';
11 import { Moycomail } from 'src/app/correos/moyco';
12 import { NzButtonSize } from 'ng-zorro-antd/button';
13 import { RolResponse } from 'src/app/models/rol';
14 import { RolService } from 'src/app/services/rol.service';
15 import { UsuarioService } from 'src/app/services/usuario.service';
16 import { Usuariomail } from 'src/app/correos/usuario';
17 import { Webinarmail } from 'src/app/correos/webinar';
18 import { id_ID } from 'ng-zorro-antd/i18n';
19
20 interface ItemData {
21   asunto: string;
22   email: string;
23   mensaje: string;
24   nombre: string;
25 }

```

Figura 82 “librerías para el componente ts” (Datos Obtenidos en el Software visual basic framework angular 12).

Crearemos las variables que nos permitan ir a visualizar los drawer y ver el contenido de la tabla

Crearemos una variable para que nos de la fecha del día que se hace el registro

La vista que se inicie en true el tamaño del botón

```

export class UsuariosComponent implements OnInit {
  hoy = new Date();
  vista = true;
  size: NzButtonSize = 'large';
  validateForm!: FormGroup;
  visible = false;
  evisible=false;
  numrog:number;
  visiblecorreo: boolean;
  id_departamento = '';
  id_rol = '';

```

Figura 83 “variables de usuarios” (*Datos Obtenidos en el Software visual basic framework angular 12*).

la variable del número de registro y la validación de campos

Y los valores del id departamento y rol que sean igual al tipo string

Después crearemos las funciones de correo

```

dataCorreo: ItemData;
asunto= '';
mensaje: any;
mailType= '0';

```

Figura 84 “variables de envío de correos” (*Datos Obtenidos en el Software visual basic framework angular 12*).

Se presentarán el arreglo de edición y de agregado la diferencia de las interfaces es que una ira primero a buscar el id y con eso desplegar el contenido de los campos de id

Y la otra solo agregara los campos de id

Y después declararemos las variables como arreglo en las interfaces

```

usuarioedit: UsuarioResponse = {
  id:0,
  nombre: '',
  message:'',
  ap_paterno: '',
  ap_materno: '',
  correo: '',
  id_departamento: 1,
  departamento:'',
  id_rol: 1,
  rol:'',
  cuenta: '',
  password: '',
  id_status: 1,
  fecha_reg: ''
}
usuarionew: Usuario = {
  nombre: '',
  ap_paterno: '',
  ap_materno: '',
  correo: '',
  id_departamento: 1,
  id_rol: 1,
  cuenta: '',
  password: '',
  id_status: 1,
  fecha_reg: ''
}
listOfData: UsuarioResponse[] = [];
listOfDataDepto: DepartamentoResponse[] = [];
listOfDataRol: RolResponse[] = [];

```

Figura 85 “arreglos de edición” (Datos Obtenidos en el Software visual basic framework angular 12).

Pues en el constructor se volverá a poner los servicios del correo el servicio de usuario y pues se inicializará la interfaz de la data correo y pues se inicializará el contador de usuarios dados de alta

```
constructor(  
    private fb: FormBuilder,  
    private usuarioSvr: UsuarioService,  
    private deptoSvr: DepartamentoService,  
    private rolSvr: RolService,  
    public Corresvr: CorreoService,  
    private bienm: Bienvenidamail,  
    private moy: Moycomail,  
    private usu :Usuariomail,  
    private web :Webinarmail  
) {  
    this.visiblecorreo= false;  
    this.dataCorreo = {  
        asunto: '',  
        email: '',  
        mensaje: '',  
        nombre: '',  
    }  
    this.numrog=0;  
}
```

Figura 86 variables en constructor

OnInit buscará los datos del backend y los mostrará en la tabla con la función `getall` así igual con rol y departamento igual vendrá la verificación de los campos al igual que usaremos un temporizador en una función de actualización de los campos de departamento y rol

```
ngOnInit(): void {
  let tha: this;
  this.usuarioSvr.getAll().subscribe((usuarios) => {
    console.log(usuarios);
    this.listOfData = usuarios;
    this.numrog = this.listOfData.length;
  });
  this.deptoSvr.getAll().subscribe((departamento) => {
    console.log(departamento);
    this.listOfDataDepto = departamento
  });
  this.rolSvr.getAll().subscribe((rol) => {
    console.log(rol);
    this.listOfDataRol = rol
  });
  setTimeout(() => {
    this.actualiza();
  }, 500);
;

  this.validateForm = this.fb.group({
    nombre: [null, [Validators.required]],
    ap_paterno: [null, [Validators.required]],
    ap_materno: [null, [Validators.required]],
    correo: [null, [Validators.required]],
    departamento: [null, [Validators.required]],
    rol: [null, [Validators.required]],
    cuenta: [null, [Validators.required]],
    password: [null, [Validators.required]],
  });
}
```

Figura 87 “creación de funciones en el oninit” (Datos Obtenidos en el Software visual basic framework angular 12).

Se creará la función de agregar en la que pondremos la vista de la tabla en false para apagar

la vista y el estatus de id new la fecha del día de hoy mandándola por mes día y año después pondremos la función de registrar donde se mandaran los datos de usuario new y recibe la respuesta después actualizamos la vista del usuario esto para evitar ir a actualizar la página manualmente

Después se hará la función de editar donde es prácticamente lo mismo de edit.

```
Agregar() {  
  for (const i in this.validateForm.controls) {  
    this.validateForm.controls[i].markAsDirty();  
    this.validateForm.controls[i].updateValueAndValidity();  
  }  
  if (this.validateForm.valid){  
    this.vista=false;  
    this.usuarionew.id_status = 1;  
  
    this.usuarionew.fecha_reg = moment(this.hoy).format('MM/DD/YYYY hh:mm:ss');  
    // const formValue = this.usuarionew;  
    this.usuarioSvr.registrar(this.usuarionew).subscribe((resp) => {  
      console.log("Datos",resp);  
      setTimeout(() => {  
        this.numrog=this.listOfData.length;  
        this.vista=true;  
      }, 200);  
    });  
    this.visible=false;  
  }  
}  
Editar(){  
  console.log('edit',this.usuarioedit);  
  this.usuarioedit.id_status = 1;  
  this.usuarioedit.fecha_reg = moment(this.hoy).format('MM/DD/YYYY hh:mm:ss');  
  this.usuarioSvr.Editar(this.usuarioedit).subscribe((resp) => {  
    console.log("Datos",resp);  
  });  
  this.evisible=false  
}
```

Figura 88 “función de agregar y editar” (*Datos Obtenidos en el Software visual basic framework angular 12*).

Hacemos la función para actualizar el tipo de rol y el departamento

```

actualiza() {
  this.listOfData.forEach(element => {
    element.departamento = this.deptoid(element.id_departamento);
    element.rol = this.rolid(element.id_rol);
  });
  console.log(this.listOfData);
}

```

Figura 89 “función de actualizar” (Datos Obtenidos en el Software visual basic framework angular 12).

Después se usarán las funciones para el traspaso de valores id a string buscan el departamento dentro del módulo de departamento al igual que con rol

Después aremos las funciones de abrir y cerrar clickedit se mandarán los datos del edit. y habilitar el drawer de edición

```

deptoid(id: number): string {
  let dato: string;
  let oss:any;
  oss = this.listOfDataDepdo.find(function (departamento) {
    return departamento.id == id;
  });
  dato = oss.departamento;
  return dato;
}

rolid(id: number): string {
  let tracki: string;
  let noml:any;
  noml = this.listOfDataRol.find(function (rol) {
    return rol.id == id;
  });
  tracki = noml.rol;
  return tracki;
}

open(): void {
  this.visible = true;
}

close(): void {
  this.visible = false;
  this.evisible = false;
}

clickedit(data:any):void{
  console.log("datos",data);
  this.usuariocedit=data;
  this.evisible=true;
}

```

Figura 90 “funciones de actualización” (Datos Obtenidos en el Software visual basic framework angular 12).

En la siguiente captura se usarán para capturar el correo

```
✓ opencorreo(data: any): void {  
  this.visiblecorreo = true;  
  this.usuarioedit=data;  
  this.asunto='';  
  this.mensaje='';  
  console.log('correo', this.usuarioedit)  
}  
  
✓ closecorreo(): void {  
  this.visiblecorreo = false;  
}  
  
✓ contactForm() {  
  let idFolio = parseInt(this.mailType);  
  switch (idFolio) {  
    ✓ case 0:  
      this.dataCorreo.mensaje=this.usu.moymes(this.usuarioedit);  
      break;  
    ✓ case 1:  
      this.dataCorreo.mensaje=this.moym.moymes();  
      break;  
    ✓ case 2:  
      this.dataCorreo.mensaje=this.web.moymes();  
      break;  
    ✓ default:  
      this.dataCorreo.mensaje=this.mensaje;  
  }  
  // contactForm(form)  
  this.dataCorreo.nombre=this.usuarioedit.nombre;  
  this.dataCorreo.email=this.usuarioedit.correo;  
  this.dataCorreo.asunto=this.asunto;  
  ✓ this.Corresvr.sendMessage(this.dataCorreo).subscribe(() => {  
    console.log("Formulario de contacto", "Mensaje enviado correctamente", 'success');  
  } );  
  this.visiblecorreo = false;  
}  
}
```

Figura 91 “funciones del envío de correos” (Datos Obtenidos en el Software visual basic framework angular 12).

En la tabla de patrocinio se agregarán que en los drawer tablas de contacto para que tenga uno o más contactos

```

<button [nzSize]=" size"
  (click)="nuevocon(patrocinioeditado.id)"
  nz-button
  nzType="primary"
  nzShape="round">
  +
</button>
</div>
<div>
  <a>contactos</a>
</div>
</div>
<div [nzGutter]="8"
  nz-row>
    <nz-table #conTable
      *ngIf="ESvista"
      [nzData]="datacontacto"
      [nzScroll]="{ x: '550px', y: '440px' }">
        <thead>
          <tr>
            <th>Nombre</th>
            <th>Telefono</th>
            <th>Puesto</th>
            <th>Correo</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let data of conTable.data ; index as i">
            <td>{{ data.nombre }}</td>
            <td>{{ data.telefono }}</td>
            <td>{{ data.puesto }}</td>
            <td>{{ data.correo }}</td>
            <td nzRight>
              <nz-divider nzType="vertical"></nz-divider>
              <i [nzTheme]=" 'twotone' "
                [nzTwotoneColor]=" '#bdd01de' "
                (click)="conted(data,i)"
                nz-icon
                nzType="edit"
                nzTheme="outline"
                style="font-size: 20px;"></i>
            </td>
          </tr>
        </tbody>
      </nz-table>
    </div>
  </div>

```

Figura 92 “HTML patrocinadores” (Datos Obtenidos en el Software visual basic framework angular 12).

```

<div>
  <button [nzSize]=" size"
    (click)="nuevodommed()"
    nz-button
    nzType="primary"
    nzShape="round">
    +
  </button>
</div>
<div>
  <a>Domicilio</a>
</div>
</div>

<div [nzGutter]="8"
  nz-row>
    <nz-table #dTable
      *ngIf="domisvista"
      [nzData]="domidom"
      [nzScroll]="{ x: '1110px', y: '250px' }">
        <thead>
          <tr>
            <th>calle</th>
            <th>colonia</th>
            <th>numext</th>
            <th>numint</th>
            <th>municipio</th>
            <th>estado</th>
            <th>codigopostal</th>
            <th>pais</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let data of dTable.data ; index as a">
            <td>{{ data.calle }}</td>
            <td>{{ data.colonia }}</td>
            <td>{{ data.numex }}</td>
            <td>{{ data.numint }}</td>
            <td>
              {{ data.municipio }}
            </td>
            <td>{{ data.estado }}</td>
            <td>
              {{ data.codigopostal }}
            </td>
            <td>
              {{ data.Action }}
            </td>
          </tr>
        </tbody>
      </nz-table>
    </div>
  </div>

```

Figura 93 “domicilio html de patrocinadores” (Datos Obtenidos en el Software visual basic framework angular 12).

Crearemos los módulos necesarios como el de patrocinio y el modulo

Crearemos la clase de el arreglo de contactos y domicilio ingresándoles su

```

export class ContactoNew {
  id:number;
  id_patrocinio:number;
  nombre: string;
  puesto:string;
  telefono:string;
  correo:string;
  id_status: number;
  fecha_reg: Date;

  constructor(){
    this.id=0;
    this.id_patrocinio=0;
    this.nombre= '';
    this.puesto='';
    this.telefono='';
    this.correo='';
    this.id_status=1;
    this.fecha_reg= new Date;
  }
}

```

Figura 94 “modulo de las variables de lo que tiene contactos”
 (Datos Obtenidos en el Software visual basic framework angular 12).

```

export class DomicilioNew {
  id:number;
  id_patrocinio:number;
  calle:string;
  colonia:string;
  numex:string;
  numint:string;
  estado:string;
  municipio:string;
  codigopostal:string;
  pais:string;
  id_status:number;
  fecha_reg: Date;

  constructor(){
    this.id=0;
    this.id_patrocinio=0;
    this.calle= '';
    this.colonia='';
    this.numex='';
    this.numint='';
    this.municipio='';
    this.estado='';
    this.codigopostal='';
    this.pais='';
    this.id_status=1;
    this.fecha_reg= new Date;
  }
}

```

Figura 95 “modulo de contactos” (Datos Obtenidos en el Software visual basic framework angular 12).

Después de crear contacto y domicilio crearemos un objeto donde se guardarán contacto y domicilio

```

import { ContactoNew } from './contactonew';
import { DomicilioNew } from './domicilionew';

export interface PatrocinioNew{
  razon_social: string;
  id_giro:number;
  rfc:string
  id_status:number;
  lcontacto:ContactoNew[];
  ldomicilio:DomicilioNew[];
  fecha_reg: Date;
}

export interface PatrocinioNewResponseI extends PatrocinioNew{
  id:number;
  messege:string;
  razon_social: string;
  id_giro:number;
  giro :string;
  id_status:number;
  lcontacto:ContactoNew[];
  ldomicilio:DomicilioNew[];
  fecha_reg: Date;
}

```

Figura 96 “ts de patrocinadores (Datos Obtenidos en el Software visual basic framework angular 12).

En ts se importarán las librerías necesarias al igual que meter la interface para el correo que se necesita

```

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { PatrocinioNew, PatrocinioNewResponseI, } from 'src/app/models/patrocinio';
import { Bienvenidamail } from 'src/app/correos/bienvenida';
import { ContactoNew } from 'src/app/models/contactonew';
import { ContactoResponse, } from 'src/app/models/contacto';
import { ContactoService } from 'src/app/services/contacto.service';
import { CorreoService } from 'src/app/services/correo.service';
import { DomicilioNew } from 'src/app/models/domicilionew';
import { DomicilioResponse } from 'src/app/models/domicilio';
import { DomicilioService } from 'src/app/services/domicilio.service';
import { FromEventTarget } from 'rxjs/internal/observable/fromEvent';
import { Giro } from 'src/app/models/giro';
import { GiroResponse } from 'src/app/models/giro';
import { GiroService } from 'src/app/services/giro.service';
import { Moycomail } from 'src/app/correos/moyco';
import { NzButtonSize } from 'ng-zorro-antd/button';
import { PatrocinioService } from 'src/app/services/patrocinio.service';
import { Usuariomail } from 'src/app/correos/usuario';
import { Webinarmail } from 'src/app/correos/webinar';
import { contacto } from 'src/app/models/contacto';

interface ItemData {
  asunto: string;
  email: string;
  mensaje: string;
  nombre: string;
}

@Component({
  selector: 'app-patrocinio',
  templateUrl: './patrocinio.component.html',
  styleUrls: ['./patrocinio.component.css']
})
export class PatrocinioComponent implements OnInit {

```

Figura 97 “librerías interfaces de correo” (Datos Obtenidos en el Software visual basic framework angular 12).

Aquí declararemos las variables que se usaran al igual que las clases de arreglo que usaremos y el tipo de dato usado para el arreglo la validaciones

```

export class PatrocinioComponent implements OnInit {
  vista = true;
  ESvista = false;
  contanew = true;
  domineew=true;
  domisvista= false;
  size: NzButtonSize = 'default';
  sizes:NzButtonSize='small';
  asunto= '';
  mensaje: any;
  mailType= '0';
  Pvisible = false;
  cvisible = false;
  covisible=false;
  dvisible= false;
  edvisible =false;
  ecvisible=false;
  edovisible=false;
  iconac:number=0;
  idomi:number=0;
  visiblecorreo=false;
  dataCorreo: ItemData;
  edcvalidatefrom!:FormGroup;
  dovalidateForm!:FormGroup;
  validateForm!: FormGroup;
  cvalidateForm!: FormGroup;
  edvalidateFrom!:FormGroup;
  edcvalidateFrom!:FormGroup;
  dvalidateFrom!:FormGroup;
  domidom:DomicilioNew=[];
  datacontacto:ContactoNew=[];
  aggdomi:DomicilioNew;
  aggcontacto: ContactoNew;
  edomidom:DomicilioNew=[];
  edatacontacto:ContactoNew=[];
  editadomi:DomicilioNew;
  editacontacto: ContactoNew;

```

Figura 98 “variable de patrocinadores (*Datos Obtenidos en el Software visual basic framework angular 12*).

En esta parte se crean los objetos realizados como el de edición y el de agregado

```

editadomi:DomicilioNew;
editacontacto:ContactoNew;
(property) PatrocinioComponent.patrocinioagg: PatrocinioNew
patrocinioagg :PatrocinioNew = {
  razon_social: '',
  id_giro: 1,
  rfc:'',
  id_status:1,
  lcontacto:[],
  ldomicilio:[],
  fecha_reg:new Date(),
}
patrocinioeditado : PatrocinioNewResponseI = {
  id:0,
  razon_social: '',
  id_giro: 1,
  giro:'',
  messege:'',
  rfc:'',
  id_status:1,
  lcontacto:[],
  ldomicilio:[],
  fecha_reg:new Date(),
}

listOfmain: PatrocinioNewResponseI [] = [];
listOffrom:GiroResponse[]=[];

```

Figura 99 “creación de arreglos del patrocinadores” (Datos Obtenidos en el Software visual basic framework angular 12).

En el constructor usamos datos de los servicios que se usara después a las funciones a las variables ya creadas se inicializara como es agregar contacto edita contacto será de un nuevo contacto y la otras de un nuevo domicilio y lo demás es el correo y el número de registro

```

constructor(
  private fb: FormBuilder,
  private Patsvr:PatrocinioService,
  private dome:DomicilioService,
  private gir:GiroService,
  private cont :ContactoService,
  private bienm:Bienvenidamail,
  private moym: Moycomail,
  private usu :Usuariomail,
  private web :Webinarmail,
  public Corresvr: CorreoService
){
  this.aggcontacto=new ContactoNew();
  this.editacontacto=new ContactoNew();
  this.aggdomi=new DomicilioNew();
  this.editadomi=new DomicilioNew();
  this.visiblecorreo= false;
  this.dataCorreo = {
    asunto: '',
    email: '',
    mensaje: '',
    nombre: '',
  }
  this.numreg=0;
}

```

Figura 100 “constructor” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta ocasión en el onint un servicio para giro que se desplegara ala interface de la lista de ellos y el get all de patrocinio donde está su servicio y las validaciones de domicilio y contacto en los diferentes drawers

```

ngOnInit(): void {
  this.gir.getAll().subscribe((Giro) => {
    console.log(Giro);
    this.listOffrom = Giro
  });
  this.Patsvr.getAll().subscribe((patrocinio) => {
    console.log(patrocinio);
    this.listOfmain = patrocinio;
    this.numreg=this.listOfmain.length;
  });

  this.validateForm = this.fb.group({

    razon: [null, [Validators.required]],
    giro: [null, [Validators.required]],
    rfc:[null,[Validators.required]],
  });
}

```

Figura 101 “oninit donde se obtendran los datos de la base de datos y se mostraran en la tabla” (Datos Obtenidos en el Software visual basic framework angular 12).

La primera función después de hacer la validación de campos vacíos ira a introducir el valor de actualización de la vista de la página y su número de registros al igual que va a poner que el valor l domicilio y lcontacto son iguales a domidom y a data contacto que son las interfaces del arreglo

En la primera función del open será que todos los arreglos vallan vacíos para que sean nuevos después el botón del drawer agregar contacto sea vacío para poder usar que sea nuevo contrato después de mandar click

Y contad es el botón para poder abrir el drawer de edición de contacto y hace es lo mismo con domicilio

```

    }
    ✓ Agregar(){
    ✓   for (const i in this.validateForm.controls) {
    ✓       this.validateForm.controls[i].markAsDirty();
    ✓       this.validateForm.controls[i].updateValueAndValidity();
    ✓   }
    ✓   this.vista=false;
    ✓   this.patrocinioagg.Idomicilio=this.domidom;
    ✓   this.patrocinioagg.lcontacto=this.datacontacto;
    ✓   this.Patsvr.registrar(this.patrocinioagg).subscribe((resp) => {
    ✓       console.log("Datos",resp);
    ✓       this.numreg=this.listOfmain.length;
    ✓       this.vista=true;
    ✓   });
    ✓   this.Pvisible=false;
    ✓ };

    ✓ open(){
    ✓   this.Pvisible=true;
    ✓   this.datacontacto=[];
    ✓   this.domidom=[];
    ✓   this.listOfmain=[];
    ✓   this.Esvista=false;
    ✓   this.domisvista=false;
    ✓ }
    ✓ nuevocont(){
    ✓   this.cvisible=true;
    ✓   this.aggcontacto=new ContactoNew();
    ✓ }
    ✓ nuevocon(id_pat:number){
    ✓   this.ecvisible=true;
    ✓   this.editacontacto= new ContactoNew();
    ✓   this.contanew=true;
    ✓   this.editacontacto.id_patrocinio=id_pat;
    ✓ }
    ✓ conted(data:any,index:number){
    ✓   this.ecvisible=true;
    ✓   this.editacontacto.id=data.id;
    ✓   this.editacontacto.nombre=data.nombre;
    ✓   this.editacontacto.puesto=data.puesto;
    }

```

Figura 102 “funciones de agregación” (Datos Obtenidos en el Software visual basic framework angular 12).

En este paso se veo la función de domicilio donde será igual pero también vemos las funciones de los drawers que siguen estan ahí

```

    nuevodomi(){
        this.dvisible=true;
        this.aggdomi=new DomicilioNew();
    }

    nuevodomed(){
        this.edovisible=true;
        this.aggdomi=new DomicilioNew();
        this.contanew=true;
    }

    domied(data:any,dindex:number){
        this.edovisible=true;
        this.editadomi.id=data.id;
        this.editadomi.calle=data.calle;
        this.editadomi.colonia=data.colonia;
        this.editadomi.numex=data.numex;
        this.editadomi.numint=data.numint;
        this.editadomi.estado=data.estado;
        this.editadomi.municipio=data.municipio;
        this.editadomi.codigopostal=data.codigopostal;
        this.editadomi.pais=data.pais;
        this.idomi=dindex;
        this.dominew = false;

    }

    close(){
        this.Pvisible=false;
        this.edvisible =false;
    }

    cclose(){
        this.cvisible=false;
    }

    escclose(){
        this.ecvisable=false;
    }

    dclose(){
        this.edovisible=false ;
    }

    edclose(){
        this.edovisible=false;
    }

```

Figura 103 “funciones de patrocinio” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta parte se realizarán las funciones de agregar o edición en aceptar conti después de la validación y de iniciar la recarga de datos se introduce un if que si es contacto new ingrese el push para que haga el dato si no que se cambie lo del contacto

En agregar conti se hace lo mismo se pone el push, pero solo para agregar y hace es lo mismo con domicilio

```
✓ aceptaconti(){  
  for (const i in this.edcvalidateForm.controls){  
    this.edcvalidateForm.controls[i].markAsDirty();  
    this.edcvalidateForm.controls[i].updateValueAndValidity();  
  }  
  if (this.edcvalidateForm.valid){  
    this.Esvista=false;  
    if(this.contanew){  
      this.datacontacto.push(this.editacontacto);  
    }else{  
      this.datacontacto[this.icontac]=this.editacontacto;  
    }  
    console.log( "hecho",this.editacontacto);  
    this.ecvisible=false;  
    setTimeout(() => {  
      this.Esvista=true;  
    },200);  
  }  
}  
✓ Agregarconti(){  
  for (const i in this.cvalidateForm.controls) {  
    this.cvalidateForm.controls[i].markAsDirty();  
    this.cvalidateForm.controls[i].updateValueAndValidity();  
  }  
  if (this.cvalidateForm.valid){  
    this.Esvista=false;  
    this.datacontacto.push(this.aggcontacto);  
    console.log( "hecho",this.datacontacto);  
    this.cvisible=false;  
    setTimeout(() => {  
      this.Esvista=true;  
    }, 200);  
  }  
}  
✓ Aceptadomi(){  
  for (const i in this.dovalidateForm.controls) {  
    this.dovalidateForm.controls[i].markAsDirty();  
    this.dovalidateForm.controls[i].updateValueAndValidity();  
  }  
  if (this.dovalidateForm.valid){
```

Figura 104 “funciones de contacto y domicilio” (Datos Obtenidos en el Software visual basic framework angular 12).

En el edit. pat es donde se crea la función general para poder mandar a editar todo igual
en el open se hace una variable que evalúe el id

Y el patrocinio edit. es cuando abren todo el drawer y se usa para extraer los datos del
renglón de la tabla que se va editar

Y ya después se hace paso para correo

```
src > app > pages > patrocinio > TS patrocinio.component.ts > PatrocinioComponent > aceptaconti
359
360 Editapat(){
361   this.vista=false;
362   this.patrocinioiditado.lcontacto= this.datacontacto;
363   this.patrocinioiditado.ldomicilio=this.domidom;
364   this.Patsvr.editar(this.patrocinioiditado).subscribe(( res )=>{
365     console.log(res);
366     setTimeout(() => {
367       this.numreg=this.listOfmain.length;
368       this.vista=true;
369     }, 100);
370   });
371   this.edvisible=false;
372 }
373 patrocinioedit(reg:any){
374   this.edvisible=false;
375   this.ESvista=true;
376   this.domisvista=true;
377   this.patrocinioiditado.id=reg.id;
378   this.patrocinioiditado.id_giro=reg.id_giro;
379   this.patrocinioiditado.rfc=reg.rfc;
380   this.patrocinioiditado.razon_social=reg.razon_social;
381   let contacto=new ContactoNew;
382   contacto.id_patrocinio=reg.id;
383   this.edvisible=true;
384   this.Patsvr.getbyidpatrocinio(contacto).subscribe((res) =>{
385     console.log(res);
386     this.datacontacto=res;
387   });
388   let domicilio = new DomicilioNew;
389   domicilio.id_patrocinio=reg.id
390   this.edvisible=true;
391   this.Patsvr.getbyiddomicilio(domicilio).subscribe((res) =>{
392     console.log(DomicilioNew);
393     this.domidom=res;
394   });
395 }
396
397 opencorreo(data: any): void {
398   this.visiblecorreo = true;
399   this.patrocinioiditado.lcontacto=this.editacontacto=data;
400   this.asunto='';
401   this.mensaje='';
402   console.log('correo', this.editacontacto)
403 }
404
```

Figura 105 “funciones de edición y la de correo” (Datos Obtenidos
en el Software visual basic framework angular 12).

Crearemos el correo masivo y el individual se crea el HTML del correo individual

```
Go to component
<div class="user-indi">
  <h2>correo individual</h2>
  <form nz-form [formGroup]="validateForm" (ngSubmit)="enviaCorreo()">
    <nz-form-item>
      <nz-form-control nzErrorTip="Ingresa el nombre!">
        <nz-input-group>
          <input type="text" nz-input formControlName="nombre" placeholder="Nombre" [(ngModel)]="nombre">
        </nz-input-group>
      </nz-form-control>
    </nz-form-item>

    <nz-form-item>
      <nz-form-control nzErrorTip="Ingresa el email!">
        <nz-input-group>
          <input type="email" nz-input formControlName="email" placeholder="Email" [(ngModel)]="email">
        </nz-input-group>
      </nz-form-control>
    </nz-form-item>

    <nz-form-item>
      <nz-form-control nzErrorTip="Ingresa el asunto!">
        <nz-input-group>
          <input type="text" nz-input formControlName="asunto" placeholder="Asunto" [(ngModel)]="asunto">
        </nz-input-group>
      </nz-form-control>
    </nz-form-item>

    <nz-form-item>
      <nz-form-control nzErrorTip="Ingresa el mensaje!">
        <nz-input-group>
          <textarea rows="4" nz-input formControlName="mensaje" placeholder="Mensaje" [(ngModel)]="mensaje">
        </nz-input-group>
      </nz-form-control>
    </nz-form-item>

    <nz-form-item>
      <nz-form-control nzErrorTip="Ingresa el mensaje!">
        <nz-input-group>
          <nz-radio-group [(ngModel)]="mailType" formControlName="radio">
            <label nz-radio nzValue="0">Bienvenida</label>
            <label nz-radio nzValue="1">Moyco</label>
            <label nz-radio nzValue="2">Webinar</label>
          </nz-radio-group>
        </nz-input-group>
      </nz-form-control>
    </nz-form-item>
  </form>
</div>
```

Figura 106 “correo individual html” (Datos Obtenidos en el Software visual basic framework angular 12).

Pasaremos al ts donde tendremos el import de miembros y la interface de los datos

```

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { BienvenidaMail } from 'src/app/correos/bienvenida';
import { CorreoService } from 'src/app/services/correo.service';
import { MiembroResponse } from 'src/app/models/miembro';
import { MiembroService } from 'src/app/services/miembro.service';
import { Moycomail } from 'src/app/correos/moyco';
import { NzMessageService } from 'ng-zorro-antd/message';

interface ItemData{
  nombre: string;
  membresia: number;
  tipomembresia: string;
}

```

Figura 107 “librerías e interfaces” (Datos Obtenidos en el Software visual basic framework angular 12).

Declararemos los valores de espera y las interfaces después crearemos los servicios de correo y las plantillas a usar

```

mailType= '0';
miembro: ItemData;
asunto='';
nombres='';
emails='';
mensaje='';
validateForm!: FormGroup;
isAwaiting=false;

```

Figura 108 “variables” (Datos Obtenidos en el Software visual basic framework angular 12).

```

constructor(
  private fb: FormBuilder,
  public correoSrv: CorreoService,
  private moycom: Moycomail,
  private bienm: Bienvenidamail
) {
  this.miembro = {
    nombre: '',
    membresia: 0,
    tipomembresia: '',
  }
}

```

Figura 109 “constructor de correo individual” (Datos Obtenidos en el Software visual basic framework angular 12).

On int se encargará de verificar los datos mandados y que no haya campos vacíos

```

}

ngOnInit(): void {
  this.validateForm = this.fb.group({
    nombre: [null, [Validators.required]],
    email: [null, [Validators.required]],
    asunto: [null, [Validators.required]],
    mensaje: [null, [Validators.required]],
    radio: [null, [Validators.required]],
  });
}

```

Figura 110 “oninit en correo individual” (Datos Obtenidos en el Software visual basic framework angular 12).

En esta parte se mandará a ingresar los datos mediante el correo de mensaje siempre y cuando el bono se allá mandado, otra de las cosas el servicio se encargará del envío del mensaje y el enviar se hace para rectificar la dirección si ya está bien se pasa al enviar mensaje

Y también la validación de datos en onint

```
contactForm(form: { nombre: string; mensaje: string; }) {
  this.miembro.nombre=form.nombre;
  this.miembro.membresia= 129;
  this.miembro.tipomembresia= 'Silver';
  let esta = this;
  let idFolio = parseInt(this.mailType);
  switch (idFolio) {
    case 0:
      esta.mensaje = this.bienm.moymes(esta.mensaje);
      break;
    case 1:
      esta.mensaje = this.moym.moymes();
      break;
    default:
      esta.mensaje = form.mensaje;
  }
  this.correoSrv.sendMessage(form).subscribe(() => {
    console.log("Formulario de contacto", "Mensaje enviado correctamente", 'success');
  });
}

enviaCorreo(){
  this.isAwaiting= true;
  for (const i in this.validateForm.controls) {
    this.validateForm.controls[i].markAsDirty();
    this.validateForm.controls[i].updateValueAndValidity();
  }

  if(this.validateForm.valid){
    console.log("no se ingreso nada");
    this.isAwaiting= false;
  }else{
    this.isAwaiting= false;
  }
}
```

Activar Windows
Ve a Configuración para activar Windows

Figura 111 “correo masivo” (Datos Obtenidos en el Software visual
basic framework angular 12).

Mismo HTML en el ts cambian ya que ahora se manda a traer los datos de la lista de respuesta de miembros se crean las variables el mensaje de texto los servicios en el constructor

La interface de data correo el servicio de buscar todos los miembros que tengan correo

```

styleUrls: ['./correo-masiv.component.css']
})
export class CorreoMasivComponent implements OnInit {

  listOfData: MiembroResponse[] = [];
  dataCorreo: ItemData;
  mailType = '0';
  validateForm!: FormGroup;
  isAwaiting = false;
  mestexto = '';

  constructor(
    private fb: FormBuilder,
    private miembroSvr: MiembroService,
    public correoSrv: CorreoService,
    private msg: NzMessageService,
    private moym: Moycomail,
    private bienm: Bienvenidamail
  ) {
    this.dataCorreo = {
      asunto: '',
      email: '',
      mensaje: '',
      nombre: '',
    }
  }

  ngOnInit(): void {
    this.miembroSvr.getAll().subscribe((miembros) => {
      console.log('miembros', miembros);
      this.listOfData = miembros
    });

    this.validateForm = this.fb.group({
      asunto: [null, [Validators.required]],
      mensaje: [null, [Validators.required]],
      radio: [null, [Validators.required]],
    });
  }
}

```

Figura 112 “codigo de correo masivo ts2(Datos Obtenidos en el Software
visual basic framework angular 12).

La primera función es para checar si hubo algún cambio el submitfrom hace que el campo este lleno ya que después nos manda a la función de enviar

Después ingresaremos el switch case para ver qué es lo que mandaremos si bienvenida o promociones en moyco

```

handleChange(info: NzUploadChangeParam): void {
  if (info.file.status !== 'uploading') {
    console.log(info.file, info.filelist);
  }
  if (info.file.status === 'done') {
    this.msg.success(`${info.file.name} file uploaded successfully`);
  } else if (info.file.status === 'error') {
    this.msg.error(`${info.file.name} file upload failed.`);
  }
}

submitForm(): void {
  for (const i in this.validateForm.controls) {
    this.validateForm.controls[i].markAsDirty();
    this.validateForm.controls[i].updateValueAndValidity();
  }
  this.isAwaiting= true;
  this.Envia();
}

contactForm(form: any) {
  let esta = this;
  let idFolio = parseInt(this.mailType);
  switch (idFolio) {
    case 0:
      esta.dataCorreo.mensaje = this.bienm.moymes(esta.dataCorreo);
      break;
    case 1:
      esta.dataCorreo.mensaje = this.moym.moymes();
      break;
    default:
      esta.dataCorreo.mensaje = form.mensaje;
  }

  this.dataCorreo.asunto = form.asunto;
}

```

Figura 113 “codigo del ts submitrom” (*Datos Obtenidos en el Software visual basic framework angular 12*).

Después en envíos se van los datos de la información, así como la estructura de cómo se

ira ponen cada código y formatos

```
Enviar(): void {  
  
    let esta = this;  
    let idFolio = parseInt(this.mailType);  
  
    for (const key in this.validateForm.controls) {  
        this.validateForm.controls[key].markAsDirty();  
    }  
  
    if (this.validateForm.valid ) {  
        console.log('enviar');  
  
        switch (idFolio) {  
            case 0:  
                esta.dataCorreo.mensaje = this.bienm.moymes(esta.dataCorreo);  
                break;  
            case 1:  
                esta.dataCorreo.mensaje = this.moym.moymes();  
                break;  
            default:  
                esta.dataCorreo.mensaje = this.mestexto;  
        }  
  
        setTimeout(() => {  
            console.log('hi');  
  
            this.listOfData.forEach(function (miembro) {  
                esta.dataCorreo.nombre = miembro.nombre + " " + miembro.apellido + " " + miembro.apellido;  
                esta.dataCorreo.email = miembro.email;  
  
                esta.correoSrv.sendMessage(esta.dataCorreo).subscribe(() => {  
                    console.log("Formulario de contacto", "Mensaje enviado correctamente", 'success');  
                });  
            });  
            this.isAwaiting= false;  
        }, 500);  
    } else{  
        this.isAwaiting= false;  
    }  
}
```

Figura 114 “captura de la función enviar” (*Datos Obtenidos en el Software visual basic framework angular 12*).

Les mostrare el cascaron de todo lo que hizo

Para correo el sistema tienes que ir a la terminal y poner ng serve para que no tengas que truncarlo y volver a correr cada vez que hagas una modificación tienes que poner ng-server

0

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\desarrollo\admins> ng serve -c
```

Figure 115“es la implementación de el modo de arranque para que la aplicación con el comando de ng - serve para poder observar las interfaces” (Datos Obtenidos en el Software visual basic termial cmd de visual basic framework angular 12).

Login

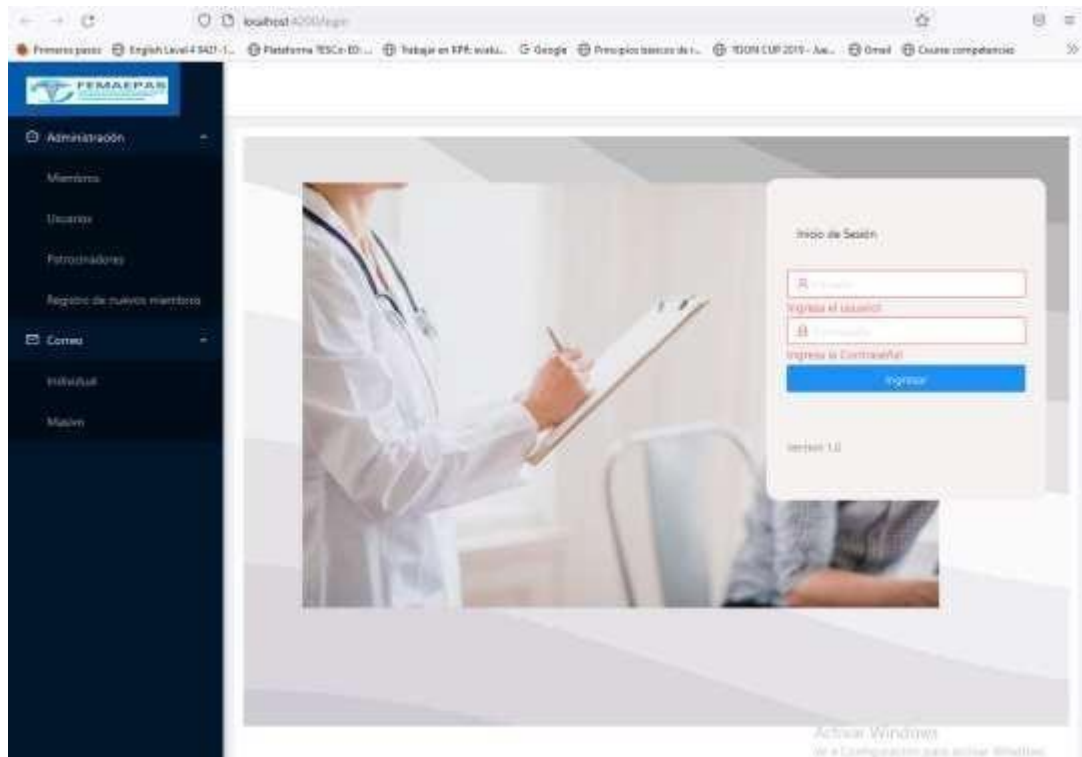


Figura 116 “interfaz de login” (Datos Obtenidos por el autor en el browser mozillafirefox angular

localhost 4200).

Registro de miembros

Datos necesarios para su registro

☐ Silver ☐ Gold ☒ platinum

Nombre completo Apellido paterno Apellido materno

Teléfono Correo Electrónico Estado o Localidad

País Nivel de escolaridad Área de especialización

Cédula Profesional Referencia por ...

Preview Card: FEMAEPAS, José Antonio Herrera Salas, 12100055, 2021, www.femaepas.com

Figura 117 “interfaz de registro de miembros” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).

Tabla de miembros

Monitor de Miembros

Nombre	Telefono	E-mail	Edad	Profesion	Especialidad	Acciones
--------	----------	--------	------	-----------	--------------	----------

Cargar

Figura 118 “Interfaz del monitor de miembros” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).

Usuarios

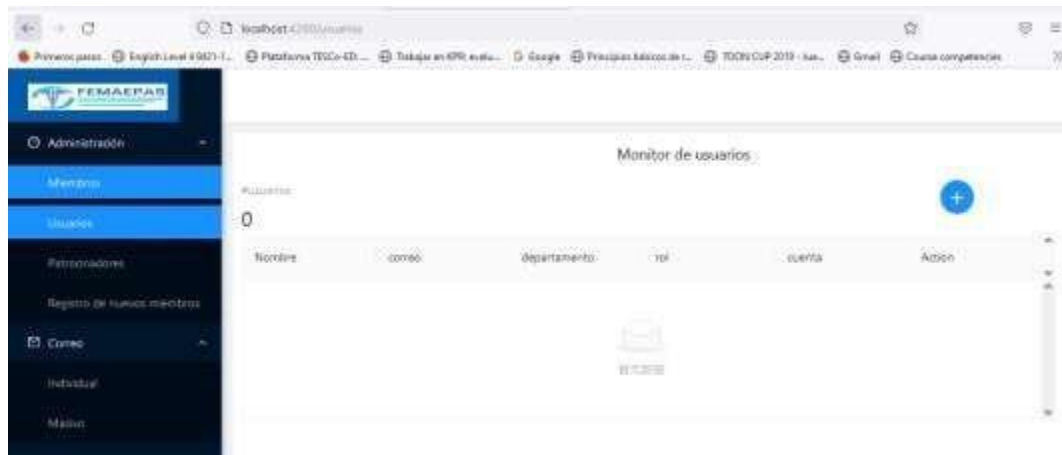


Figura 119 “interfaz de monitor de usuario” (Datos Obtenidos por el autor en el browser mozillafirox angular localhost 4200).

Drawer de registro la función de edición aparece cuando hago el backend por que la necesitamos la comunicación de la base de datos para poder extraer el id

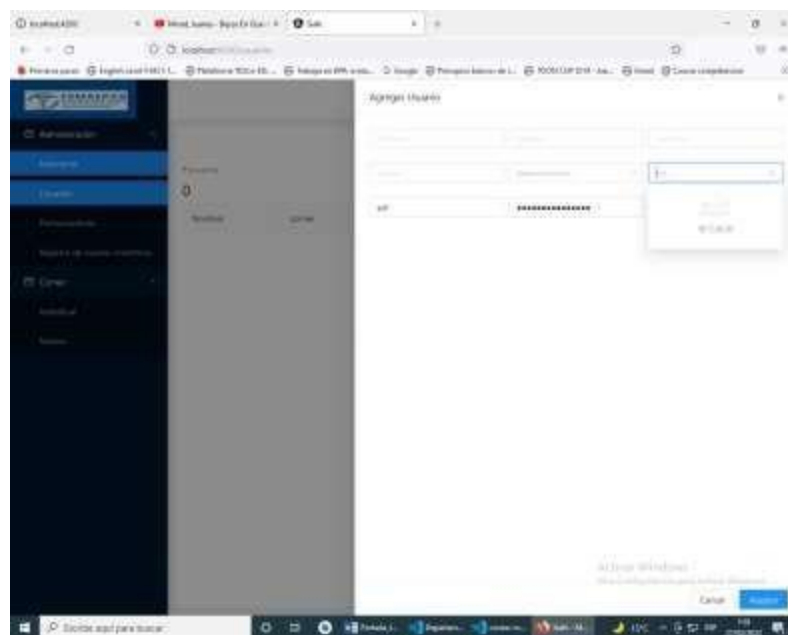


Figura 120 “drawers para la alta de usuarios” (Datos Obtenidos por el autor en el browser mozillafirox angular localhost 4200).

Patrocinador



Figura 121 “Interfaz del monitor de patrocinador” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).

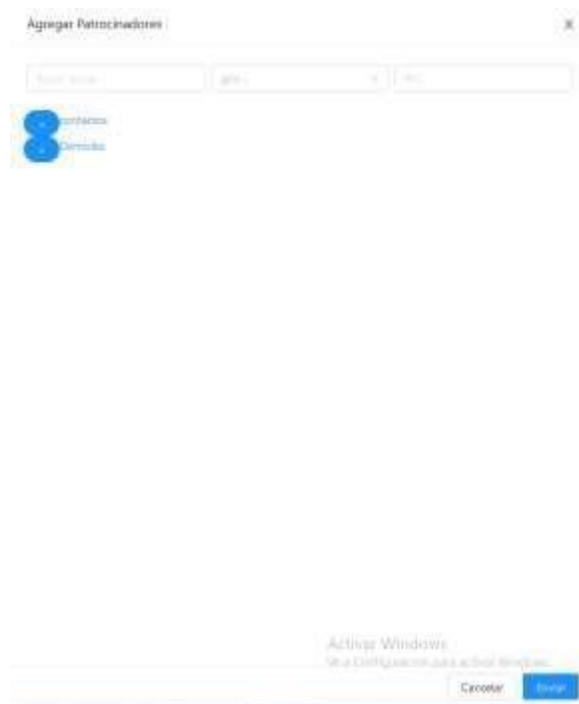


Figura 122 Interfaz de alta de usuarios(Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).

Lo curioso de esto es que este arreglo en los botones me permite hacer la agregación y edición de manera temporal él porque es fácil para poder grabar esta parte necesitamos el id de patrocinio y es mejor tener estos datos en memoria para guardar patrocinio y hacerlos editarlos

Agregar Patrocinadores
X

+ contactos

Nombre	Telefono	Puesto	Correo	Action
micHELL	44303032	ventasM	michi100.mr@gmail.com	

< 1 >

+ Domicilio

calle	colonia	numext	numint	municipio	estac
arroyo	acuales2	12	12	coaco	

< 1 >

Figura 123 “Intefaz de patrocinadores con la vista de las tablas de contactos y de datos” (Datos Obtenidos por el autor en el browser mozillafierox angular localhost 4200).

Para la edición temporal por si nos equivocamos y queremos repetirlo hacemos lo siguiente

Se da click en el lápiz y te manda al drawe de contacto, pero ya lo puedes llegar a cambiar temporal mente ya que está en memoria y como ven me trae todos los datos que se le pidieron

Edita el Contacto

×

micello

44303032

ventasm

michi100.mr@gmail.c

Figura 124 interfaz de la edicion de contacto(Datos Obtenidos por el autor en el browser mozillafirox angular localhost 4200).

edita Domicilio

×

arroyo

acuales

12

12

mexico

coacalco

1233m

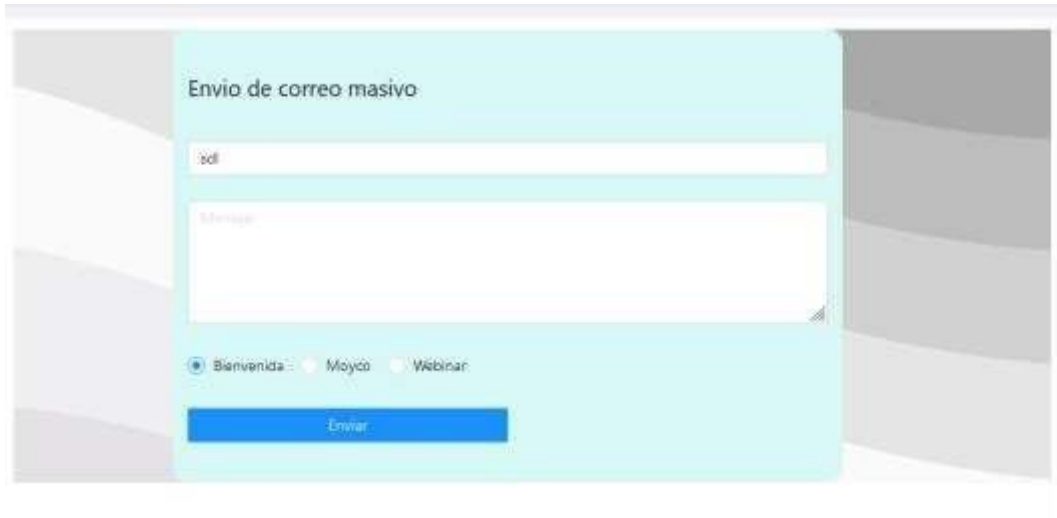
mexico

cerrar

aceptar

Figura 125 “Interfaz de domicilio” (Datos Obtenidos por el autor en el browser mozillafirox angular localhost 4200).

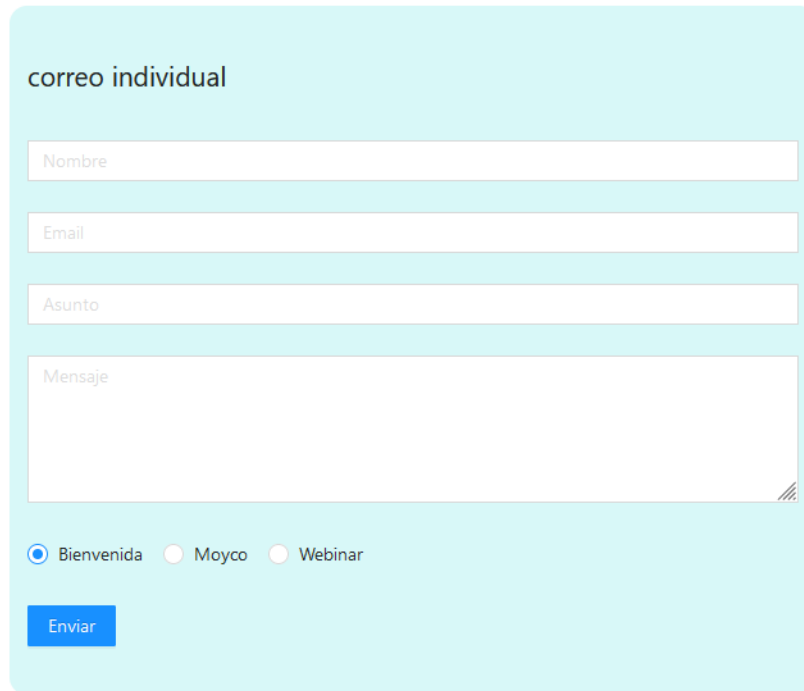
Correo masivo



The screenshot shows a web form titled "Envío de correo masivo" (Mass email sending). It features a light blue header bar with the title. Below the header, there is a white input field for the email address, containing the text "odl". Underneath the input field is a large white text area for the email body. At the bottom of the form, there are three radio buttons for selecting a template: "Bienvenida" (selected), "Mostrar", and "Webinar". A blue "Enviar" (Send) button is located at the bottom right of the form.

Figura 126 “Interfaz de correo masivo” (Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).

Correo individual



correo individual

Nombre

Email

Asunto

Mensaje

☒ Bienvenida ☐ Moyco ☐ Webinar

Enviar

Figura 127 “muestra de prueba del correo” (Datos Obtenidos en el Software visual basic terminal cmd de visual basic framework angular 12).

3.3. Creación del backend

Se usará la paquetería de los servicios que usaremos

```
1 {
2   "name": "appback",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "dev": "tsc-watch --onSuccess \"node dist/index.js\"",
8     "tsc": "tsc",
9     "start": "ts-node src/index.ts"
10  },
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "bcryptjs": "^2.4.3",
15    "class-validator": "^0.13.1",
16    "cors": "^2.8.5",
17    "express": "^4.17.1",
18    "jsonwebtoken": "^8.5.1",
19    "morgan": "^1.10.0",
20    "mysql": "^2.18.1",
21    "nodemailer": "^6.6.3",
22    "nodemailer-express-handlebars": "^4.0.0",
23    "reflect-metadata": "^0.1.13",
24    "typeorm": "^0.2.37"
25  },
26  "devDependencies": {
27    "@types/bcryptjs": "^2.4.2",
28    "@types/cors": "^2.8.12",
29    "@types/express": "^4.17.13",
30    "@types/jsonwebtoken": "^8.5.4",
31    "@types/morgan": "^1.9.3",
32    "@types/node": "^16.6.1",
33    "@types/nodemailer": "^6.4.4",
```

Figura 128 "info de pack json" (Datos Obtenidos por el autor en el software nodejs. backend.)

Después ingresaremos los datos para integrar los datos de la base de datos dentro el puerto 3306 después hacemos la referencia de datos en los entytis

```

{
  "type": "mysql",
  "host": "localhost",
  "port": 3306,
  "username": "root",
  "password": "michell",
  "database": "dbfemaepas",
  "synchronize": true,
  "logging": false,
  "entities": [
    "src/entity/**/*.ts"
  ],

  "cli": {
    "entitiesDir": "src/entity",
    "migrationsDir": "src/migration",
    "subscribersDir": "src/subscriber"
  }
}

```

Figura 129 “typorm” (Datos Obtenidos por el autor en el software no-dejs. backend.)

Se usará un índice donde se guardarán las rutas de las rutas que se crearán para poder hacer las funciones

```

C:\desarrollo\backend\src\routes express';
import miembro from './miembro';
import claverreferido from './claverreferido';
import usuario from './usuario';
import roles from './roles';
import departamento from './departamento';
import login from './login';
import correo from './correo';
import giro from './giro';
import patrocino from './patrocino';

const routes = Router();

routes.use('/correo', correo);
routes.use('/miembros', miembro);
routes.use('/referido', claverreferido);
routes.use('/usu', usuario);
routes.use('/depto', departamento);
routes.use('/rol', roles);
routes.use('/login', login);
routes.use('/giro', giro);
routes.use('/patrocinio', patrocino);

export default routes;

```

Figura 130 “rutas” (Datos Obtenidos por el autor en el software no-dejs. backend.)

Crearemos una ruta donde ira al usuario controller donde se mandará con un post para poder generar la búsqueda de cuenta

```
✓ import { Router } from 'express';  
import { UsuarioController } from '../controller/UsuarioController';  
  
const router = Router();  
  
router.post('/', UsuarioController.getByCuenta);  
  
export default router;
```

Figura 131 “ruta login” (Datos Obtenidos por el autor en el software nodejs. Backend)

Se vinculará al router de correo para hacer la función de envío de datos

```
1 ✓ import { Router } from 'express';  
2 import { CorreoController } from '../controller/CorreoController';  
3  
4 const router = Router();  
5  
6  
7 router.post('/', CorreoController.envio);  
8  
9 export default router;  
0
```

Figura 132 “ruta correo” (Datos Obtenidos por el autor en el software nodejs. backend.)

Aquí se crearán las rutas de las funciones de creación get hace referencia de buscar todos los usuarios

Y la búsqueda de getbyid se usará para buscar el id de uno por uno de los usuarios

Y las otras son de crear un nuevo usuario

Una edición y la eliminación

```
//import { UsuarioController } from '../controller/UsuarioControlle
import { Router } from 'express';
import { UsuarioController } from '../controller/UsuarioControlle

const router = Router();

// Get all users
router.get('/all', UsuarioController.getAll);

// Get one user
router.get('/:id', UsuarioController.getById);

// Create a new user
router.post('/', UsuarioController.new);

// Edit user
router.patch('/:id', UsuarioController.edit);

// Delete
router.delete('/:id', UsuarioController.delete);
```

Figura 133 “ruta usuario” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se hará una ruta de los roles para crearlo y obtener los datos de la información

```
✓ import { Router } from 'express';
import RolesController from '../controller/RolesController';

const router = Router();

// Get all
router.get('/all', RolesController.getAll);

// Get one
router.get('/:id', RolesController.getById);

// Create a new
router.post('/', RolesController.new);

export default router;
```

Figura 134 ruta roles (Datos Obtenidos por el autor en el software no-dejs. Backend)

Al igual que con departamento

```
✓ import { Router } from 'express';
  import DepartamentoController from '../controller/Departam

const router = Router();

// Get all users
router.get('/all', DepartamentoController.getAll);

// Get one user
router.get('/:id', DepartamentoController.getById);

// Create a new user
router.post('/', DepartamentoController.new);

export default router;
```

Figura 135 “ruta departamento” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se creará las rutas de miembros

```
import { MiembroController } from '../controller/MiembroController';
import { Router } from 'express';

const router = Router();

// Get all
router.get('/all', MiembroController.getAll);

// Get one
router.get('/:id', MiembroController.getById);

// Create a new
router.post('/', MiembroController.new);

// Edit
router.patch('/:id', MiembroController.edit);

// Delete
router.delete('/:id', MiembroController.delete);

export default router;
```

Figura 136 “ruta miembros” (Datos Obtenidos por el autor en el software nodejs. backend.)

También igual se creará una ruta del clave de referidos

```

✓ import { Router } from 'express';
import ClaveReferidoController from '../controller/ClaveReferidoContro

const router = Router();

// Get all
router.get('/all', ClaveReferidoController.getAll);

// Create a new
router.post('/', ClaveReferidoController.new);

export default router;

```

Figura 137 “ruta clave referido” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se creara la ruta para los giros que provendran el patrocinio

```

✓ import { Router } from 'express';
import GiroController from '../controller/GiroController';

const router = Router();

// Get all
router.get('/all', GiroController.getAll);

// Get one
// router.get('/:id', GiroController.getById);

// Create a new
router.post('/', GiroController.new);

export default router;

```

Figura 138 “ruta giro” (Datos Obtenidos por el autor en el software nodejs. backend.)

Igual se creará la ruta del patrocinio con referencia del controller

```
import { Router } from "express";
import { PatrocinioController } from "../controller/PatrocinioControll
import { Patrocinio } from "../entity/Patrocinio";

const router = Router();

// Get all
router.get('/all',PatrocinioController.getAll);
router.get('/allc',PatrocinioController.getAllC);
router.get('/alld',PatrocinioController.getAlld);
router.post('/contacto',PatrocinioController.getByidpatrcinio);
router.post('/domicilio',PatrocinioController.getByidomicilio);

router.post('/',PatrocinioController.new);
router.post('/edit',PatrocinioController.edit);

export default router;
```

Figura 139 “ruta patrocinio” (*Datos Obtenidos por el autor en el software nodejs. backend.*)

Crearemos los datos de la entidad donde se referenciará a la base de datos y donde se usarán para que viajen los datos a la aplicación

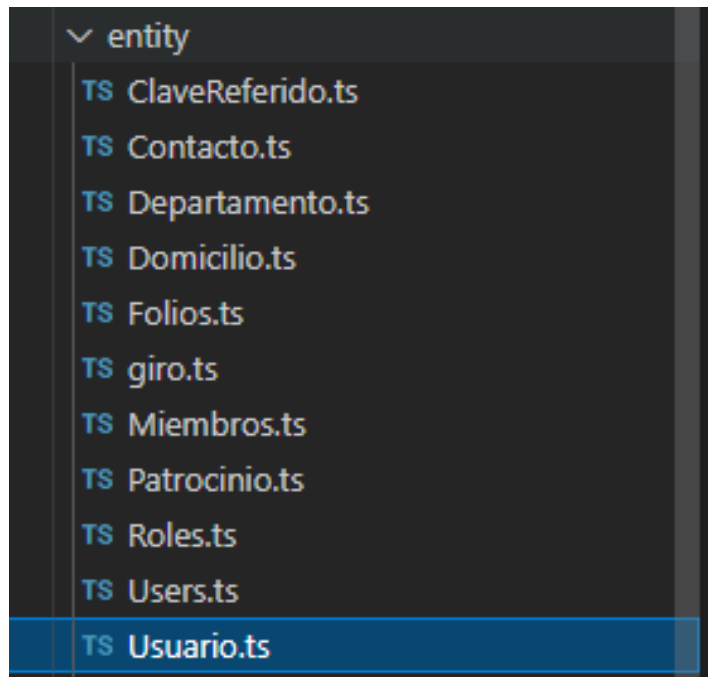


Figura 140 “creacion de los entytis” (Datos Obtenidos por el autor en el software nodejs. backend.)

A miembros y a usuarios se

Buscaran los datos lo de pasword y de la validación del pasword

```
hashPassword(): void {  
  const salt = bcrypt.genSaltSync(10);  
  this.password = bcrypt.hashSync(this.password, salt);  
}  
  
checkPassword(password: string): boolean {  
  return bcrypt.compareSync(password, this.password);  
}  
}
```

Figura 141 “funciones de encriptado y de verificacion de contraseña”
(Datos Obtenidos por el autor en el

Importaremos las librerías de requerimientos, datos al igual que ingresamos una librería de node que nos permite hacer servicios stop

Crearemos una función estática donde se crean las variables del requerimiento y de la respuesta al igual que creamos una variable constante donde se ingresará el cuerpo del requerimiento

```
import { Request, Response } from 'express';  
import nodemailer from 'nodemailer';  
  
export class CorreoController {  
  static envio = async (req: Request, res: Response) => {  
    const { asunto, email, mensaje, nombre } = req.body;
```

Figura 142 “correo controllador y la libreria de respuesta y requerimeintos” (Datos Obtenidos por el autor en el software nodejs. backend.)

En esta funcio de la creacion del transporte donde se ingresa el host donde tenemos el servidor de correos en este casa ya se usa este controlador en la pagina de femaepas.com

Y crearemos un puerto donde si es el mismo puerto sera de tipo true despues crearemos el usuario y contraseña para ingesar al servidor

Despues se creara un valor de tls donde se ingresara el erro si hay un cambio en la contraseña o el correo es falso

Usaremos los datos de la aplicación que llegaran ala informacion como por ejemplo el correo por defaul y aquin se le mandara el asunto y tambien se hara un mensaje de formato html

```
var transporter = nodemailer.createTransport({
  host: "single-2365.banahosting.com",
  port: 465, //587
  secure: true, // true for 465, false for other ports
  auth: {
    user: 'server@femaepas.com', // Cambialo por tu email
    pass: 'S3rverlog1' // Cambialo por tu password
  },
  tls: {
    // do not fail on invalid certs
    rejectUnauthorized: false,
  },
});
var mailOptions = {
  from: 'server@femaepas.com',
  to: email,
  subject: asunto,
  html: mensaje
};
```

Figura 143 “creacion del transporte” (Datos Obtenidos por el autor en el software nodejs. backend.)

Después buscaremos si la variable transporte se manda a la función error mandara el error

Y si no se mandara la respuesta en caso de que no haya ninguna de estas opciones se

mandara que ya existe

```
try {
  transporter.sendMail(mailOptions, function (error, info) {
    if (error) {
      console.log(error);
    } else {
      console.log('Email enviado: ' + info.response);
    }
  });
} catch (e) {
  return res.status(409).json({ message: 'el correo ya existe!' });
}
```

Figura 144 “validacion del transporte” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se importará el repositorio requerimiento y respuesta ahora se realizará la parte de departamento igual se importa la identidad

En el static getall donde se creará la conexión de la base de datos llamada departamento

```
import { getRepository } from 'typeorm';
import { Request, Response } from 'express';
import { Departamento } from '../entity/Departamento';

export class DepartamentoController {

  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(Departamento);
    let listData;
```

Figura 145 “controlador de departamento sus librerías y su variable estática” (Datos Obtenidos por el autor en el software nodejs. backend.)

Después crearemos la lista de datos donde haremos que busque si hay datos que imprima el dato si no que mande un mensaje de error donde diga no hay resultado

```
try {
  listData = await dataRepository.find();

  if (listData.length > 0) {
    res.send(listData);
  } else {
    res.status(404).json({ message: 'Not result' });
  }
} catch (e) {
  res.status(404).json({ message: 'Somenting goes wrong!' });
}
};
```

Figura 146 “función getall” (Datos Obtenidos por el autor en el software nodejs. backend.)

El get byid se usará el mismo proceso solo que aquí se obtiene el parámetro del id es el que se buscara dentro del id

La función new se usará los valores del departamento de id_status y la fecha de registro será igual al requerimiento de donde se usará un body después crearemos la constante de nuevo departamento después se ingresará los datos del departamento

Se hará la creación del puente de la conexión con la base de datos después usaremos que el repositorio se guarde si hay un error se mandara que existe si no hay error se mandara que se creó correctamente

```

static getById = async (req: Request, res: Response) => {
  const { id } = req.params;
  const dataRepository = getRepository(Departamento);
  try {
    const data = await dataRepository.findOneOrFail(id);
    res.send(data);
  } catch (e) {
    res.status(404).json({ message: 'Not result' });
  }
};

static new = async (req: Request, res: Response) => {
  const { departamento, id_status, fecha_reg } = req.body;
  const data = new Departamento();
  console.log('New departamento: ==> ', departamento);
  data.departamento = departamento;
  data.fecha_reg = fecha_reg;
  data.id_status = id_status;

  const dataRepository = getRepository(Departamento);
  try {
    await dataRepository.save(data);
  } catch (e) {
    console.log(res.statusMessage);
    return res.status(409).json({ message: ' ya existe' });
  }
  // All ok
  res.send(' creado');
};

```

Figura 147 “funciones del getById y la funcion new” (Datos Obtenidos por el autor en el software nodejs. backend.)

Después se creará el de rol que es lo mismo al igual que clave referido

Y del giro

```

import { getRepository } from 'typeorm';
import { Request, Response } from 'express';

import { Roles } from '../entity/Roles';

export class RolesController {

  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(Roles);
    let listData;

    try {
      listData = await dataRepository.find();

      if (listData.length > 0) {
        res.send(listData);
      } else {
        res.status(404).json({ message: 'Not result' });
      }
    } catch (e) {
      res.status(404).json({ message: 'Something goes wrong!' });
    }
  };

  static getById = async (req: Request, res: Response) => {
    const { id } = req.params;
    const dataRepository = getRepository(Roles);
    try {
      const data = await dataRepository.findOneOrFail(id);
      res.send(data);
    } catch (e) {
      res.status(404).json({ message: 'Not result' });
    }
  };

  static new = async (req: Request, res: Response) => {
    const { rol, id_status, fecha_reg } = req.body;
    const data = new Roles();
    console.log('New rol: ==> ', rol);
    data.rol = rol;
    data.fecha_reg = fecha_reg;
    data.id_status = id_status;

    const dataRepository = getRepository(Roles);
    try {

```

Figura 148 “funciones de roles controllers” (Datos Obtenidos por el autor en el software nodejs. backend.)

```

import { getRepository } from 'typeorm';
import { Request, Response } from 'express';
import { ClaveReferido } from '../entity/ClaveReferido';
export class ClaveReferidoController {
  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(ClaveReferido);
    let listData;

    try {
      listData = await dataRepository.find();

      if (listData.length > 0) {
        res.send(listData);
      } else {
        res.status(404).json({ message: 'Not result' });
      }
    } catch (e) {
      res.status(404).json({ message: 'Somenthing goes wrong!' });
    }
  };

  static new = async (req: Request, res: Response) => {
    const { clave,nombre,id_status,fecha_reg} = req.body;
    const data = new ClaveReferido();
    console.log('New clave: ==> ',clave);
    data.clave = clave
    data.nombre = nombre;
    data.id_status = id_status;
    data.fecha_reg =fecha_reg;

    const dataRepository = getRepository(ClaveReferido);
    try {
      await dataRepository.save(data);
    } catch (e) {
      return res.status(409).json({ message: 'clave ya existe' });
    }
    // All ok
    res.send('clave creada');
  };
}
export default ClaveReferidoController;

```

Figura 149 “funciones de clave referido controller” (Datos Obtenidos por el autor en el software nodejs. backend.)

```

import { getRepository } from 'typeorm';
import { Request, Response } from 'express';
import { Giro } from '../entity/giro';

export class GiroController {

  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(Giro);
    let listData;

    try {
      listData = await dataRepository.find();

      if (listData.length > 0) {
        res.send(listData);
      } else {
        res.status(404).json({ message: 'Not result' });
      }
    } catch (e) {
      res.status(404).json({ message: 'Somenthing goes wrong!' });
    }
  };

  static new = async (req: Request, res: Response) => {
    const { giro, id_status, fecha_reg } = req.body;
    const data = new Giro();
    console.log('New rol: ==> ', Giro);
    data.giro = giro;
    data.fecha_reg = fecha_reg;
    data.id_status = id_status;

    const dataRepository = getRepository(Giro);
    try {
      await dataRepository.save(data);
    } catch (e) {
      console.log(res.statusMessage);
      return res.status(409).json({ message: 'el giro ya esta' });
    }
    // All ok
    res.send('giro creado');
  };
}

```

Figura 150 “funciones de girocontroller” (Datos Obtenidos por el autor en el software nodejs. backend.)

Crearemos los controladores de miembro y el de usuario

Las functions son parecidas de datos en el get all y get id

```
import { getRepository } from 'typeorm';
import e, { Request, Response } from 'express';
import { Usuario } from '../entity/Usuario';
export class UsuarioController {

  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(Usuario);
    let listData;

    try {
      listData = await dataRepository.find();

      if (listData.length > 0) {
        res.send(listData);
      } else {
        res.status(404).json({ message: 'Not result' });
      }
    } catch (e) {
      res.status(404).json({ message: 'Somethings goes wrong!' });
    }
  };

  static getById = async (req: Request, res: Response) => {
    const { id } = req.params;
    console.log('parametro', id);
    const dataRepository = getRepository(Usuario);
    try {
      const data = await dataRepository.findOneOrFail({
        where: {
          id: id
        }
      });
      console.log('datos', data);
      res.send(data);
    } catch (e) {
      console.log('error', e);
      res.status(404).json({ message: 'Not result' });
    }
  };
};
```

Figura 151 “funciones de recuperación de datos” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se creará el buscar por cuenta donde hay un parámetro que será la cuenta y el password

donde se hará el canal a la base de datos donde buscará la cuenta de la base de datos

Y después se checará el password donde si el password es correcto se manda los datos si no se manda un error

```
static getByCuenta = async (req: Request, res: Response) => {
  const { cuenta, password } = req.body;
  console.log('login Usuario: ==> ', req.body);
  const dataRepository = getRepository(Usuario);
  const datausu = new Usuario();
  try {
    const datausu = await dataRepository.findOneOrFail({
      where: {
        cuenta: cuenta
      }
    });
    console.log("hay un error", datausu);

    const pass = datausu.checkPassword(password);

    if (pass) {
      res.send(datausu);
    } else {
      res.status(205).json({ message: 'Password Incorrecto' });
    }
  } catch (e) {
    res.status(404).json({ message: 'Not result' });
  }
};
```

Figura 152 “función getbycuenta” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se creará la función del new del requerimiento y la respuesta y que se creara una constante donde se mandaran los datos que se enviaran dentro de la aplicación después se creara los datos del repositorio y se mandaran a guardar después se mandara una función de encriptado de datos

```

static new = async (req: Request, res: Response) => {
  const { nombre, ap_paterno, ap_materno, correo, id_departamento, id_rol, cuenta, password, id_status } = req.body;
  const data = new Usuario();

  console.log('New Usuario: ==> ', nombre);
  console.log('datos', req.body);
  data.nombre = nombre;
  data.ap_paterno = ap_paterno;
  data.ap_materno = ap_materno;
  data.correo = correo;
  data.id_departamento = id_departamento;
  data.id_rol = id_rol;
  data.cuenta = cuenta;
  data.password = password;
  data.fecha_reg = fecha_reg;
  data.id_status = id_status;

  const dataRepository = getRepository(Usuario);

  try {
    data.hashPassword();
    await dataRepository.save(data);
    return res.status(201).json({ status: 'Ok', message: 'Se agrego usuario ' });
  } catch (e) {
    console.log(res.statusMessage);
    return res.status(409).json({ status: 'No', message: 'Usuario ya existe' });
  }
};

```

Figura 153 “función new de usuario” (Datos Obtenidos por el autor en el software nodejs. backend.)

Crearemos la función de edición donde se crea primero se crea el parámetro para la búsqueda del id después se creará la respuesta que serán los demás datos se crea el canal de datos para el sistema

La data será igual al repositorio y se buscara el id después se igualarán los datos metidos con la aplicación y la base de datos crearemos un error donde si no en cuenta el id será que no encontró el usuario

```

static edit = async (req: Request, res: Response) => {
  let data;
  const { id } = req.params;
  const { nombre, ap_paterno, ap_materno, correo, id_departamento, id_rol, cuenta, password, id_status } = req.body;
  const dataRepository = getRepository(Usuario);
  console.log('parametro', id);
  console.log('edit Usuario: ==> ', nombre);
  console.log('datos', req.body);
  try {
    data = await dataRepository.findOneOrFail({
      where: {
        id: id
      }
    });
    data.nombre = nombre;
    data.ap_paterno = ap_paterno;
    data.ap_materno = ap_materno;
    data.correo = correo;
    data.id_departamento = id_departamento;
    data.id_rol = id_rol;
    data.cuenta = cuenta;
    data.password = password;
    data.id_status = id_status;
    data.fecha_reg = fecha_reg;
  } catch (e) {
    console.log('erro', e);
    return res.status(404).json({ message: 'Usuario no encontrado' });
  }
}

```

Figura 154 “función editar datos” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se creará un dónde se guarden datos y servicios de actualización y la función del valor de borrado

```

    try {
      await dataRepository.save(data);
    } catch (e) {
      return res.status(409).json({ message: 'Usuario esta en uso' });
    }

    res.status(201).json({ message: 'Usuario Actualizado' });
  };

  static delete = async (req: Request, res: Response) => {
    const { id } = req.params;
    const dataRepository = getRepository(Usuario);
    let data: Usuario;

    try {
      data = await dataRepository.findOneOrFail(id);
    } catch (e) {
      return res.status(404).json({ message: 'Usuario no encontrado' });
    }
  };

```

Figura 155 “funciones de eliminar de usuario y la parte de guardar los datos que se editaron” (Datos Obtenidos por el autor en el software nodejs. backend.)

Ahora crearemos un miembro

Se importa las librerías del repositorio request y miembros

Se creará el get all

```

import { getRepository } from 'typeorm';
import { Request, Response } from 'express';
import { Miembros } from '../entity/Miembros';
export class MiembroController {
  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(Miembros);
    let listData;

    try {
      listData = await dataRepository.find({
        select: ['id',
          'nombre',
          'apellidop',
          'apellidom',
          'telefono',
          'email',
          'estado',
          'password',
          'profesion',
          'escolaridad',
          'cedula',
          'clavereferido',
          'membresia',
          'tipomembresia',
          'especialidad',
          'role',
          'fechareg'
        ]
      });
    } catch (e) {
      res.status(404).json({ message: 'Algo salio mal!' });
    }

    if (listData.length > 0) {
      res.send(listData);
    } else {
      res.status(404).json({ message: 'No hay datos' });
    }
  }
}

```

Figura 156 “funcion para obtener datos” (Datos Obtenidos por el autor en el software nodejs. backend.)

Después se crearán las funciones de id

```
static getById = async (req: Request, res: Response) => {
  const { id } = req.params;
  const userRepository = getRepository(Miembros);
  try {
    const user = await userRepository.findOneOrFail(id);
    res.send(user);
  } catch (e) {
    res.status(404).json({ message: 'Not result' });
  }
};
```

Figura 157 “función de buscar id” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se creara la búsqueda de email donde el canal del repositorio donde se busca el email el cual la respuesta mandara el dato

```
static getEmail = async (req: Request, res: Response) => {
  const { email } = req.params;
  const dataRepository = getRepository(Miembros);
  try {
    const data = await dataRepository.findOneOrFail(email);
    res.send(data);
  } catch (e) {
    res.status(404).json({ message: 'Not result' });
  }
};
```

Figura 158 “funcion de obtener datos por email” (Datos Obtenidos por el autor en el software nodejs. backend.)

Crearemos una función del new donde se usará el requerimiento y la respuesta de datos

Se crea su constante de los miembros y datos que se mandaran la aplicación

```

static new = async (req: Request, res: Response) => {

  const { nombre, apellidop, apellidom, telefono, email, estado, profesion, escolaridad,
    especialidad, cedula, claverreferido, password, membresia, tipomembresia, role, fechareg }
    = req.body;
  const data = new Miembros();
  console.log('New miembro: ==> ', nombre);
  data.nombre = nombre;
  data.apellidop = apellidop;
  data.apellidom = apellidom;
  data.telefono = telefono;
  data.email = email;
  data.estado = estado;
  data.profesion = profesion;
  data.escolaridad = escolaridad;
  data.especialidad = especialidad;
  data.cedula = cedula;
  data.claverreferido = claverreferido;
  data.password = password;
  data.membresia = membresia;
  data.tipomembresia = tipomembresia;
  data.role = role;
  data.fechareg = fechareg;

```

Figura 159 “inicio de la función miembro” (Datos Obtenidos por el autor en el software nodejs. backend.)

En esta parte se ingresa los datos del servicio se crea el miembro y se manda a grabar después se verifica si el correo no está repetido si es el caso se manda al catch donde te dice que ya existe

```

const dataRepository = getRepository(Miembros);
try {

  const newmiembro = await dataRepository.save(data);
  console.log('New: ==>', newmiembro);

  res.send(newmiembro);

} catch (e) {
  return res.status(400).json({ message: ' ya existe!' });
};

```

Figura 160 “variable del repositorio donde se guarda miembros” (Datos Obtenidos por el autor en el software nodejs. backend.)

Se hace la edición es el mismo proceso que el de usuarios después la función

hacemos el de eliminar

```
let data;
const { id } = req.params;
const { nombre, apellidop, apellidom, telefono, email, estado, profesion, escolaridad, especialidad, cedula, claverreferido, password, membresia, tipomembresia, role, fechareg } = req.body;

const dataRepository = getRepository(Miembros);
// Try get user
try {
  data = await dataRepository.findOneOrFail(id);
  data.nombre = nombre;
  data.apellidop = apellidop;
  data.apellidom = apellidom;
  data.telefono = telefono;
  data.email = email;
  data.estado = estado;
  data.profesion = profesion;
  data.escolaridad = escolaridad;
  data.especialidad = especialidad;
  data.cedula = cedula;
  data.claverreferido = claverreferido;
  data.password = password;
  data.membresia = membresia;
  data.tipomembresia = tipomembresia;
  data.role = role;
  data.fechareg = fechareg;
} catch (e) {
  return res.status(404).json({ message: 'Miembro no encontrado' });
}

try {
  await dataRepository.save(data);
} catch (e) {
  return res.status(409).json({ message: 'Miembro esta en uso' });
}

res.status(201).json({ message: 'Miembro Actualizado' });
};
```

Figura 161 “función editar” (Datos Obtenidos por el autor en el software nodejs. backend.)

```

static delete = async (req: Request, res: Response) => {
  const { id } = req.params;
  const userRepository = getRepository(Miembros);
  let user: Miembros;

  try {
    user = await userRepository.findOneOrFail(id);
  } catch (e) {
    return res.status(404).json({ message: 'Miembro no encontrado' });
  }

};

export default MiembroController;

```

Figura 162 “función eliminar eliminar” (Datos Obtenidos por el autor en el software nodejs. backend.)

Crearemos la los imports de patrocinio de contacto y domicilio

```

import { getRepository } from "typeorm";
import e, { Request, Response } from 'express';
import { Patrocinio } from "../entity/Patrocinio";
import { Contacto } from "../entity/Contacto";
import { Domicilio } from "../entity/Domicilio";

```

Figura 163 “librería de patrocinio” (Datos Obtenidos por el autor en el software nodejs. backend.)

Después se creará la función de la función de obtener la búsqueda de la base de datos si la anchura de los datos es mayor a cero devolverá los datos de la base de datos si no mandara que hay un error

```

export class PatrocinioController {
  static getAll = async (req: Request, res: Response) => {
    const dataRepository = getRepository(Patrocinio);

    let listData;

    try {
      listData = await dataRepository.find();

      if (listData.length > 0) {
        res.send(listData);
      } else {
        res.status(404).json({ message: 'Not result' });
      }
    } catch (e) {
      res.status(404).json({ message: 'Somenthing goes wrong!' });
    }
  };
};

```

Figura 164 “función gett all” (Datos Obtenidos por el autor en el software nodejs. backend.)

Función para buscar el id de patrocinio en los campos de domicilio y contacto donde se buscarán mediante el id del patrocinio donde se hará la función del patrocinio

```

static getByidpatrcinio = async (req: Request, res: Response) => {

  const { id_patrocinio } = req.body;
  console.log(' patrocinio ==> ', req.body);
  const dataRepository = getRepository(Contacto);
  let listadata;
  try {
    listadata = await dataRepository.find({
      where: {
        id_patrocinio: id_patrocinio
      }
    });
    res.send(listadata);
  } catch (e) {

    res.status(404).json({ message: 'No resultados' });
  }
};

static getByidomicilio = async (req: Request, res: Response) => {
  const { id_patrocinio } = req.body;
  console.log(' patrocinio ==> ', req.body);
  const dataRepository = getRepository(Domicilio);
  let listadata;
  try {
    listadata = await dataRepository.find({
      where: {
        id_patrocinio: id_patrocinio
      }
    });
    res.send(listadata);
  } catch (e) {

    res.status(404).json({ message: 'No resultados' });
  }
};

```

Figura 165 “funciones de busquedas para buscar domicilio contacto y la id del patrocinio” (Datos Obtenidos por el autor en el software nodejs. backend.)

En esta parte se indicará que los valores de la constante del nuevo patrocinio donde se conectará el canal del patrocinio se usará una función try donde el nuevo patrocinio se guardará en el repositorio

```

static new = async (req: Request, res: Response) => {
  const { razon_social, id_giro, rfc, id_status, lcontacto, ldomicilio, fecha_reg }
    = req.body;
  const data = new Patrocinio();

  console.log('New Patrocinio: ==> ', razon_social);
  console.log('datos', req.body);
  data.razon_social = razon_social;
  data.id_giro = id_giro;
  data.rfc = rfc;
  data.id_status = id_status;
  data.fecha_reg = fecha_reg;
  const dataRepository = getRepository(Patrocinio);
  try {
    const newpatrocinio = await dataRepository.save(data);
  }
}

```

Figura 166 “funcion nuev” (Datos Obtenidos por el autor en el software nodejs. backend.)

Después ingresaremos el repositorio dentro del contacto donde los nombres sea igual al arreglo de contactos después ingresaremos la variable del nuevo contacto será igual a la función de guardar datos

```

const newpatrocinio = await dataRepository.save(data);
console.log("newpatrocinio", newpatrocinio);
for (let i in lcontacto) {
  const datas = new Contacto();
  const datasRepository = getRepository(Contacto);
  datas.nombre = lcontacto[i].nombre;
  datas.id_patrocinio = newpatrocinio.id;
  datas.correo = lcontacto[i].correo;
  datas.puesto = lcontacto[i].puesto;
  datas.telefono = lcontacto[i].telefono;
  datas.id_status = lcontacto[i].id_status;
  datas.fecha_reg = lcontacto[i].fecha_reg;
  console.log("valor", i);
  console.log("datos", datas);
  const newcontacto = await datasRepository.save(datas);
  console.log("contacto", newcontacto);
}

```

Figura 167 “contacto nuevos” (Datos Obtenidos por el autor en el software nodejs. backend.)

Y eso se hará igual con domicilio después se ingresará un contador de tiempo el cual después de 100 milisegundos mandará a grabar todo el dato y si falla te manda que ya existe o no lo encuentra

```
const datasRepository = getRepository(Domicilio);
damas.calle = ldomicilio[i].calle;
damas.id_patrocinio = newpatrocinio.id;
damas.colonia = ldomicilio[i].colonia;
damas.numexe = ldomicilio[i].numex;
damas.numint = ldomicilio[i].numint;
damas.municipio = ldomicilio[i].municipio;
damas.estado = ldomicilio[i].estado;
damas.codigopostal = ldomicilio[i].codigopostal;
damas.pais = ldomicilio[i].pais;
damas.id_status = lcontacto[i].id_status;
damas.fecha_reg = lcontacto[i].fecha_reg;
console.log("valor", i);
console.log("datos", damas);
const newcontacto = await datasRepository.save(damas);
console.log("contacto", newcontacto);
}

setTimeout(() => {

    return res.send(newpatrocinio);
}, 100);

} catch (e) {
    console.log(res.statusMessage);
    return res.status(409).json({ status: 'No', message: ' ya existe' });
}

};
```

Figura 168 creación del patrocinio de domicilio(Datos Obtenidos por el autor en el software nodejs. Backend).

En esta parte se creara el repositorio de patrocinio donde se primero se buscara el id del

patrocinio después se guardara los datos en el repositorio después se creara un repositorio de contacto donde se muestra el canal del contacto donde se crea un ciclo para el contacto donde se crea un data nuevo después se pregunta si el id es mayor a cero se buscara el id y se mandara al body de ese dato pero si no hay datos se creara uno nuevo incluyendo sus datos de la fecha de registro y del status y después se crearan los patrocinios para domicilio

```
const { id, razon_social, id_giro, rfc, id_status, lcontacto, ldomicilio, fecha_reg } = req.body;
const dataRepository = getRepository(Patrocinio);

console.log('edit patrocinio: ==> ', razon_social);
console.log('datos', req.body);
try {
  data = await dataRepository.findOneOrFail({
    where: {
      id: id
    }
  });
  data.razon_social = razon_social;
  data.id_giro = id_giro;
  data.rfc = rfc;
  await dataRepository.save(data);

  const datasRepository = getRepository(Contacto);
  for (let i in lcontacto) {
    let datas = new Contacto();
    if(lcontacto[i].id > 0){
      datas = await datasRepository.findOneOrFail({
        where: {
          id: lcontacto[i].id
        }
      });
    } else {
      datas.id_status = lcontacto[i].id_status;
      datas.fecha_reg = lcontacto[i].fecha_reg;
    }
    datas.id_patrocinio = lcontacto[i].id_patrocinio;
    datas.nombre = lcontacto[i].nombre;
    datas.correo = lcontacto[i].correo;
    datas.puesto = lcontacto[i].puesto;
    datas.telefono = lcontacto[i].telefono;
    await datasRepository.save(datas);
  }
}
```

Figura 169 “editar patrocinadores ,contacto y domicilio” (Datos
Obtenidos por el autor en el software nodejs. backend.)

Se creará domicilio donde se llevará el mismo concepto que arriba

```

const datamRepository = getRepository(Domicilio);
for (let i in ldomicilio){
    let damas= new Domicilio();
    if(lcontacto[i].id>0 ){
damas = await datamRepository.findOneOrFail({
    where: {
        id: ldomicilio[i].id
    }
});
    }else{
        damas.id_status=lcontacto[i].id_status;
        damas.fecha_reg=lcontacto[i].fecha_reg;
    }
    damas.calle = ldomicilio[i].calle;
    damas.colonia = ldomicilio[i].colonia;
    damas.numexe= ldomicilio[i].numex;
    damas.numint = ldomicilio[i].numint;
    damas.municipio = ldomicilio[i].municipio;
    damas.estado = ldomicilio[i].estado
    damas.codigopostal = ldomicilio[i].codigopostal;
    damas.pais = ldomicilio[i].pais;
    console.log("datos", damas);
    await datamRepository.save(damas);
}
}

```

Figura 170. “Función de Domicilio”. (Datos Obtenidos por el autor en el software Visual Stdudio Code).

Después de guardar se hará una búsqueda si lo encuentran se activará la respuesta de actualizado si no se usara la respuesta de no encontrado

```

    await datamRepository.save(damas);
  }
} catch (e) {
  console.log('erro', e)
  return res.status(404).json({ message: 'no encontrado' });
}

res.status(201).json({ message: ' Actualizado' });
;

export default PatrocinioController;

```

Figura 171 “*catch de patrocinio*” (Datos Obtenidos por el autor en el software nodejs. backend.)

Haremos una prueba de que funciona el get all y una el new y el byid que en todos los controllers son iguales esta prueba se hará con insomnia

Para abrir el puerto del backend es npm start

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma

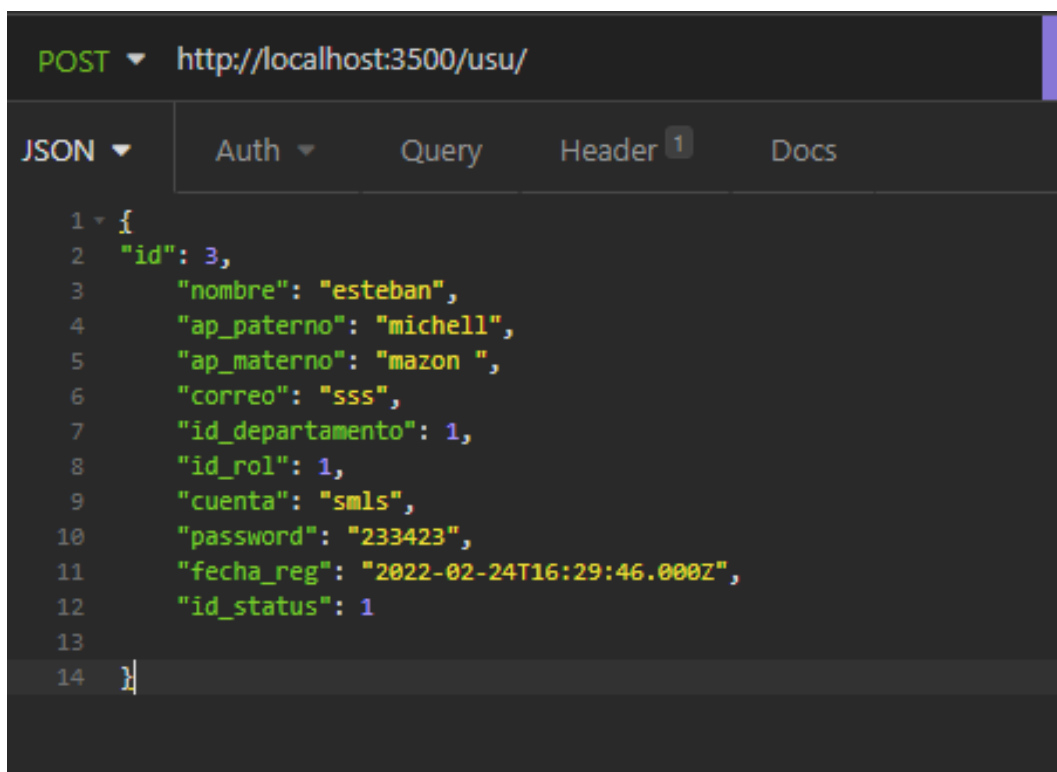
PS C:\desarrollo\backend> npm start

> appback@1.0.0 start
> ts-node src/index.ts

```

Figura 172 modo de arranque del servidor *Datos Obtenidos en el Software visual basic terminal cmd de visual basic framework angular 12).*

Crearemos un usuario y veremos si sirve para la respuesta de datos



```
POST http://localhost:3500/usu/

JSON Auth Query Header 1 Docs

1 {
2   "id": 3,
3   "nombre": "esteban",
4   "ap_paterno": "michell",
5   "ap_materno": "mazon ",
6   "correo": "sss",
7   "id_departamento": 1,
8   "id_rol": 1,
9   "cuenta": "smls",
10  "password": "233423",
11  "fecha_reg": "2022-02-24T16:29:46.000Z",
12  "id_status": 1
13
14 }
```

Figura 173. “Prueba insomnia con el registro de un usuario *Iremos a la parte de backend y se debe demostrar*”.(Datos Obtenidos por el autor en el software Insomnia).

```
New Usuario: ==> esteban
datos {
  id: 3,
  nombre: 'esteban',
  ap_paterno: 'michell',
  ap_materno: 'mazon ',
  correo: 'sss',
  id_departamento: 1,
  id_rol: 1,
  cuenta: 'sm1s',
  password: '233423',
  fecha_reg: '2022-02-24T16:29:46.000Z',
  id_status: 1
}
POST /usu/ 201 1023.760 ms - 46
```

Figura 174. “mensaje del bakend cuando ingresa un usuario”.(Datos Obtenidos por el autor en el software nodejs. backend.

Chcaremos la get all como vemos ya funciona el backend

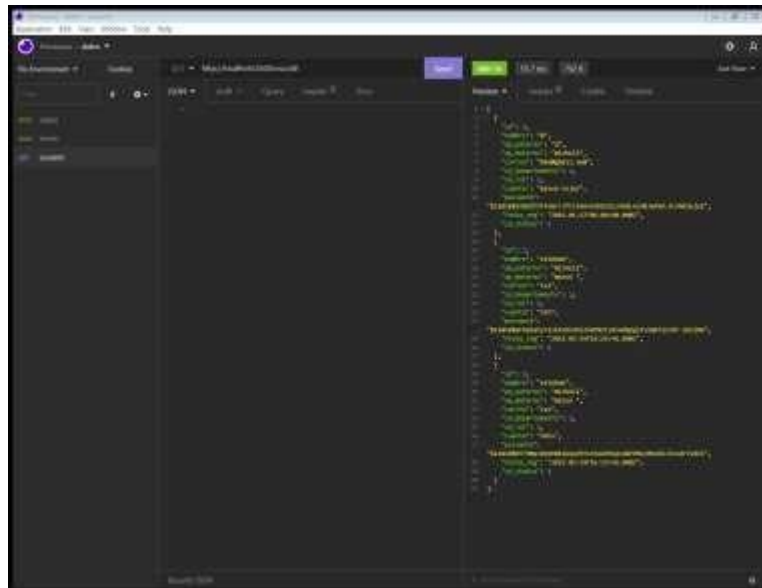


Figura 175 “entorno de prueba” ”.(Datos Obtenidos por el autor en el software Insomnia).

Capitulo IV Resultados

Login

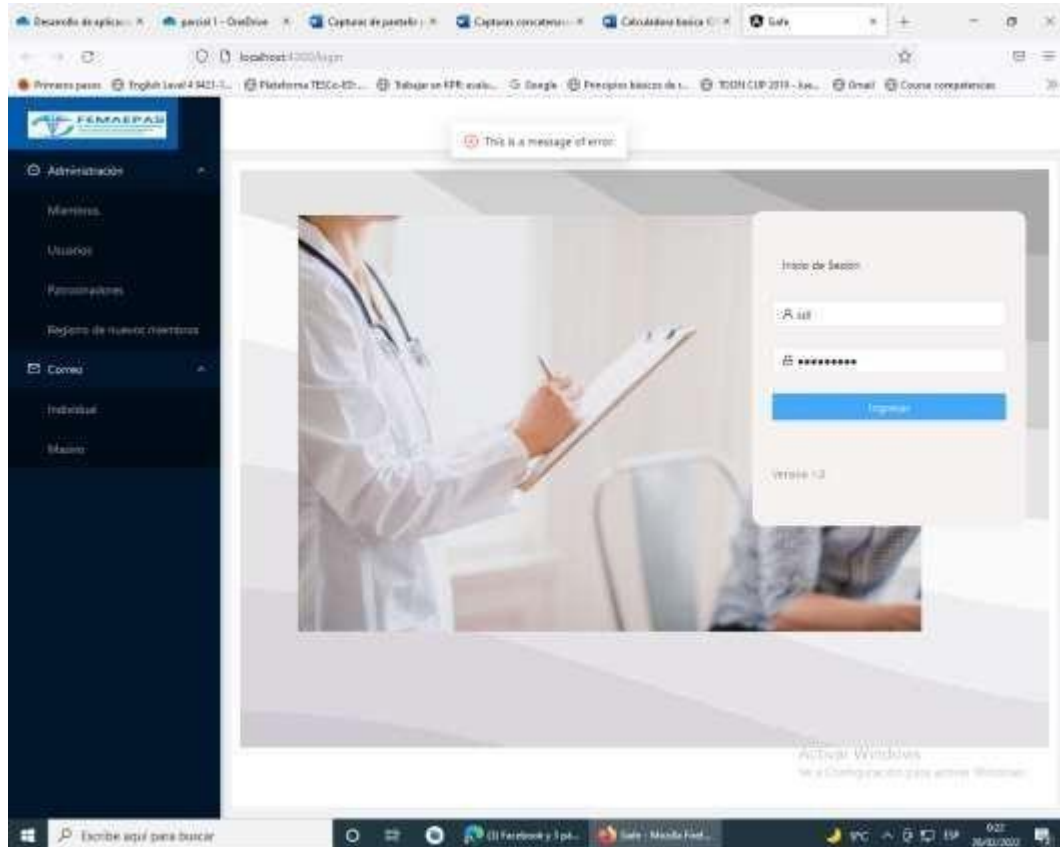


Figura 176: "mensaje de error en el sistema al no introducir bien la contraseña y la cuenta" ".(Datos Obtenidos por el autor en el browser mozillafirefox angular localhost 4200).

```
POST /login 205 259.761 ms - 33
login Usuario: ==> { cuenta: 'sdl', password: '"233423",' }
hay un error Usuario {
```

Figura 177 “El mensaje que se ve en el backend ””.(Datos Obtenidos por el autor en el software nodejs. backend.

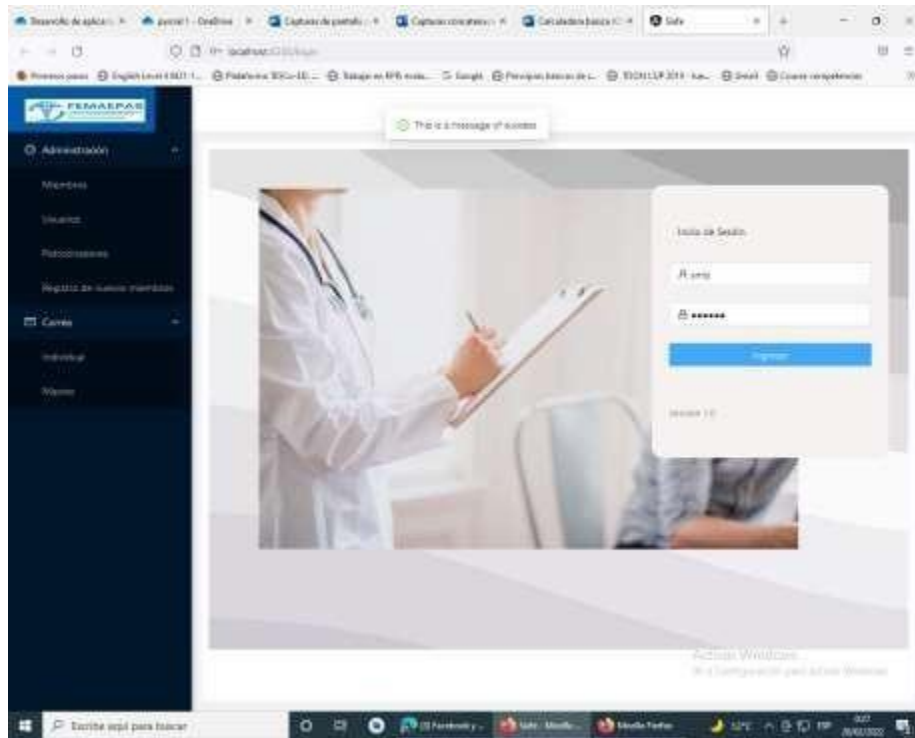


Figura 178 cuando el resultado es correcto”.(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Te da los datos del usuario

```
id: 3,  
nombre: 'esteban',  
ap_paterno: 'michell',  
ap_materno: 'mazon',  
correo: 'sss',  
id_departamento: 1,  
id_rol: 1,  
cuenta: 'smls',  
password: '$2a$10$BY7H0qimyNnmkipqax93sei6uh9uazowFRMUzM0UJGJIxsdr1vdAS',  
fecha_reg: 2022-02-24T16:29:46.000Z,  
id_status: 1
```

Figura 179 “muestra los datos ingresados del usuario” Datos Obtenidos por el autor en el software nodejs. Backend respuesta de datos.

El registro de un cliente

The screenshot shows a web browser window with the URL `localhost:4200/registro`. The page features a dark blue sidebar on the left with the following menu items: 'Administración', 'Usuarios', 'Registro de nuevos miembros' (highlighted), 'Cómo', 'Institución', and 'Inicio'. The main content area is titled 'Datos necesarios para su registro' and includes a membership level selector (Silver, Gold, Platinum) and a grid of input fields for personal and contact information. Below the form is a 'Registrar' button. An 'Activar Windows' watermark is visible in the bottom right corner of the page.

Datos necesarios para su registro			
Silver Gold Platinum			
	Nombre	Apellido	Telefono
Correo	Email	Telefono	Telefono
Estudiante	Universidad	Calle	Calle
Calle	Calle	Calle	Calle
Registrar			

Figura 180 “registro de miembros”.(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

En esta parte se usó el servicio de obtener todos los datos ahora veremos sus funciones de edición y se muestra el numero de miembros que se registraron

Nombre	Teléfono	E-mail	Estado	Profesión	Escolaridad	Acciones
en estado civil	5563433	michigra@gmail.com	mexico	estudiante	Preparatoria	[Edit] [Delete]
Juan Carlos Ramirez	5563433	michigra@gmail.com	mexico	estudiante	Universidad	[Edit] [Delete]

Figura 181 “muestra de la tabla con datos” .(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

El botón editar solo trae los datos de ese renglón para hacer la edición solo se editara correo y el nombre completo el estado y el teléfono

Datos del miembro

Nombre completo: en estado civil Apellido paterno: michigra

Apellido materno: RAMIREZ Teléfono: 5563433

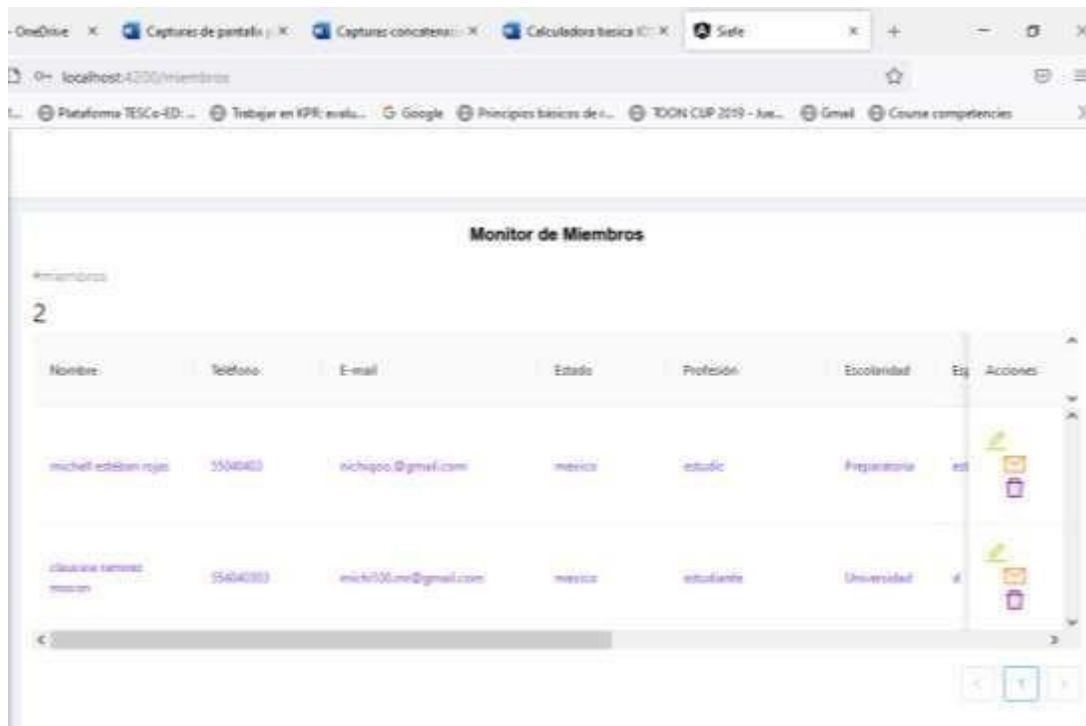
E-mail: michigra@gmail.com Estado: mexico

Guardar cambios

Figura 182 “edición” .(Datos Obtenidos por el autor en el browser

mozilla fierox 4200)

Y al momento de editar puse mi nombre y lo guardamos y así lo pinto en la tabla



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/miembros'. The page title is 'Monitor de Miembros'. Below the title, there is a counter showing '2' members. The main content is a table with the following columns: Nombre, Telefono, E-mail, Estado, Profesión, Escolaridad, and Acciones. The table contains two rows of data.

Nombre	Telefono	E-mail	Estado	Profesión	Escolaridad	Acciones
Michelle Estiben Rojas	55040403	michigoo@gmail.com	México	estudic	Preparatoria	edit
Alfonso Ramos	55040303	amich100me@gmail.com	México	estudiante	Universidad	edit

Figura 183 actualización de la tabla.(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Ingresaremos el usuario como vendedor y el departamento será compras

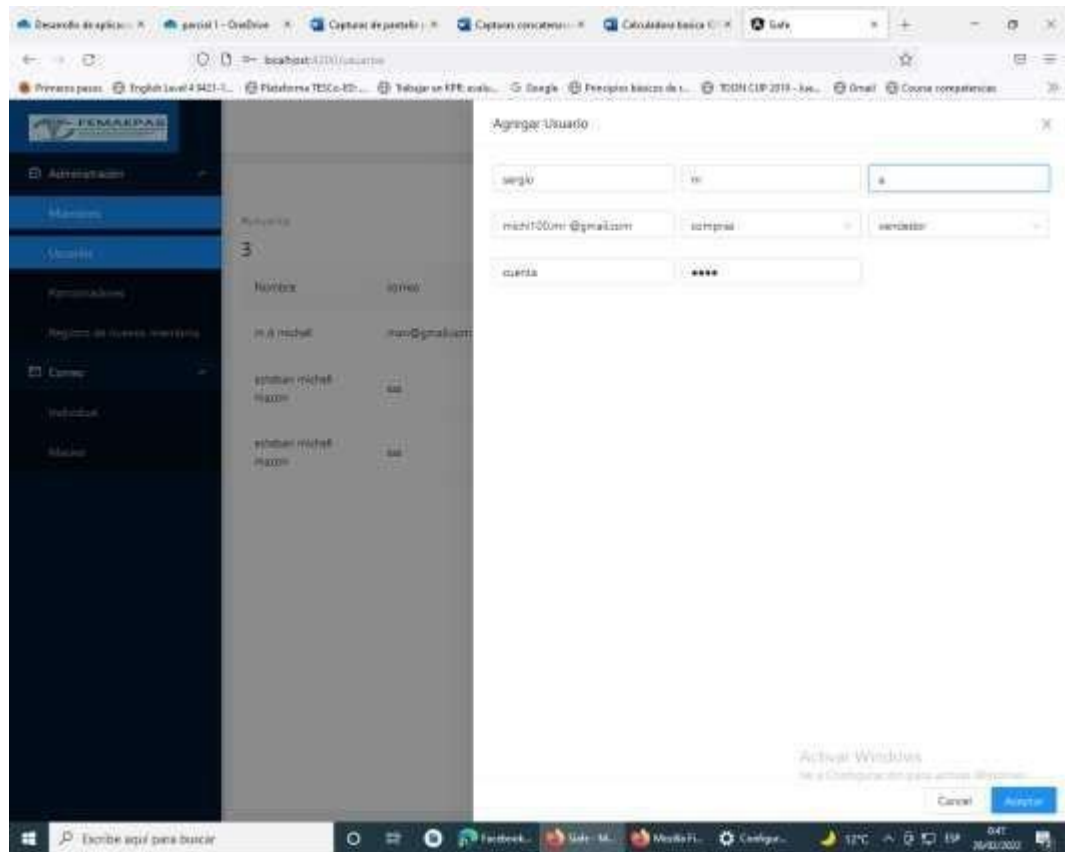


Figura 184 “alta de usuario”. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Al darle el botón nos actualizara la tabla

Monitor de usuarios

Recursos

4

Nombre	correo	departamento	rol	cuenta	Action
m o michell	mich@msd.com	contaduria	Administrador	mason nja	  
eribian michell	asa	contaduria	Administrador	asa	  
eribian michell	asa	contaduria	Administrador	asa	  
jorge m a	mich100.m@gmail.com	compras	vendedor	cuenta	  

Activar Windows
Ve a Configuración para activar Windows

Figura 185 “tabla llena de usuarios” .(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Como la función de editar ya se las mostrare y en esta tabla les mostrare la de correo la función del correo

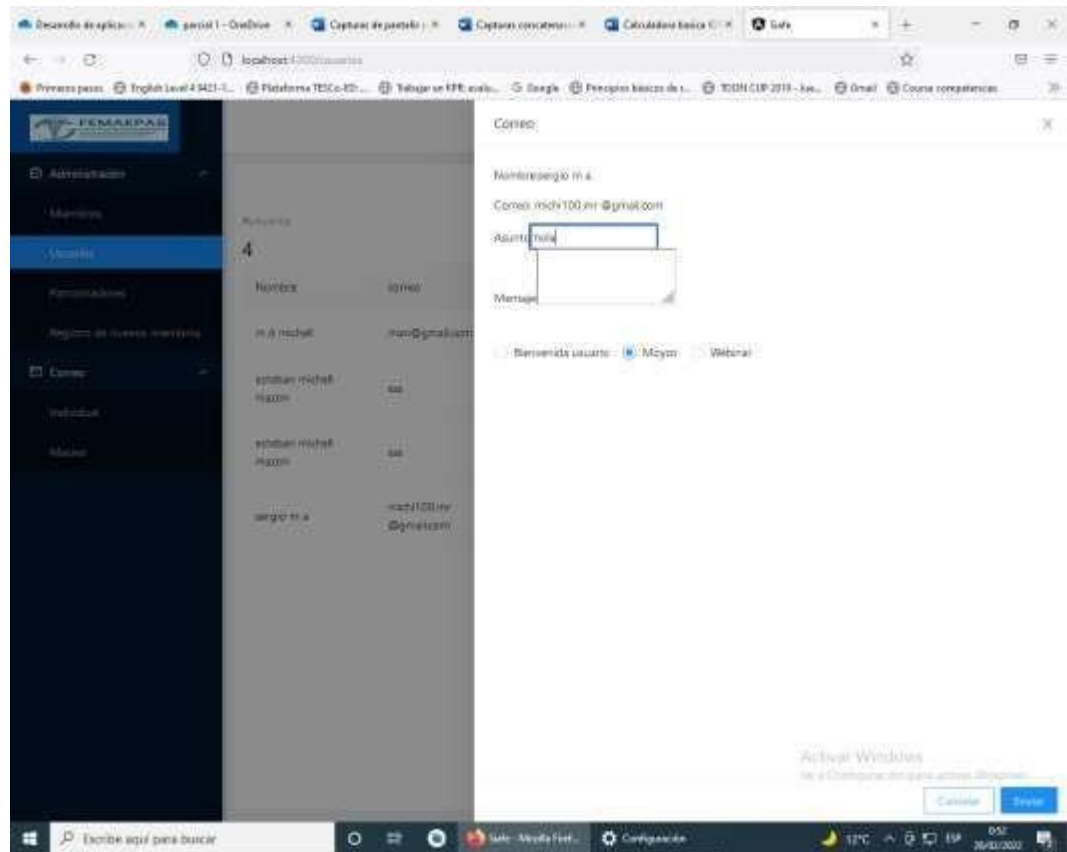


Figura 186 “envío de correo”.(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Y se envía

Después se usara la tabla de patrocinio ya vieron un poco del avance pero les mostraremos la edición y un poco de código se les explicara hay dos tablas en el drawer ya que si recordamos los fundamentos de base de datos un empresa puede tener mas de muchos contactos y muchos domicilio así que primero realizamos el registro se guarda en memoria para poder hacerlo y después de eso se obtiene el id patrocinador y con eso se hace

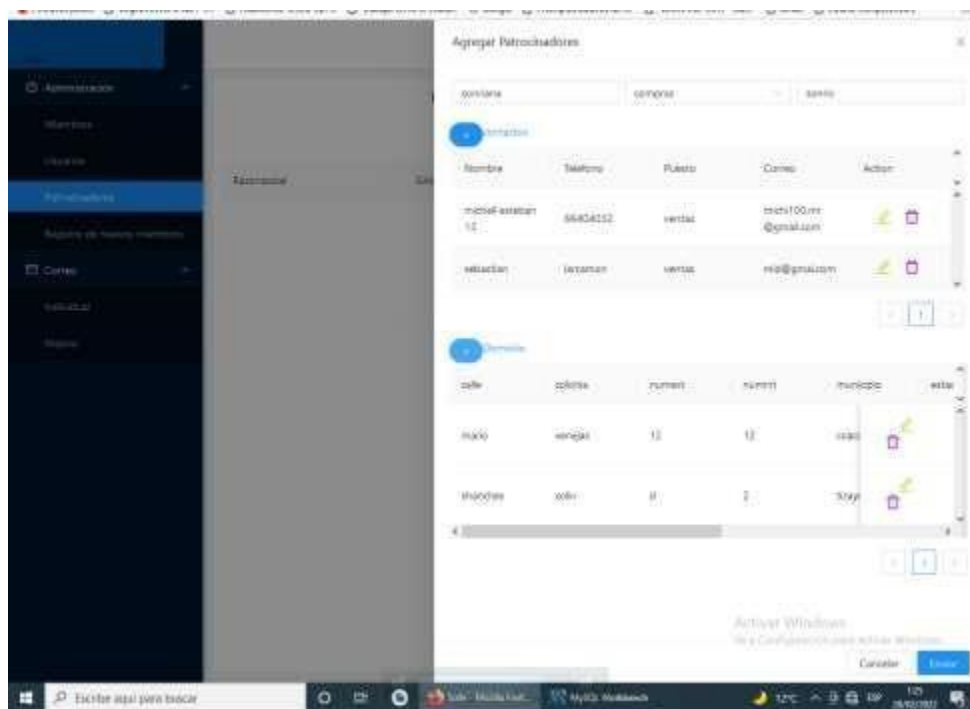


Figura 187 “alta de patrocinado” .(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

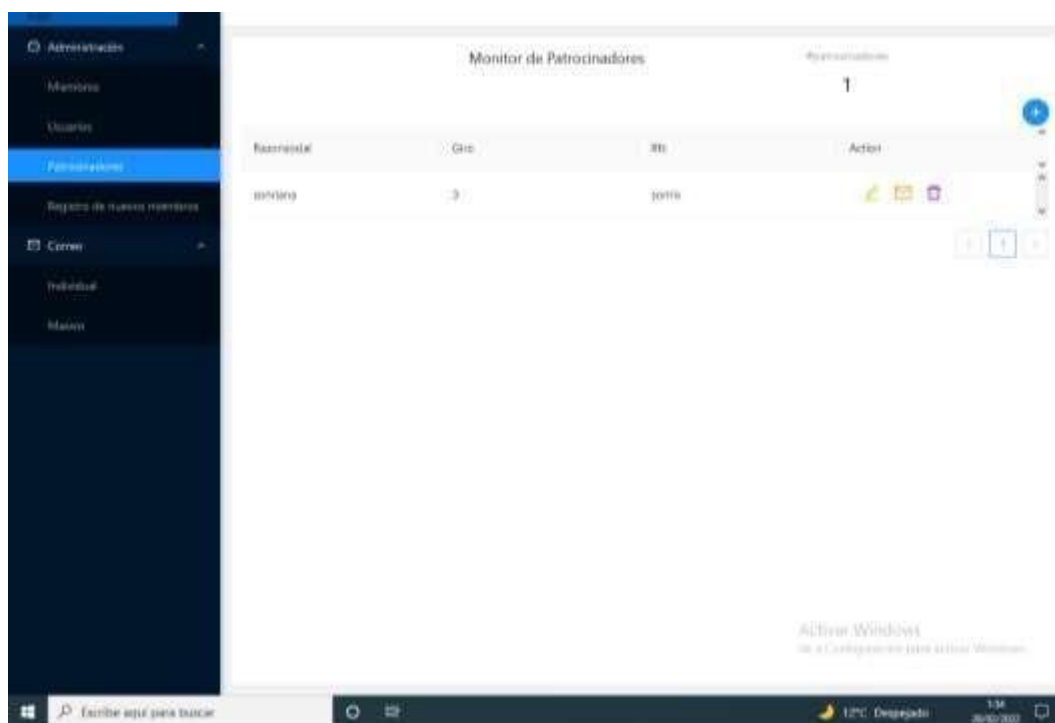


Figura 188 “tabla de patrocinadores” .(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Al darle el boton editar ya te aparecen las tablas correspondientes

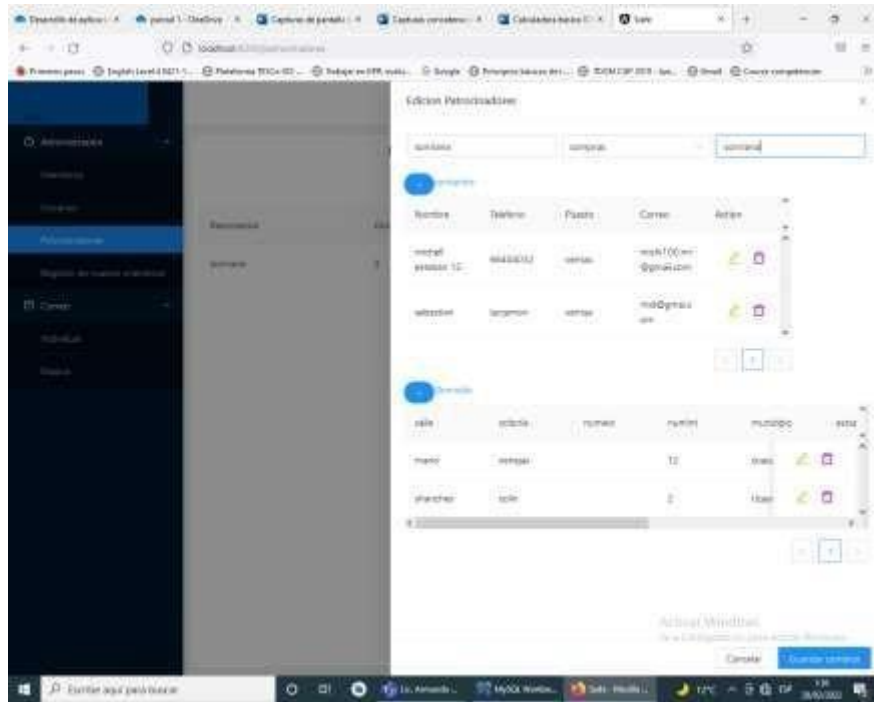


Figura 189 “interfaz de edición” .(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Aquí editaremos un domicilio que le cambiaremos el numero interior y exterior le damos aceptar

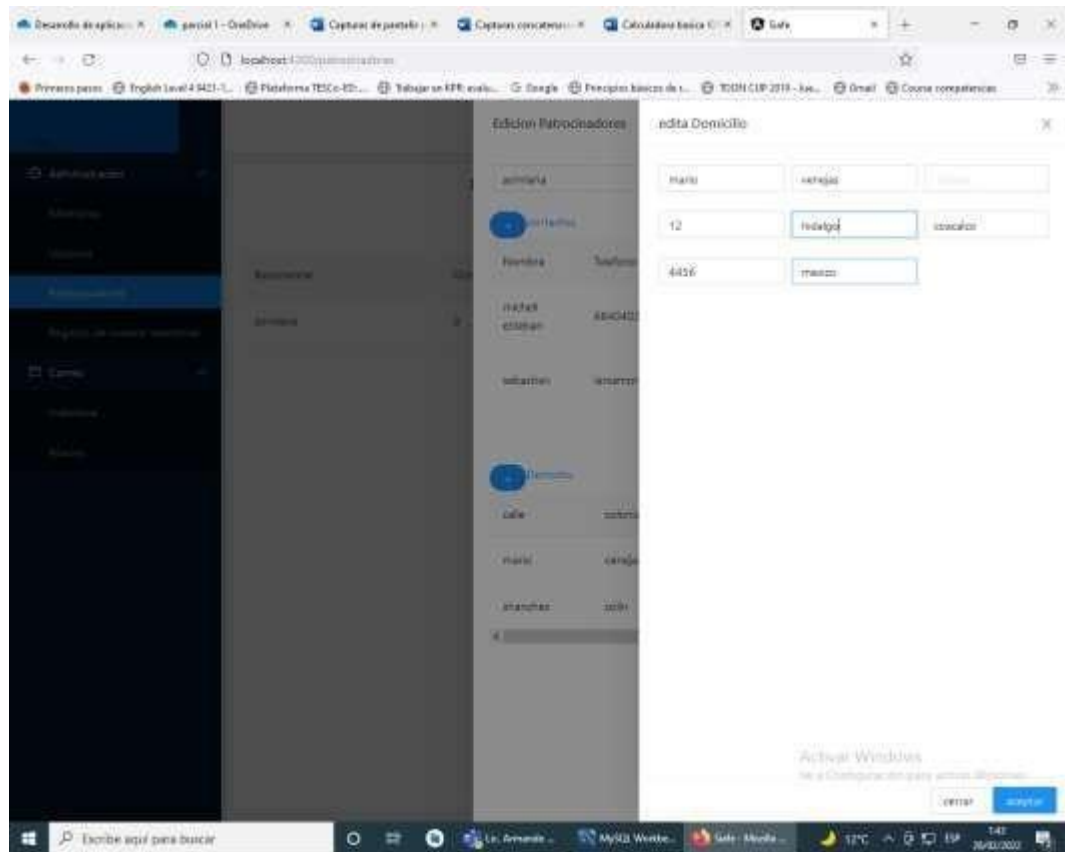


Figura 190 “edición de domicilio”.(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Se abrio y se guardaron los cambio que se hicieron ahra creare uno nuevo

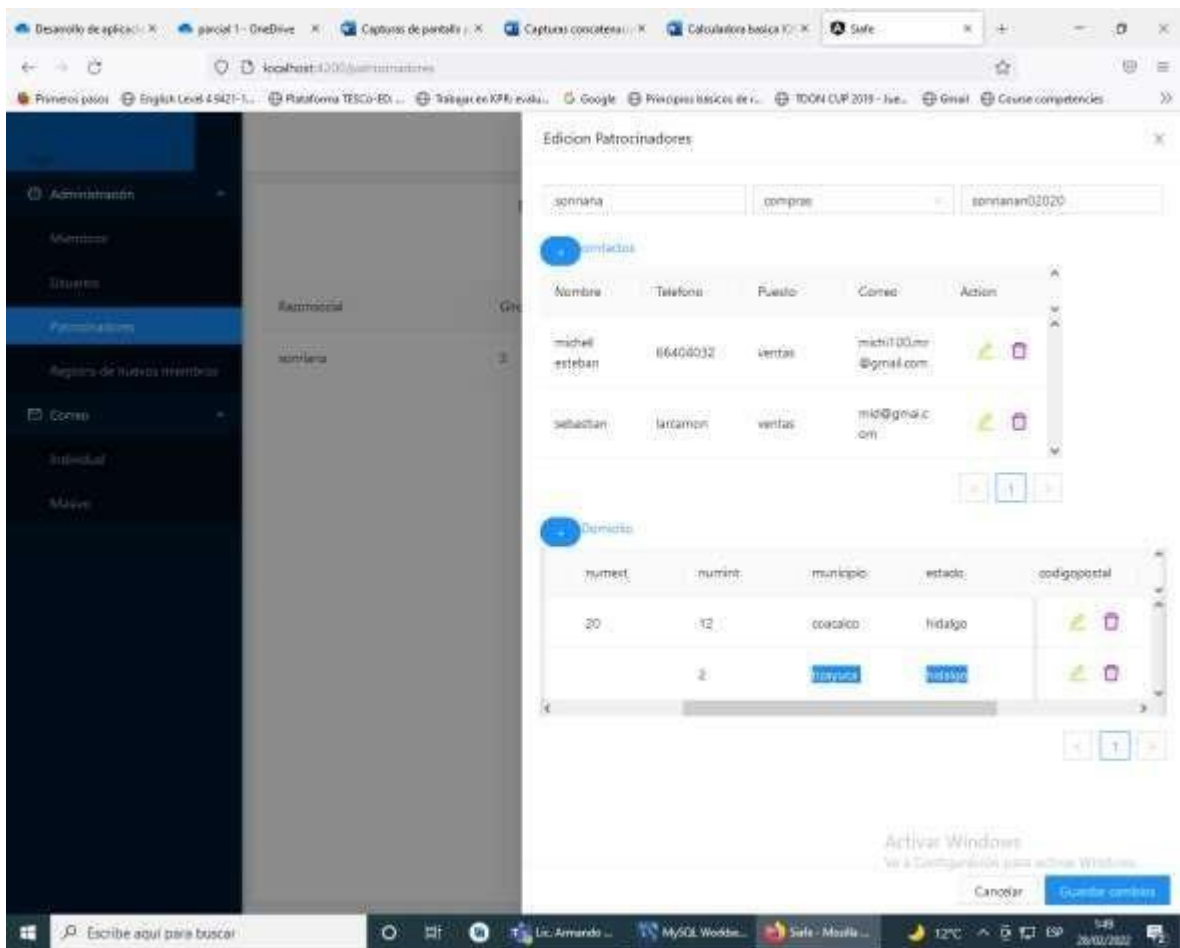


Figura 191 “nuevo patrocinador contacto o domicilio dentro de la edición”. (Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Y al guardar cambios se hara uno nuevo

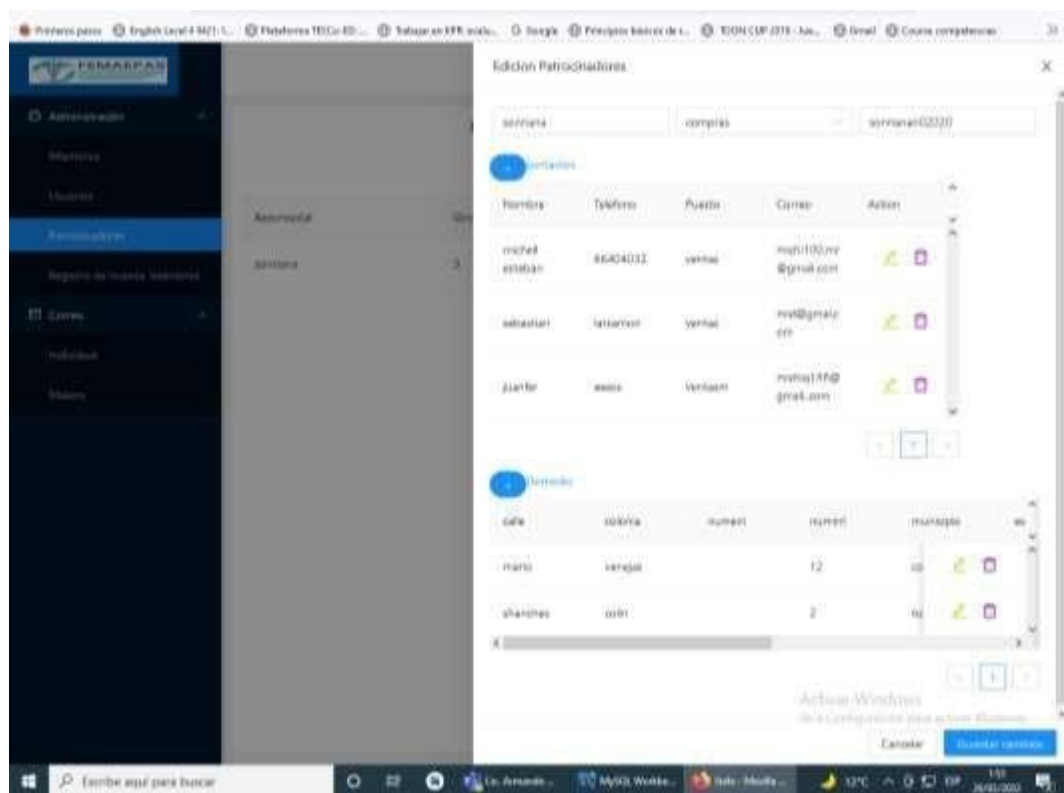


Figura 192 “actualización del patrocinador con sus nuevos contactos y domicilio”. (Datos Obtenidos por el autor en el browser mozilla firefox 4200)

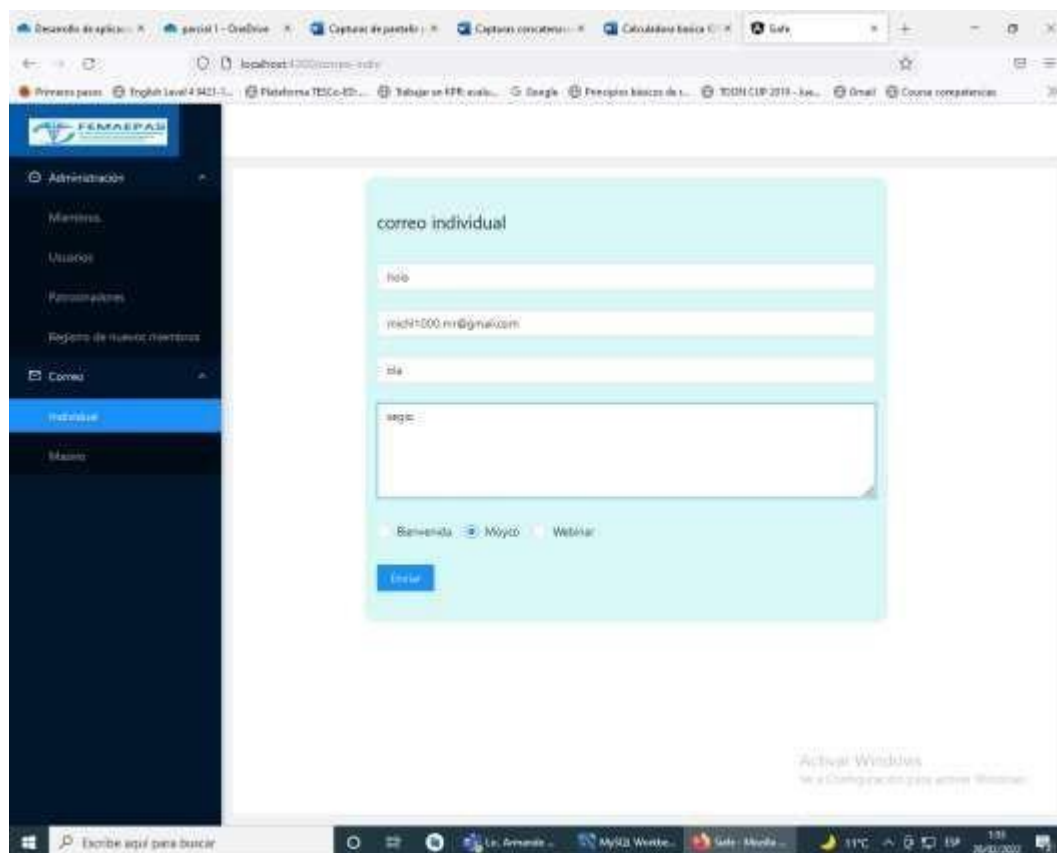


Figura 193 “envío de correo”.(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

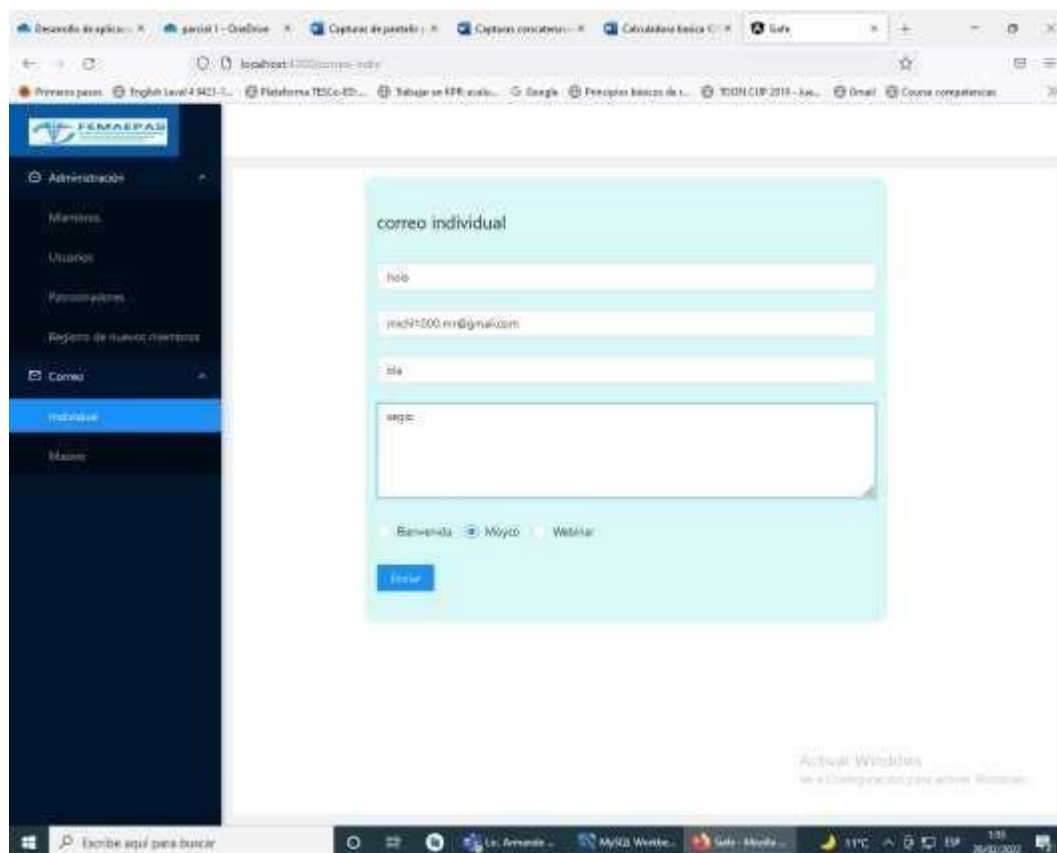


Figura 194 “*envio individual*” .(Datos Obtenidos por el autor en el browser mozilla fierox 4200)

Resultado de correos enviados



Figura 195 “masivo bienvenida” .(Datos Obtenidos por el autor en el browser de gmail)



Figura 196 “patrocinadores moyco”. (Datos Obtenidos por el autor en el browser de gmail)



Figura 197 “individual”. (Datos Obtenidos por el autor en el browser de gmail)

Bienvenido(a) Recibidos X

server@femaepas.com
para mí

mar, 30 nov 2021



Federación Mundial Académica y Empresarial de las Áreas y Profesi salud

Bienvenido(a)

Claudia solar maxon

MEMBRESÍA silver No. 123232

En nombre de FEMAEPAS le doy la bienvenida oficialmente como asociado a nuestra Federación, con gran orgullo y entusiasmo, me permito animarle a portal (www.femaepas.com) y redes sociales, para que conozca los beneficios que le ofrecemos con su Membresía silver, y confiamos que su experiencia satisficció. Nos interesa que participe activamente dentro de esta comunidad a través de sus comentarios, aportaciones y sugerencias, para que contr todos el valor de la interacción global con apoyo de la tecnología digital. Como parte de los beneficios por ser miembro de la Federación, podrá acceder a medicas con contenido de gran valor para consolidar a la profesión, además de los beneficios que nos ofrecen los patrocinadores a través de la Federac

Figura 198 “registro de un miembro”.(Datos Obtenidos
por el autor en el browser de gmail)

Conclusiones

Durante las residencias se adquirieron nuevos conocimientos sobre la programación Web ágil, los cuales permitieron enriquecer y fortalecer nuevas habilidades y estrategias en este rubro, que hicieron que los conocimientos adquiridos se reforzaran y se permitiera manejar distintas perspectivas que motivaron el deseo de seguir aprendiendo y motivaron el interés por buscar e indagar sobre distintas maneras de resolver problemas de varios tipos, agilizando la codificación y estructuración de interfaces mediante un editor de código fuente en el cual se permitió realizar la instalación de un framework llamado An-

gular, el cual dio facilidad para codificar más ágil el código al igual que la estructura HTML y el lenguaje typescript, la librería usada para el presente proyecto fue ngzorro la cual ayudó a implementar en el front-end la forma y la estética del proyecto en lado del backend se usó la typeorm que es un orm(modelo de objetos relacional) el cual permite hacer la conexión y creación de cualquier base de datos dentro del proyecto, al igual que se realizó un servicio mediante node js para poder tener el servidor backend andando y sin problemas al igual que se realiza el método prueba y error para ver la funcionalidad del servidor y como herramienta de apoyo para dicho método se efectuó insomnia que es un modelo de prueba para tecnología rest dentro de ello las tareas mencionadas para backend era hacer la conexión de datos entre el gestor de base de datos y el backend al igual que la información permitida y rutas de conexión de datos en el frontend se llevó a cabo la administración y seguridad de la información así como el control de datos y flujo de información de la empresa y de las membresías

la recomendación que se menciona es hacer un mejor esfuerzo mutuamente, ya que no existen proyectos grandes o chicos solo proyectos, los cuales tenemos que resolver y no solo estancarse con un solo lenguaje de programación, si hay demasiados por conocer y que puede salir en el transcurso del uso del Programa.

Competencias Realizadas

Fundamentos de la Investigación

Conceptos básicos de fundamentos de investigación como proceso de construcción social.

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de investigación obtenemos la siguiente información Identifica conceptos básicos de fundamentos de investigación como proceso de construcción social.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
 - Capacidad de investigación
 - Capacidad de crítica y autocrítica
 - Capacidad de trabajar en equipo
 - Capacidad de comunicación oral y escrita
 - Compromiso ético
 - Capacidad de abstracción, análisis y síntesis
- Habilidad en el uso de Tecnologías de la información y de la comunicación

Herramientas de la comunicación oral y escrita en la investigación

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de investigación obtenemos la siguiente información Aplica herramientas formales de comunicación oral y escrita en la investigación documental, en la elaboración de documentos académicos.

Genéricas:

- Capacidad de abstracción, análisis y síntesis

- Capacidad de aplicar los conocimientos en la práctica
 - Capacidad de comunicación oral y escrita
- Habilidad en el uso de Tecnologías de la información y de la comunicación.
 - Capacidad de investigación
 - Capacidad de crítica y autocrítica
 - Capacidad de trabajar en equipo

Estudio del desarrollo de su profesión y su estado actual

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de investigación obtenemos la siguiente información Analiza el desarrollo de su profesión, para conocer los aspectos sobresalientes en los ámbitos local, nacional e internacional empleando herramientas de investigación científica.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
 - Capacidad de comunicación oral y escrita
- Habilidad en el uso de Tecnologías de la información y de la comunicación.
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
 - Capacidad de investigación
 - Capacidad de crítica y autocrítica
 - Capacidad de trabajar en equipo
 - Capacidad para tomar decisiones
 - Compromiso ético

Proceso de elaboración de una investigación documental

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de investigación obtenemos la siguiente información Aplica métodos y técnicas de investigación documental, de acuerdo con parámetros previamente establecidos.

Genéricas:

- Capacidad de abstracción, análisis y síntesis

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
- Habilidad en el uso de Tecnologías de la información y de la comunicación.
 - Capacidad de investigación
 - Capacidad de crítica y autocrítica
 - Capacidad de trabajar en equipo
 - Compromiso ético
- Capacidad de aprender y actualizarse permanentemente
 - Capacidad de comunicación oral y escrita

Fundamentos de Programación

Fundamentos de programación orientada a objetos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de programación obtenemos la siguiente información Comprende los conceptos de la programación orientada a objetos, para aplicarlos en la resolución de problemas

. Genéricas:

- Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas).
 - Capacidad de análisis y síntesis.
 - Conocimientos básicos de la carrera.
 - Capacidad de comunicación oral y escrita.

Metodología de solución de problemas

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de programación obtenemos la siguiente información Conoce el modelo de las 6'D para la solución de problemas reales.

Genéricas:

- Capacidad de aplicar los conocimientos en la práctica.
 - Capacidad para actuar en nuevas situaciones.
 - Habilidad para trabajar en forma autónoma.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.

Acercamiento a las clases y objetos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamentos de programación obtenemos la siguiente información Conoce y aplica la estructura y formato de las clases y objetos para su modelado.

Genéricas:

- Capacidad creativa.
- Capacidad para identificar, plantear y resolver problemas.
 - Capacidad de análisis y síntesis.
 - Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación las de programación orientada a objetos

Programación orientada a objetos

Entorno de desarrollo

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación orientada a objetos obtenemos la siguiente información Conoce y aplica la estructura de proyectos en un entorno de desarrollo para la solución de problemas.

Genéricas:

- Comunicación oral y escrita
- Habilidades del manejo de la computadora
- Capacidad de aplicar los conocimientos en la práctica
- Habilidad para trabajar en forma autónoma

Clases y objetos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación orientada objetos Comprende y aplica la estructura de clases para la creación de objetos y utiliza clases predefinidas para facilitar el desarrollo de aplicaciones

Genéricas:

- Solución de problemas
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad de aprender
- Capacidad de generar nuevas ideas (creatividad)
- Habilidad para trabajar en forma

Métodos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación orientada objetos Comprende y aplica los diferentes tipos de métodos, tomando en cuenta el ámbito y tiempo de vida de los datos durante la ejecución de un programa.

Genéricas:

- Capacidad de análisis y síntesis
- Habilidad para buscar y analizar información proveniente de fuentes diversas
- Solución de problemas
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad de aprender
- Habilidad para trabajar en forma autónoma

Herencia y polimorfismo

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación orientada objetos Comprende y aplica los conceptos de herencia y polimorfismo en programas que utilicen clases base, clases derivadas, clases abstractas e interfaces.

Genéricas:

- Capacidad de análisis y síntesis
- Habilidad para buscar y analizar información proveniente de fuentes diversas
 - Solución de problemas
- Capacidad de aplicar los conocimientos en la práctica
 - Habilidad para trabajar en forma autónoma
 - Capacidad de aprender

Arreglos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación orientada a objetos. Conoce y aplica programas que implementen el uso de arreglos para reconocerlos como una herramienta para agrupar datos y facilitar la solución de problemas.

Genéricas:

- Capacidad de análisis y síntesis
- Habilidad para buscar y analizar información proveniente de fuentes diversas
 - Solución de problemas
 - Trabajo en equipo
- Capacidad de aplicar los conocimientos en la práctica
 - Capacidad de aprender

Estructura y Organización de Datos

Fundamentos de estructura de datos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para estructura y organización de datos. Identifica las diferentes estructuras de datos, respecto a su implementación a la solución de problemas.

Genéricas:

- Solución de problemas
- Capacidad crítica y autocrítica
- Capacidad de los conocimientos en la práctica
- Capacidad de generar nuevas ideas (creatividad)

Estructuras lineales

Específica(s): mediante el plan de trabajo del TCNM, 2010 para estructura y organización de datos Aplica las principales estructuras de datos lineales en la solución de problemas.

Genéricas:

- Solución de problemas
- Capacidad crítica y autocrítica
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad de generar nuevas ideas (creatividad)

Estructuras no lineales

Específica(s): mediante el plan de trabajo del TCNM, 2010 para estructura y organización de datos Aplica las principales estructuras de datos no lineales en la solución de problemas

Genéricas:

- Solución de problemas
- Capacidad crítica y autocrítica
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad de generar nuevas ideas (creatividad)

Métodos de ordenamiento y búsqueda

Específica(s): mediante el plan de trabajo del TCNM, 2010 para estructura y organización de datos Clasifica y aplica técnicas para facilitar el ordenamiento, búsqueda y recuperación de información en dispositivos de almacenamiento primario y secundario.

Genéricas:

- Solución de problemas
- Capacidad crítica y autocrítica
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad de generar nuevas ideas (creatividad)

Archivos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para estructura y organización de datos Comprende y aplica programas que manipulen archivos de texto y binarios en función de una problemática dada.

Genéricas:

- Capacidad de análisis y síntesis
 - Solución de problemas
- Habilidad para trabajar en forma autónoma
- Capacidad de aplicar los conocimientos en la práctica
 - Capacidad de aprender

FUNDAMENTOS DE BASE DE DATOS

Introducción a las bases de datos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamento de base de datos Conoce y comprende los conceptos básicos de base de datos para proponer soluciones en el tratamiento de información.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes.
 - Capacidad de abstracción, análisis y síntesis
 - Capacidad de comunicación oral y escrita.

Diseño de Bases de Datos con el modelo E-R

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamento de base de datos Conoce y aplica el modelo E-R para el diseño conceptual de bases de datos con el fin de organizar la información y atender necesidades del entorno.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Modelo relacional

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamento de base de datos Conoce y aplica el modelo relacional para la generación de esquemas de base de datos con el fin de organizar la información y atender necesidades del entorno.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Normalización de bases de datos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamento de base de datos Aplica el proceso de normalización al diseño de los esquemas de bases de datos para detectar anomalías y garantizar la integridad de la información.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Álgebra relacional

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamento de base de datos Aplica operadores de álgebra relacional básica y extendida para acceder a la información de base de datos.

Genéricas:

- Capacidad de abstracción análisis y síntesis.
- Capacidad de aplicar los conocimientos en la práctica.
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Introducción al lenguaje SQL

Específica(s): mediante el plan de trabajo del TCNM, 2010 para fundamento de base de datos Aplica los comandos básicos del lenguaje SQL para la definición y manipulación de bases de datos.

Genéricas:

- Capacidad de abstracción análisis y síntesis.
- Capacidad de aplicar los conocimientos en la práctica.
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Taller de Base de datos

El SGBD y el Lenguaje de Definición de Datos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de base de datos Identifica y aplica las herramientas de los SGBD comerciales para crear bases de datos. Crea y aplica el esquema de integridad de las

bases de datos para asegurar la validez de los datos en el procesamiento de la información.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes.
 - Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Lenguaje de Manipulación de Datos Competencias

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de base de datos Aplica las sentencias del lenguaje SQL para manipular la información almacenada en la base de datos.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Administración de base de datos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de base de datos Crea y aplica los esquemas de seguridad y recuperación de bases de datos para garantizar la protección y consistencia en el procesamiento de la información.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Tecnologías de conectividad a base de datos

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de base de datos. Aplica tecnologías emergentes de conectividad bases de datos para la solución de problemas reales.

Genéricas:

- Capacidad de abstracción, análisis y síntesis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Base de datos Distribuidas

Fundamentos de bases de datos distribuidas

Específica(s): mediante el plan de trabajo del TCNM, 2010 para de base de datos distribuidas. Identifica la arquitectura de los sistemas de bases de datos distribuidos, niveles de abstracción y lenguajes para definir arquitecturas de un sistema de gestión de bases de datos.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes.
 - Capacidad de abstracción, análisis y síntesis
 - Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Diseño de bases de datos distribuidas

Específica(s): mediante el plan de trabajo del TCNM, 2010 para de base de datos distribuidas. Aplica el diseño de bases de datos distribuidas incluyendo niveles de transparencia y fragmentación de datos.

Implementa un diccionario de datos para bases de datos distribuidas.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes.
 - Capacidad de abstracción, análisis y síntesis
 - Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas
 - Capacidad de trabajo en equipo

Procesamiento de consultas distribuidas

Específica(s): mediante el plan de trabajo del TCNM, 2010 para de base de datos distribuidas. Crea estrategias de optimización de procesamiento de consultas distribuidas

para la recuperación de datos.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes.
 - Capacidad de abstracción, análisis y síntesis
 - Capacidad de aplicar los conocimientos en la práctica
- Capacidad para identificar, plantear y resolver problemas Capacidad de trabajo en equipo

Manejo de Transacciones Competencias

Específica(s): mediante el plan de trabajo del TCNM, 2010 para base de datos distribuidas. Administra transacciones en sistemas distribuidos para garantizar la integridad y seguridad de datos en una BDD.

Genéricas:

- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes.
 - Capacidad de abstracción, análisis y síntesis
 - Capacidad de aplicar los conocimientos en las prácticas

Ingeniería de software

Introducción a la Ingeniería de Software y Sistemas de Información

Específica(s): mediante el plan de trabajo del TCNM, 2010 para ingeniería de software. Identifica los fundamentos teóricos que integran la ingeniería de software y los sistemas de información con la finalidad de comprender la relación entre sus elementos.

Genéricas:

- Habilidad para buscar y analizar información proveniente de fuentes diversas.
 - Capacidad de análisis y síntesis.
- Habilidades básicas de manejo de la computadora.

Modelos de la Ingeniería de Software

Específica(s): mediante el plan de trabajo del TCNM, 2010 para ingeniería de

software. Identifica los diferentes modelos de desarrollo de software para identificar el más adecuado y diseñar sistemas de información eficientes.

Genéricas:

- Capacidad de análisis y síntesis
- Habilidades básicas de manejo de la computadora.
 - Solución de problemas.
- Capacidad de aplicar conocimientos en la práctica.

Análisis del Proyecto de Software

Específica(s): mediante el plan de trabajo del TCNM, 2010 para ingeniería de software. Planifica actividades con la finalidad de estimar tiempos y costos del proyecto de software. Aplica métricas orientadas al tamaño para medir y controlar los recursos necesarios para el desarrollo del proyecto de software.

Genéricas:

- Capacidad de análisis y síntesis.
- Capacidad de organizar y planificar.
 - Solución de problemas
 - Toma de decisiones.
 - Trabajo en equipo.
- Capacidad de aplicar los conocimientos en la práctica.
- Capacidad de generar nuevas ideas (creatividad)

Análisis y Modelado del Proyecto de Software

Específica(s): mediante el plan de trabajo del TCNM, 2010 para ingeniería de software. Utiliza técnicas de análisis y modelado para construir la arquitectura del sistema mediante el uso de herramientas informáticas específicas.

Genéricas:

- Habilidades básicas de manejo de la computadora.
 - Solución de problemas
 - Trabajo en equipo.
- Capacidad de aplicar los conocimientos en la práctica.
 - Capacidad de generar nuevas ideas(creatividad).

Calidad de Software

Específica(s): mediante el plan de trabajo del TCNM, 2010 para ingeniería de software. Analiza y comprende el enfoque de calidad aplicada al software para determinar los procesos de implementación y aseguramiento de la misma en un entorno globalizado.

Genéricas:

- Capacidad de análisis y síntesis
- Comunicación oral y escrita.
- Habilidad para buscar y analizar información proveniente de fuentes diversas.
 - Solución de problemas.
- Capacidad de aplicar los conocimientos en la práctica.

Taller de investigación I

Tipos de investigación

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de investigación I Distingue los tipos de investigación, así como los métodos y técnicas de cada uno de ellos.

Genéricas:

- Capacidad de investigación
- Habilidad para trabajar en forma autónoma
- Capacidad de abstracción, análisis y síntesis
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
 - Habilidad en el uso de Tecnologías de la información y de la comunicación
 - Capacidad de trabajar en equipo

Estructura del protocolo de Investigación.

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de investigación I Diseña un protocolo de investigación.

Genéricas:

- Capacidad de investigación
- Habilidad para trabajar en forma autónoma
- Capacidad de abstracción, análisis y síntesis
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
 - Habilidad en el uso de Tecnologías de la información y de la comunicación.
 - Capacidad de trabajar en equipo.

Comunicación del protocolo de investigación.

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de investigación I Redacta el informe del protocolo y lo presenta oralmente.

Genéricas:

- Capacidad de investigación
- Capacidad de abstracción, análisis y síntesis
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
 - Habilidad en el uso de Tecnologías de la información y de la comunicación
 - Capacidad de trabajar en equipo.

Taller de investigación II

Evaluación y complementación del protocolo de investigación

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de investigación I Analiza y conforma la actualización del protocolo de investigación para darle seguimiento.

Genéricas:

- Capacidad de investigación
- Habilidad para trabajar en forma autónoma
- Capacidad de abstracción, análisis y síntesis
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
 - Habilidad en el uso de Tecnologías de la información y de la comunicación.
 - Capacidad de trabajar en equipo

Desarrollo de la metodología del proyecto de investigación

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de investigación I Realiza el proyecto de investigación ante diversos escenarios con actitud crítica y constructiva para la solución de problemas relacionados con su campo profesional.

Genéricas:

- Capacidad de investigación
- Habilidad para trabajar en forma autónoma
- Capacidad de abstracción, análisis y síntesis
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
- Habilidad en el uso de Tecnologías de la información y de la comunicación.
- Capacidad de trabajar en equipo

Presentación del Informe de investigación.

Específica(s): mediante el plan de trabajo del TCNM, 2010 para taller de investigación I Desarrolla la presentación escrita y oral del proyecto mediante el uso de TIC's para su argumentación profesional en plenaria o sínodo.

Genéricas:

- Capacidad de investigación
- Habilidad para trabajar en forma autónoma
- Capacidad de abstracción, análisis y síntesis
- Habilidades para buscar, procesar y analizar información procedente de diversas fuentes
- Habilidad en el uso de Tecnologías de la información y de la comunicación. Capacidad de trabajar en equipo

Programación Web

HTML, XML y CSS

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación web. Conoce y aplica las herramientas para construir páginas web, considerando la interfaz gráfica de usuario, un lenguaje de marcado y de presentación.

Genéricas:

- Comunicación oral y escrita.
- Habilidad para buscar, analizar, clasificar y sintetizar información proveniente de fuentes diversas.
 - Capacidad de trabajar en equipo
- Capacidad de aplicar los conocimientos en la práctica

Programación de Lado del Cliente

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación web
 Conoce y aplica un lenguaje de programación del lado del cliente, para la construcción de aplicaciones web dinámicas, considerando Frameworks ya existentes.

Genéricas:

- Capacidad de análisis y síntesis.
- Comunicación oral y escrita.
- Habilidad para buscar, analizar, clasificar y sintetizar información proveniente de fuentes diversas.
 - Solución de problemas.
- Capacidad de trabajar en equipo

Programación del lado del servidor

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación web
 Conoce y aplica un lenguaje de programación del lado del servidor, para la construcción de aplicaciones web dinámicas, considerando su conectividad a orígenes de datos.

Genéricas:

- Capacidad de análisis y síntesis.
- Comunicación oral y escrita.
- Habilidad para buscar, analizar, clasificar y sintetizar información proveniente de fuentes diversas.
 - Solución de problemas.
- Capacidad de trabajar en equipo
- Capacidad de aplicar los conocimientos en la práctica
 - Capacidad de aprender

Cómputo en la nube y servicios

Específica(s): mediante el plan de trabajo del TCNM, 2010 para programación web
Conoce y aplica los tipos de servicios, para lograr interconectividad entre aplicaciones,
considerando cómputo en la nube.

Genéricas:

- Capacidad de análisis y síntesis.
- Habilidad para buscar, analizar, clasificar y sintetizar información proveniente de fuentes diversas.
 - Solución de problemas.
 - Toma de decisiones.
 - Capacidad crítica y autocrítica
 - Capacidad de trabajar en equipo
- Capacidad de aplicar los conocimientos en la práctica
 - Capacidad de aprender
 - Búsqueda del logro

Bibliografía:

Angular (s.f.). "Que es angular "Obtenido de angular : <https://angular.io/>

Argote Sevillano, R. (9-may-2019). "Características de la programación reactiva en typescript."

Azutare, c. (s.f.). "Desarrollo web agil en angular".

Desarrolloweb.com.(s.f.)."Que es rest." Obtenido de
<https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html>

Descripcion de director de sistemas. (2021)."Funciones de un director de sistemas"

Obtenido

de_ [https://www.bing.com/search?Q=que+hace+un+director+de+sistemas&FOR](https://www.bing.com/search?Q=que+hace+un+director+de+sistemas&FORM=AWRE)
M=AWRE

Descripcion de gerente administrativo. (s.f.)."Funciones de un gerente administrativo"

Obtenido de <https://mx.talent.com/>

Descripción de desarrollo de software. (2018) "que es un desarrollador de software"
Obtenido de <https://neuvoo.com.mx/neuvoopedia/es/desarrollador-de-software/>

Eduardo Vinicio Pico Gutiérrez, S. C. (2018). "El software contable como herramienta técnica en las microempresas de la provincia de Santa Elena, Ecuador.
Revistakillkana."

Entorno cliente servidor. (s.f.). "Cliente servidor" Obtenido de
<https://es.ccm.net/contents/148-entorno-cliente-servidor>

Gomez, A. P. (2018). " *Fundamento de la gestión de datos.* Area de innovación y desarrollo S.L".

Haro, E. (2019). " *Desarrollo backend para aplicaciones web, Servicios Web Restful: Node.js vs Spring Boot.* Revista Ibérica de Sistemas e Tecnologías de Información".

MANUEL, J. (20 de septiembre de 2010). "¿Qué son los servidores web y por qué son necesarios?" Obtenido de <https://duplika.com/blog/que-son-los-servidores-web-y-por-que-son-necesarios/>

Mendoza, A. (2018). " *Base de datos*".

Ng-zorro. (s.f.). " que es ng-zorro" Obtenido de <https://ng-zorro.gitee.io/>

Node.js. (2022). "que es el node js" Obtenido de node.js: <https://nodejs.org/es/>

ORACLE. (s.f.). " *QUE ES UNA BASE DE DATOS.*" Obtenido de 2019:
<https://www.oracle.com/mx/database/what-is-database/>

Puciarelli, L. (2018). "Instalación Arquitectura node npm. Users".

SERVERNET. (s.f.). " *Backend y frontend*" Obtenido de
<https://www.servnet.mx/blog/backend-y-frontend-partes-fundamentales-de-la-programación-de-una-aplicación-web>

Software erp. ((28 noviembre 2014)). "Createspace Independent Publishing Platform."

Sosa Quispe, O. N. (2019). " *Determinación de la Tecnología Framework (Angular js, node js y sencha ext js) para el desarrollo de una aplicación web que permita la optimización de los procesos de abastecimiento de la municipalidad provincial de bagua grande.* Chimera Software."

TCNM. (2010). " *Plan de Trabajo TIC 2010-2025.*" Obtenido de
<http://www.ittizimin.edu.mx/ingenieria-en-tics>

Tejedor, B. R. (s.f.). " *Que es un Smtip*". Obtenido de <https://es.mailjet.com/blog/news/servidor-smtp/>

