

Marina Rent



Realizado por:

Manuel González Pérez

Fecha:

11/03/2025.

ÍNDICE DE CONTENIDO.

1.	Introducción.....	15
2.	Objetivos y Requisitos.....	15
	→Objetivos.	15
	→Requisitos.	15
3.	Estudio Previo.	16
	→Hacerlo con PHP puro.....	16
	→Hacerlo con React y Laravel.	16
	→Hacerlo con React (la página) y PHP la API.....	16
4.	Plan de Trabajo	17
5.	Diseño.	18
	→Diseño Base de Datos.....	18
	Entidades.....	18
	Relaciones.	19
	→Tablas y Normalización.....	20
	→Diagrama Relacional.....	21
	→Diseño General.	22
	→Diseño Detallado.	23
	→Boceto Aplicación.	24
6.	Implementación.....	28
	Servidor.....	28
	-Api.php.....	28
	- Modelo usuario.php.....	31
	- Modelo publicacion.php	32
	- Modelo reservas.php	32
	- Migration usuario.php	33
	- Migration publicacion.php.....	34
	- Migration reservas.php.....	34
	- Factory usuario.php	35
	- Factory publicacion.php.....	35
	- Factory reserva.php	36

-ImageController.php.....	36
-UsuarioController.php	37
-PublicacionController.php	43
-ReservaController.php	50
-Vista Email Personalizado	58
-Vista Password Reset Success.....	58
-Vista Reset Password.....	59
Cliente	61
ENRUTADOR – App.jsx	61
Login.jsx.....	64
ResetPass.jsx.....	68
Register.jsx	72
Home.jsx	79
Informativos.jsx.....	83
Alquilables.jsx.....	88
Mis Reservas.jsx	95
Carrito.jsx	97
ai.jsx	103
Admin.jsx.....	105
Perfil.jsx.....	113
Mostrador.jsx	120
buscador.jsx	128
Calendar.jsx.....	131
Days-details-modal.jsx	136
Footer.jsx	139
MarbellaMap.jxs	141
Navbar.jsx.....	142
Person-chooser.jsx	144
Time-slider.jsx	145
Usuarios.jsx	148

Modal-usuarios.jsx	154
Publicacion.jsx.....	158
Modal-publicaciones.jsx.....	163
Reservas.jsx.....	170
Modal-reservas.jsx	175
Modal-usuarios-perfil.jsx	183
7. Despliegue	187
Servidor.....	187
Cliente	192
8. Seguridad	194
9. Pruebas	195
10. Recursos	200
Recursos Hardware.....	200
Recursos Software	200
Recursos Personales	200
Presupuesto	200
11. Conclusiones.....	201
Consecución de los Objetivos.	201
Problemas encontrados.....	201
Mejoras	202
12. Bibliografía	202
Anexos.....	203
Manual de Usuario.....	203
Iniciar Sesión y Registrarte.....	203
Consultar publicaciones y añadir al carrito.....	205
Realizar reservas y consultar las reservas.....	210
Restaurar Contraseña	212
Charlar con la IA *NEW*	214

INDICE DE ILUSTRACIONES

Ilustración 1 Logo de React.....	16
Ilustración 2 Plan de Trabajo	18
Ilustración 3 Diseño E.R	19
Ilustración 4 Modelo Relacional	21
Ilustración 5 Diagrama de Flujo de Datos.....	22
Ilustración 6 Diseño a Detalle	23
Ilustración 7 Página Login	24
Ilustración 8 Página de Register	24
Ilustración 9 Página de Olvido de Contraseña.....	25
Ilustración 10 Página de Inicio	25
Ilustración 11 Página de Informativos/Alquilables.....	26
Ilustración 12 Página vista Alquiler/Informativo.	26
Ilustración 13 Mis Reservas	27
Ilustración 14 Mi Perfil.....	27
Ilustración 15 EndPoints Sanctum I	28
Ilustración 16 Middleware isAdmin	29
Ilustración 17 EndPoints Santum II	29
Ilustración 18 Endpoints Públicos I	30
Ilustración 19 Endpoints Públicos II	30
Ilustración 20 Modelo usuario.php	31
Ilustración 21 Modelo publicacion.php	32
Ilustración 22 Modelo reservas.php	32
Ilustración 23 Migracion usuario.php.....	33
Ilustración 24 Migracion publicación.php	34
Ilustración 25 Migracion reservas.php	34
Ilustración 26 Factory de usuario.php	35
Ilustración 27 Factory de publicación.php.....	35
Ilustración 28 Factory de reserva.php	36
Ilustración 29 Controller ImageController.php	36
Ilustración 30 Index de usuarios	37
Ilustración 31 Paginar Usuarios	37
Ilustración 32 Is Admin	38
Ilustración 33 Obtener Usuario Autenticado.....	38
Ilustración 34 Store de Usuarios.....	39
Ilustración 35 Obtener Datos Usuario Autenticado	40
Ilustración 36 Enviar correo personalizado	40
Ilustración 37 retornar usuario por id	41

Ilustración 38 actualizar usuario.....	41
Ilustración 39 Delete usuario.....	42
Ilustración 40 Show Publicaciones	43
Ilustración 41 Obtener Paginadas.....	43
Ilustración 42 Obtener Alquilables Paginadas	44
Ilustración 43 Obtener Informativos Paginadas	44
Ilustración 44 Obtener Informativos	45
Ilustración 45 Obtener Alquilables	45
Ilustración 46 Store de Publicación I	46
Ilustración 47 Store de Publicación II	46
Ilustración 48 Store de Publicación III	46
Ilustración 49 Publicaciones Aleatorias	47
Ilustración 50 Update de Publicaciones I	47
Ilustración 51 Update de Publicaciones II	48
Ilustración 52 Update de Publicaciones III	48
Ilustración 53 Show de Publicaciones	48
Ilustración 54 Delete de Publicaciones.....	49
Ilustración 55 Show Reservas	50
Ilustración 56 Reservas Paginadas.....	50
Ilustración 57 Reservas por Usuario	51
Ilustración 58 Actualizar Fecha Publicación I.....	51
Ilustración 59 Actualizar Fecha Publicación II.....	52
Ilustración 60 Obtener reservas por usuario	52
Ilustración 61 Store de Reservas I.....	53
Ilustración 62 Store de Reservas II	53
Ilustración 63 Capacidad Disponible de las Publicaciones	54
Ilustración 64 Disponibilidad de las Reservas.....	54
Ilustración 65 Show de Reservas	55
Ilustración 66 Update de Reservas	55
Ilustración 67 Intercambiar Fechas I	56
Ilustración 68 Intercambiar Fechas II	56
Ilustración 69 Obtener Reservas Detalladas.....	57
Ilustración 70 Delete de Reservas	57
Ilustración 71 emailpersonalizado.blade.php	58
Ilustración 72 password-reset-success.blade.php.....	58
Ilustración 73 reset.blade.php I.....	59
Ilustración 74 reset.blade.php II.....	59
Ilustración 75 Fichero Env I.....	60
Ilustración 76 Fichero Env II.....	60
Ilustración 77 App jsx I.....	61

Ilustración 78 App jsx II.....	62
Ilustración 79 App jsx III.....	62
Ilustración 80 App jsx IV.....	63
Ilustración 81 App jsx V.....	63
Ilustración 82 App jsx VI.....	64
Ilustración 83 Vista de Login.....	64
Ilustración 84 Login jsx I.....	65
Ilustración 85 Login jsx II.....	65
Ilustración 86 Login jsx III.....	66
Ilustración 87 Login jsx IV	66
Ilustración 88 Login jsx V	67
Ilustración 89 Login jsx VI	67
Ilustración 90 Reset Pass I	
ilustración 91 Vista Reset Pass II	68
Ilustración 92 Reset Pass III	68
Ilustración 93 Reset Pass jsx I	69
Ilustración 94 Reset Pass jsx II	70
Ilustración 95 Reset Pass jsx III	70
Ilustración 96 Reset Pass IV	71
Ilustración 97 Reset Pass V	71
Ilustración 98 Reset Pass VI	72
Ilustración 99 Register I	72
Ilustración 100 Register II	73
Ilustración 101 Register III	73
Ilustración 102 Register jsx I	74
Ilustración 103 Register jsx II	74
Ilustración 104 Register jsx III	75
Ilustración 105 Register jsx IV	75
Ilustración 106 Register jsx V	76
Ilustración 107 Register jsx VI	76
Ilustración 108 Register jsx VII	77
Ilustración 109 Register jsx VIII	77
Ilustración 110 Register IX	78
Ilustración 111 Register X	78
Ilustración 112 Home I.....	79
Ilustración 113 Home II.....	79
Ilustración 114 Home III.....	79
Ilustración 115 Home IV	79

Ilustración 116 Home jsx I.....	80
Ilustración 117 Home jsx II.....	80
Ilustración 118 Home jsx III	81
Ilustración 119 Home jsx IV	81
Ilustración 120 Home jsx V	82
Ilustración 121 Home jsx VI	82
Ilustración 122 Informativo I	83
Ilustración 123 Informativo II	83
Ilustración 124 Informativo III	84
Ilustración 125 Informativo IV	84
Ilustración 126 Informativos jsx I.....	84
Ilustración 127 Informativos jsx II.....	85
Ilustración 128 Informativos jsx III.....	85
Ilustración 129 Informativos jsx IV	86
Ilustración 130 Informativos jsx V	86
Ilustración 131 Informativos jsx VI	87
Ilustración 132 Informativos jsx VII	87
Ilustración 133 Informativos jsx VIII	88
Ilustración 134 Informativos jsx IX.....	88
Ilustración 135 Alquilables I.....	89
Ilustración 136 Alquilables II.....	89
Ilustración 137 Alquilables III.....	90
Ilustración 138 Alquilables jsx I.....	90
Ilustración 139 Alquilables jsx II.....	91
Ilustración 140 Alquilables jsx III.....	91
Ilustración 141 Alquilables jsx IV	92
Ilustración 142 Alquilables jsx V	92
Ilustración 143 Alquilables jsx VI	93
Ilustración 144 Alquilables jsx V	93
Ilustración 145 Alquilables jsx VI	94
Ilustración 146 Alquilables jsx VII	94
Ilustración 147 Mis Reservas I	95
Ilustración 148 Mis Reservas II	95
Ilustración 149 Mis Reservas III	96
Ilustración 150 Mis reservas IV	96
Ilustración 151 Reservas jsx.....	97
Ilustración 152 Carrito I	98
Ilustración 153 Carrito II	98
Ilustración 154 Carrito III	98
Ilustración 155 Carrito jsx I	99

Ilustración 156 Carrito jsx II	99
Ilustración 157 Carrito jsx III	100
Ilustración 158 Carrito jsx IV.....	100
Ilustración 159 Carrito jsx V.....	101
Ilustración 160 Carrito jsx VI.....	101
Ilustración 161 Carrito jsx VII.....	102
Ilustración 162 Carrito jsx VIII.....	102
Ilustración 163 IA Chatbox.....	103
Ilustración 164 ai jsx I	103
Ilustración 165 ai jsx II	104
Ilustración 166 ai jsx III	104
Ilustración 167 ai jsx IV	105
Ilustración 168 ai jsx V	105
Ilustración 169 Admin Publicaciones I.....	106
Ilustración 170 Admin Publicaciones II.....	106
Ilustración 171 Admin Publicaciones III – Editar	107
Ilustración 172 Admin Publicaciones IV – Eliminar	108
Ilustración 173 Admin Publicaciones V – Añadir	108
Ilustración 174 Admin Usuarios I.....	109
Ilustración 175 Admin Usuarios II.....	109
Ilustración 176 Admin Usuarios III – Editar	109
Ilustración 177 Admin Usuarios IV – Eliminar	110
Ilustración 178 Admin Usuarios V – Añadir	110
Ilustración 179 Admin Reservas I.....	111
Ilustración 180 Admin Reservas II.....	111
Ilustración 181 Admin Reservas III – Editar	111
Ilustración 182 Admin Reservas IV - Eliminar	112
Ilustración 183 Admin jsx I.....	112
Ilustración 184 Admin jsx II.....	113
Ilustración 185 Admin I.....	113
Ilustración 186 Admin II.....	114
Ilustración 187 Admin III.....	114
Ilustración 188 Perfil jsx I.....	115
Ilustración 189 Perfil jsx II.....	115
Ilustración 190 Perfil jsx III.....	116
Ilustración 191 Perfil jsx IV.....	116
Ilustración 192 Perfil jsx V.....	117
Ilustración 193 Perfil jsx VI.....	117

Ilustración 194 Perfil jsx VII.....	118
Ilustración 195 Perfil jsx VIII.....	118
Ilustración 196 Perfil jsx IX.....	119
Ilustración 197 Mostrador Informativos I.....	120
Ilustración 198 Mostrador Informativos II	120
Ilustración 199 Mostrador Alquilables I	121
Ilustración 200 Mostrador Alquilables II	121
Ilustración 201 Mostrador Alquilables III	122
Ilustración 202 Mostrador Alquilables IV	122
Ilustración 203 Mostrador jsx I	123
Ilustración 204 Mostrador jsx II.....	123
Ilustración 205 Mostrador jsx III	124
Ilustración 206 Mostrador jsx IV.....	124
Ilustración 207 Mostrador jsx V.....	125
Ilustración 208 Mostrador jsx VI.....	125
Ilustración 209 Mostrador jsx VII.....	126
Ilustración 210 Mostador jsx VII	126
Ilustración 211 Mostrador jsx VIII.....	127
Ilustración 212 Mostrador jsx IX.....	127
Ilustración 213 Mostrador jsx X.....	128
Ilustración 214 Buscador	128
Ilustración 215 Buscador jsx I	129
Ilustración 216 Buscador jsx II	129
Ilustración 217 Buscador jsx III	130
Ilustración 218 Buscador jsx IV	130
Ilustración 219 Buscador jsx V	131
Ilustración 220 Calendario	131
Ilustración 221 Calendario jsx I.....	132
Ilustración 222 Calendario jsx II.....	132
Ilustración 223 Calendario jsx III.....	133
Ilustración 224 Calendario jsx IV.....	134
Ilustración 225 Calendario jsx V.....	134
Ilustración 226 Calendario jsx VI.....	135
Ilustración 227 Calendario jsx VII.....	135
Ilustración 228 Calendario jsx VIII	136
Ilustración 229 Days-details-modal	136
Ilustración 230 days-details-modal I.....	137
Ilustración 231 days-details-modal II.....	137
Ilustración 232 days-details-modal III.....	138
Ilustración 233 days-details-modal IV	138

Ilustración 234 days-details-modal V	139
Ilustración 235 Footer.....	139
Ilustración 236 Footer jsx I.....	139
Ilustración 237 Footer jsx II.....	140
Ilustración 238 Footer jsx III.....	140
Ilustración 239 Footer jxs IV	141
Ilustración 240 MarbellaMap	141
Ilustración 241 MarbellaMap jsx	141
Ilustración 242 Navbar.....	142
Ilustración 243 Navbar jsx I.....	142
Ilustración 244 Navbar jsx II.....	142
Ilustración 245 Navbar jsx III.....	143
Ilustración 246 Navbar jsx IV	143
Ilustración 247 Navbar jsx V	144
Ilustración 248 Person-chooser jsx I	144
Ilustración 249 Person-chooser jsx II	144
Ilustración 250 Person-chooser jsx III	145
Ilustración 251 Time-slider	145
Ilustración 252 Time-slider jsx I	146
Ilustración 253 Time-slider jsx II	146
Ilustración 254 Time-slider jsx III	147
Ilustración 255 Time-slider jsx IV	147
Ilustración 256 Usuarios	148
Ilustración 257 Usuarios jsx I	148
Ilustración 258 Usuarios jsx II	149
Ilustración 259 Usuarios jsx III	150
Ilustración 260 Usuarios jsx IV	151
Ilustración 261 Usuarios jsx V	151
Ilustración 262 Usuarios jsx VI	152
Ilustración 263 Usuarios jsx VII	152
Ilustración 264 Usuarios jsx VIII	153
Ilustración 265 IX	153
Ilustración 266 Usuarios jsx X	154
Ilustración 267 Modal-usuarios	154
Ilustración 268 Modal-Usuarios jsx I	155
Ilustración 269 Modal-Usuarios jsx II	155
Ilustración 270 Modal-Usuarios jsx III	156
Ilustración 271 Modal-Usuarios jsx IV	156

Ilustración 272 Modal-Usuarios jsx V	157
Ilustración 273 Modal-Usuarios jsx VI	157
Ilustración 274 Publicacion	158
Ilustración 275 Publicacion jsx I	158
Ilustración 276 Publicacion jsx II	159
Ilustración 277 Publicacion jsx III	159
Ilustración 278 Publicacion jsx IV	160
Ilustración 279 Publicacion jsx V	160
Ilustración 280 Publicacion jsx VI	161
Ilustración 281 Publicacion jsx VII	161
Ilustración 282 Publicacion jsx VIII	162
Ilustración 283 Publicacion jsx IX	162
Ilustración 284 Modal-Publicaciones	163
Ilustración 285 Modal-publicaciones jsx I	163
Ilustración 286 Modal-publicaciones jsx II	164
Ilustración 287 Modal-publicaciones jsx III	164
Ilustración 288 Modal-publicaciones jsx IV	165
Ilustración 289 Modal-publicaciones jsx V	165
Ilustración 290 Modal-publicaciones jsx VI	166
Ilustración 291 Modal-publicaciones jsx VII	166
Ilustración 292 Modal-publicaciones jsx VIII	167
Ilustración 293 Modal-publicaciones jsx IX	167
Ilustración 294 Modal-publicaciones jsx X	168
Ilustración 295 Modal-publicaciones jsx XI	168
Ilustración 296 Modal-publicaciones jsx XII	169
Ilustración 297 Modal-publicaciones jsx XIII	169
Ilustración 298 Reservas	170
Ilustración 299 Reservas jsx I	170
Ilustración 300 Reservas jsx II	171
Ilustración 301 Reservas jsx III	171
Ilustración 302 Reservas jsx IV	172
Ilustración 303 Reservas jsx V	172
Ilustración 304 Reservas jsx VI	173
Ilustración 305 Reservas jsx VII	173
Ilustración 306 Reservas jsx VIII	174
Ilustración 307 Reservas jsx IX	174
Ilustración 308 Reservas jsx X	175
Ilustración 309 Modal-reservas	175
Ilustración 310 Modal-reservas jsx I	176
Ilustración 311 Modal-reservas jsx II	176

Ilustración 312 Modal-reservas jsx III	177
Ilustración 313 Modal-reservas jsx IV.....	177
Ilustración 314 Modal-reservas jsx V.....	178
Ilustración 315 Modal-reservas jsx VI.....	178
Ilustración 316 Modal-reservas jsx VII.....	179
Ilustración 317 Modal-reservas jsx VIII.....	179
Ilustración 318 Modal-reservas jsx IX.....	180
Ilustración 319 Modal-reservas jsx X.....	180
Ilustración 320 Modal-reservas jsx XI.....	181
Ilustración 321 Modal-reservas jsx XII.....	181
Ilustración 322 Modal-reservas jsx XIII	182
Ilustración 323 Modal-reservas jsx XIV.....	182
Ilustración 324 Modal-reservas jsx XV.....	183
Ilustración 325 Moda-usuarios-perfil	183
Ilustración 326 Modal-usuarios-perfil jsx I	184
Ilustración 327 Modal-usuarios-perfil jsx II	184
Ilustración 328 Modal-usuarios-perfil jsx III	185
Ilustración 329 Modal-usuarios-perfil jsx IV	185
Ilustración 330 Modal-usuarios-perfil jsx V	186
Ilustración 331 Modal-usuarios-perfil jsx VI	186
Ilustración 332 Despliegue I.....	187
Ilustración 333 Despliegue II.....	187
Ilustración 334 Despliegue III.....	188
Ilustración 335 Despliegue IV	188
Ilustración 336 Despliegue V	188
Ilustración 337 Despliegue VI	189
Ilustración 338 Despliegue VII	189
Ilustración 339 Despliegue VIII	189
Ilustración 340 Despliegue IX	190
Ilustración 341 Despliegue X	190
Ilustración 342 Despliegue XI	190
Ilustración 343 Despliegue XII	191
Ilustración 344 Despliegue XIII	191
Ilustración 345 Despliegue XIV	192
Ilustración 346 Despliegue XV	192
Ilustración 347 Despliegue XVI	193
Ilustración 348 Despliegue XVII	193
Ilustración 349 Despliegue XVIII	193

Ilustración 350 Perfil de Github	194
Ilustración 351 Pruebas I	195
Ilustración 352 Pruebas II	195
Ilustración 353 Pruebas III	
Ilustración 354 Pruebas IV	
Ilustración 355 Pruebas V	196
Ilustración 356 Pruebas VI	
Ilustración 357 Pruebas VII	
Ilustración 358 Pruebas VIII	196
Ilustración 359 Pruebas IX	
Ilustración 360 Pruebas X	197
Ilustración 361 Pruebas XI	197
Ilustración 362 Pruebas XII	198
Ilustración 363 Pruebas XIII	199
Ilustración 364 Pruebas XIX	199
Ilustración 365 Presupuesto	200
Ilustración 366 Tabla de Objetivos	201
Ilustración 367 Problema encontrado	201
Ilustración 368 Pago mensual de cohorte	202
Ilustración 369 Manual de Usuario I	203
Ilustración 370 Manual de Usuario II	203
Ilustración 371 Manual de Usuario III	204
Ilustración 372 Manual de Usuario IV	204
Ilustración 373 Consultar Publicaciones I	205
Ilustración 374 Consultar Publicaciones Home II	205
Ilustración 375 Consultar Publicaciones Home III	205
Ilustración 376 Consultar Publicaciones Home IV	206
Ilustración 377 Consultar Publicaciones Informativos I	206
Ilustración 378 Consultar Publicaciones Informativos II	207
Ilustración 379 Consultar Publicaciones Informativos III	207
Ilustración 380 Consultar Publicaciones Informativos IV	208
Ilustración 381 Consultar Publicaciones Reservables I	208
Ilustración 382 Consultar Publicaciones Reservables II	209
Ilustración 383 Consultar Publicaciones Reservables III	209
Ilustración 384 Consultar Publicaciones Reservables IV	210
Ilustración 385 Nav Menu Realizar Reservas y Consultar Reservas	210
Ilustración 386 Realizar reserva	210
Ilustración 387 Consultar Reservas I	211
Ilustración 388 Consultar Reservas II	
Ilustración 389 Consultar Reservas III	211

Ilustración 390 Restaurar Contraseña I	212
Ilustración 391 Restaurar Contraseña II	
Ilustración 392 Restaurar Contraseña III	212
Ilustración 393 Restaurar Contraseña IV email I	213
Ilustración 394 Restaurar Contraseña V email II	213
Ilustración 395 Restaurar Contraseña VI	213
Ilustración 396 Restaurar Contraseña VII	214
Ilustración 397 IA Chatbox.....	214

1. Introducción.

La importancia de este proyecto denominado (Costa Alquiler) consiste en llevar el control total de los alquileres y eventos de la zona costera de Málaga (Alquiler de Motos de Agua, alquiler de lanchas, viaje pagado a visitar los delfines ...) además de poder informar al usuario de posibles eventos en la playa como pueden ser concursos de castillos de arena, partidos de Voleyball etc.

La aplicación tendrá una sección principal donde se informar al usuario de los eventos (tanto informativos como de pago), Además de esto, el usuario podrá ver en todo momento en un calendario los eventos de ese mes. Podrá cancelar la reserva en todo momento.

Este proyecto será realizado en una página web.

2. Objetivos y Requisitos.

→Objetivos.

- Gestionar Alquileres y Eventos, ya sea tanto la reserva como la cancelación de los mismos.
- Gestionar Usuarios, permitir al usuario la creación de un usuario personal y la restauración de la contraseña de los mismos.
- Un calendario desde el cual poder gestionar y recordar tus eventos.
- Un modo administrador para poder gestionar los usuarios y los eventos y alquileres.

→Requisitos.

Los requisitos que pone el cliente son:

- Usuario y clave de acceso para poder acceder a la página.
- Menú con 4 apartados (Inicio, Alquileres, Eventos, Perfil).
- Un apartado para poder ver en todo momento los eventos y alquileres reservados.
- Poder cancelar en todo momento dicha reserva.

3. Estudio Previo.

Este apartado consiste en recopilar y hacer un análisis de los datos necesarios para poder contemplar las posibles soluciones a nuestro proyecto.

→Hacerlo con PHP puro.

La primera de las soluciones es hacerlo con php puro la página y la API (interfaz de programación de aplicaciones), pero esto tiene un inconveniente y es que el código no será de tanta calidad y tardaremos más que usando un framework.

→Hacerlo con React y Laravel.

Esta es la opción mejor premiada, ya que usando React no tendremos problemas para adelantar en nuestro proyecto porque podemos copiar plantillas ya creadas, en el caso de laravel tenemos un problema y es que la curva de aprendizaje es tremenda por lo que para poder hacer el proyecto en laravel necesitaremos mucho tiempo de aprendizaje, tiempo que estamos usando para aprender en React.

→Hacerlo con React (la página) y PHP la API.

Esta opción aunque no sea la más perfecta para usar, es la más equilibrada en cuanto a coste/tiempo ya que no disponemos de mucho más tiempo que 3 meses.

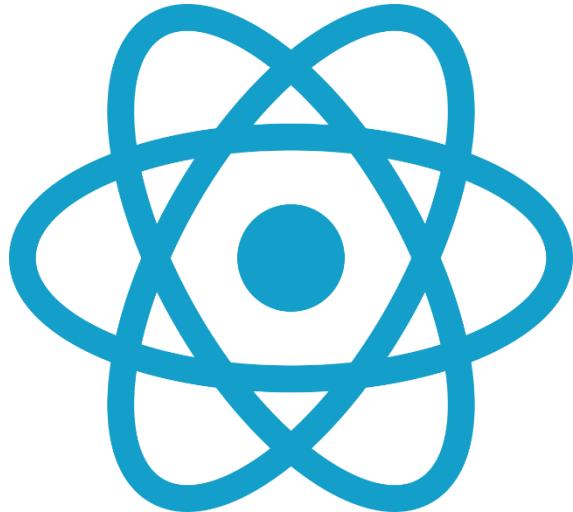


Ilustración 1 Logo de React

4. Plan de Trabajo

Plan de Trabajo	
Semanas	Tareas
Semana del 17 al 21 de Marzo de 2025	Introducción a la empresa y presentación del proyecto. Se explican las pautas generales a seguir y se realiza un primer análisis de lo que se va a desarrollar.
Semana del 24 al 28 de Marzo de 2025	Desarrollo del punto 5 del proyecto, centrado en la parte de diseño, tanto del modelo de datos como de las pantallas principales.
Semana del 31 de Marzo al 4 de Abril de 2025	Comienzo de la implementación del proyecto: estructura inicial, conexión con la API, y primeros componentes visuales.
Semana del 7 al 11 de Abril de 2025	Se continúa con la implementación. A su vez, se avanza en un curso de React para afianzar conceptos clave necesarios para el desarrollo.
Semana del 14 al 18 de Abril de 2025	Semana Santa
Semana del 21 al 25 de Abril de 2025	Reanudación de la implementación y continuidad con el curso de React para mejorar el rendimiento y la calidad del código.
Semana del 28 al 2 de Mayo de 2025	Se sigue avanzando en la lógica de la aplicación y se integran nuevas funcionalidades.
Semana del 5 al 9 de Mayo de 2025	Finalización de la parte de implementación y se empieza con la integración de aspectos de seguridad en la aplicación.
Semana del 12 al 16 de Mayo de 2025	Fase de pruebas de la aplicación para detectar errores. También se documentan los recursos usados como hardware, software, personal y presupuesto.
Semana del 19 al 23 de Mayo de 2025	Redacción de las conclusiones del proyecto y recopilación de bibliografía. Inicio de los anexos.

Semana del 26 al 30 de Mayo de 2025	Desarrollo del manual de usuario, explicando el funcionamiento básico de la aplicación.
Semana del 2 de Junio al 6 de Junio de 2025	Finalización del manual de usuario.
Semana del 9 al 13 de Junio de 2025	Entrega del proyecto final. Se revisa toda la documentación y se prepara para su presentación.

Ilustración 2 Plan de Trabajo

5. Diseño.

→Diseño Base de Datos.

Entidades.

→ Usuario: En esta tabla se almacenarán los usuarios que se registren en la aplicación y guardará Id, Nombre, Usuario, Apellidos, Fecha de Nacimiento, Email, Tipo, Contraseña.

→ Publicación: En esta tabla se almacenarán las publicaciones que son creadas por el usuario de tipo administrador y almacenarán Id, Titulo, Descripción, Fecha de publicación, Imagen, Tipo y Coste.

→ Reservas: En esta tabla se almacenarán las reservas de las publicaciones y esta tabla almacenará Id_publicación, Id_usuario y Fecha_reserva.

Relaciones.

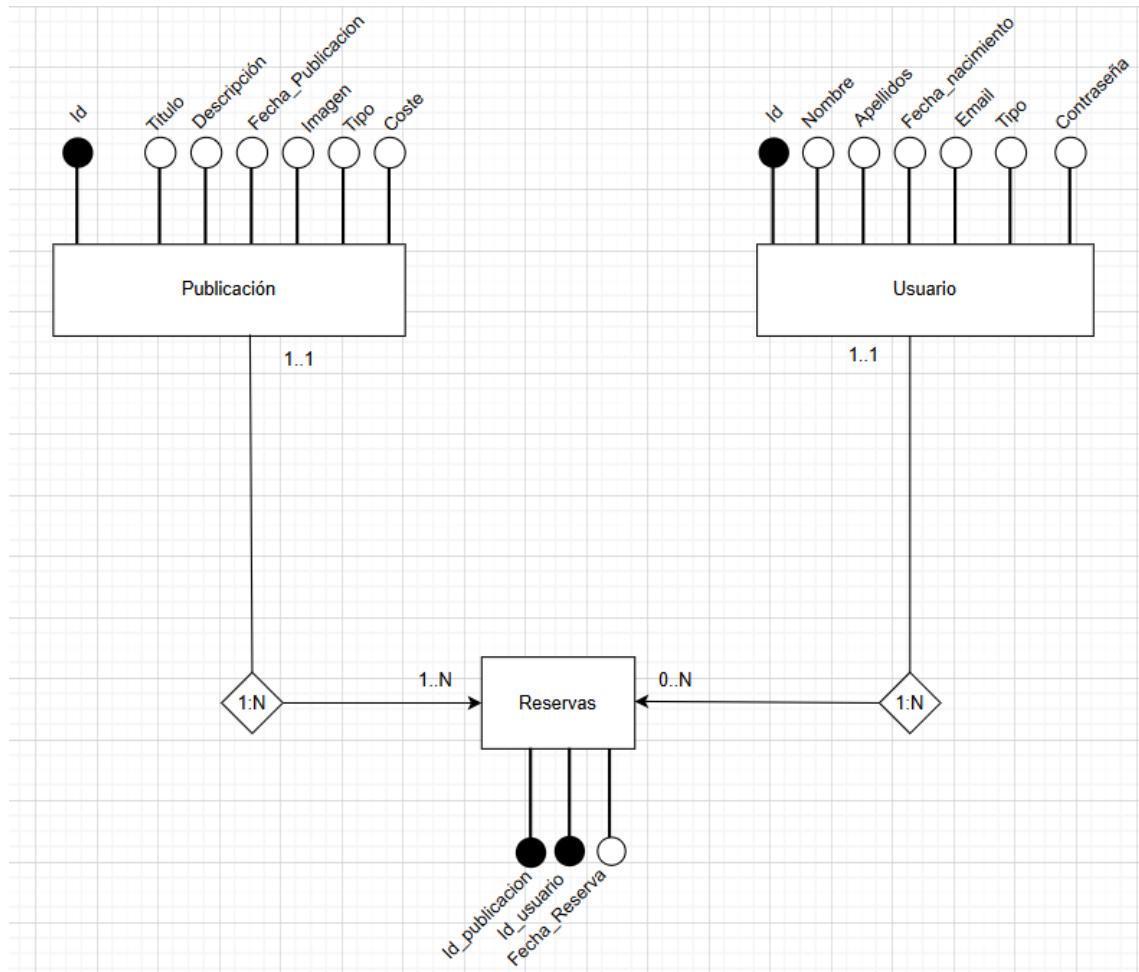


Ilustración 3 Diseño E.R

→**Tablas y Normalización.**

USUARIO(Id, Nombre, Apellidos, Fecha_Nacimiento, Email, Tipo, Contraseña).

-Clave Primaria (Id) Único, No Nulo.

1FN: Se encuentra en 1FN porque cada atributo tiene un valor único en cada fila.

2FN: Se encuentra en 2FN porque está en 1FN y además para saber el valor de los atributos necesito saber el valor de la clave primaria completa.

3FN: Se encuentra en 3FN porque está en 2FN y porque ni hay ningún campo que sustituya a la clave primaria con el cual podamos saber el valor de otro campo.

PUBLICACIÓN(Id, Id_Usuario, Titulo, Descripción, Fecha_Publicación, Imagen, Tipo, Coste).

-Clave Primaria (Id) Único, No Nulo.

-Clave Externa (Id_Usuario) referenciando a (Id) en Usuario.

En actualización: en cascada.

En borrado: en cascada.

1FN: Se encuentra en 1FN porque cada atributo tiene un valor único en cada fila.

2FN: Se encuentra en 2FN porque está en 1FN y además para saber el valor de los atributos necesito saber el valor de la clave primaria completa.

3FN: Se encuentra en 3FN porque está en 2FN y porque ni hay ningún campo que sustituya a la clave primaria con el cual podamos saber el valor de otro campo.

RESERVAS(Id_Usuario, Id_Publicación, Fecha_Reserva).

-Clave Primaria (Id_Usuario, Id_Publicación) Único, No Nulo.

-Clave Externa (Id_Publicación) referenciando a (Id) en Publicación.

En actualización: en cascada.

En borrado: en cascada.

1FN: Se encuentra en 1FN porque cada atributo tiene un valor único en cada fila.

2FN: Se encuentra en 2FN porque está en 1FN y además para saber el valor de los atributos necesito saber el valor de la clave primaria completa.

3FN: Se encuentra en 3FN porque está en 2FN y porque ni hay ningún campo que sustituya a la clave primaria con el cual podamos saber el valor de otro campo.

→Diagrama Relacional.

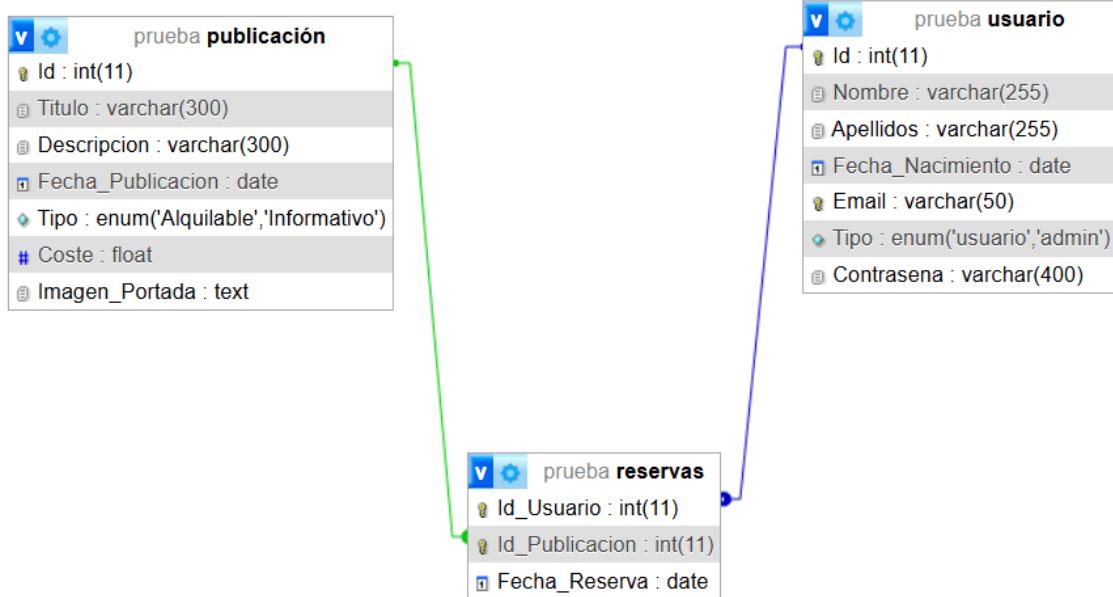


Ilustración 4 Modelo Relacional

→Diseño General.

Cuando abrimos la aplicación. Comprobamos si está registrado:

-Si está deslogueado lo llevamos a la página de login, la cual pide las credenciales para el inicio de sesión, puede ocurrir que estas credenciales sean o no sean correctas.

-Si la clave es incorrecta, mostraremos un mensaje por si queremos resetear la contraseña , resetearemos la nueva clave y accederemos al inicio al resetear la clave.

-Si la clave es correcta, accederemos al inicio.

-Si esta loggeado simplemente accedemos al inicio de la app.

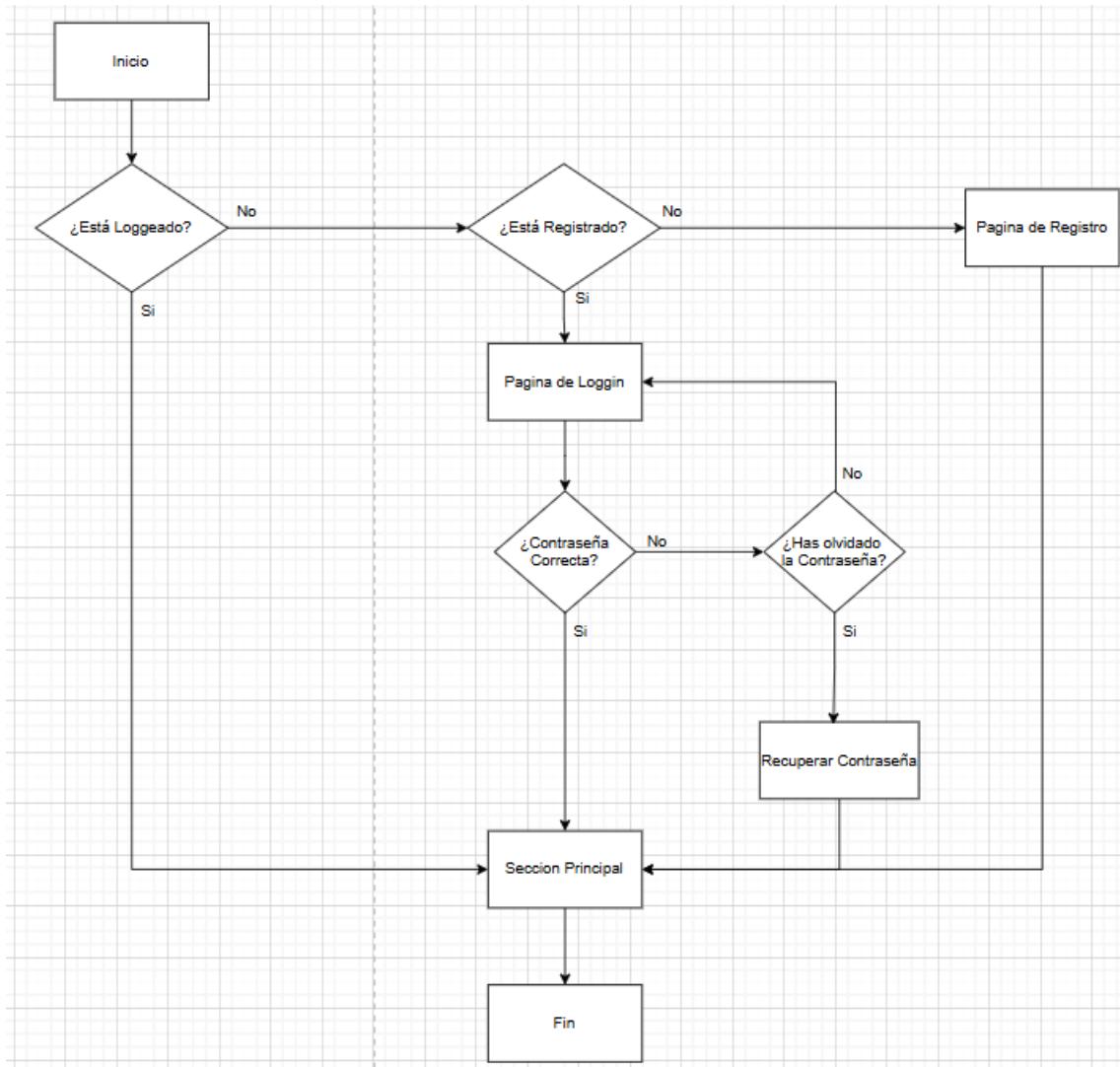


Ilustración 5 Diagrama de Flujo de Datos

→Diseño Detallado.

Cuando nos registramos en la Aplicación (En caso de que no estemos loggeados y no tengamos cuenta) o cuando nos logeamos , se nos redirigirá a la página principal de la app en la que viene la información de los eventos principales , una vez dentro habrá un menú en el que podremos viajar a las diferentes secciones de la página , como mi perfil, eventos a los que podremos acceder, alquileres

-Mi perfil: Esta ventana nos dejará cambiar todos los datos personales como el email , el correo

-Mis reservas: aquí tendremos guardados los próximos eventos a los que vamos a ir (los que están reservados), además podremos cancelar en todo momento las reservas que tenemos activas.

-Alquileres: Esta ventana nos mostrará todos los alquileres que hay posibles para reservar.

-Informativos: Esta página nos mostrara los eventos gratuitos turísticos que hay para realizar posibles.

-Página de Inicio: Página principal informativa sobre el porqué de la página web que contiene la información sobre el lugar donde van a existir los alquileres y los eventos.

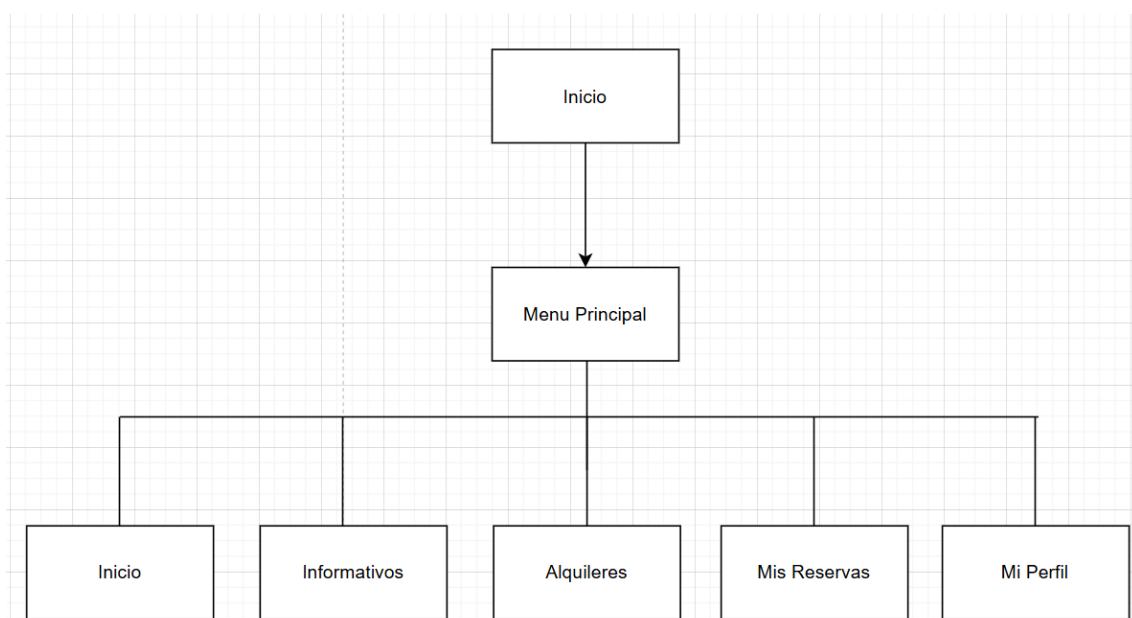


Ilustración 6 Diseño a Detalle

→Boceto Aplicación.

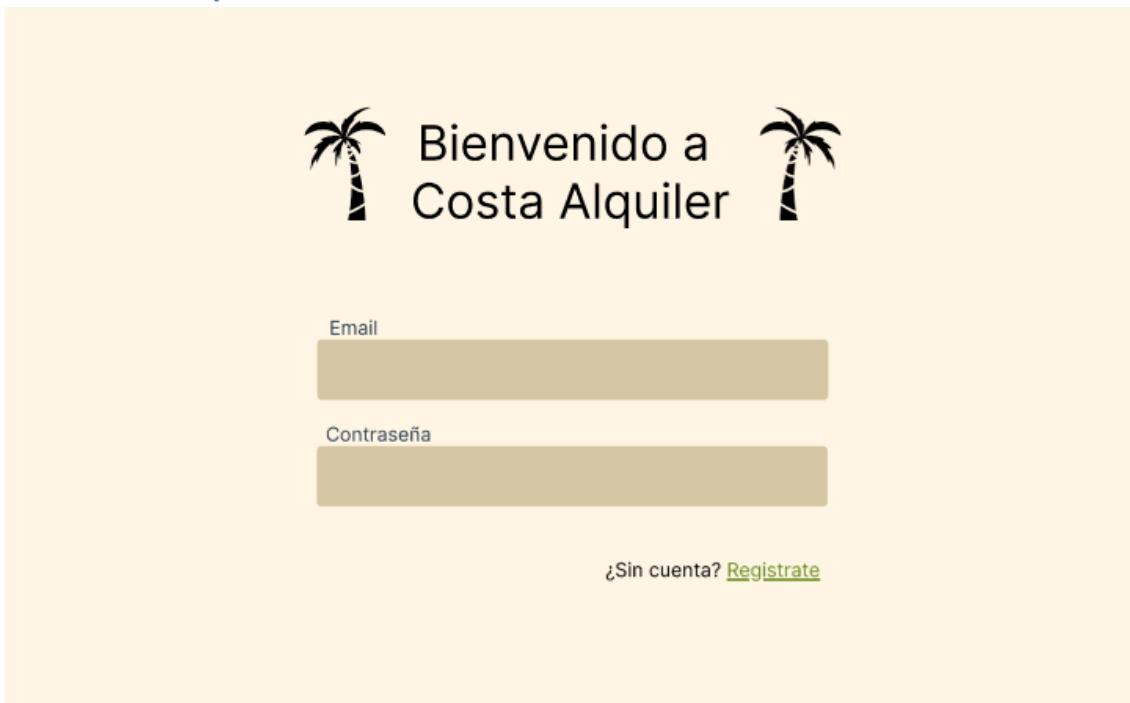


Ilustración 7 Página Login

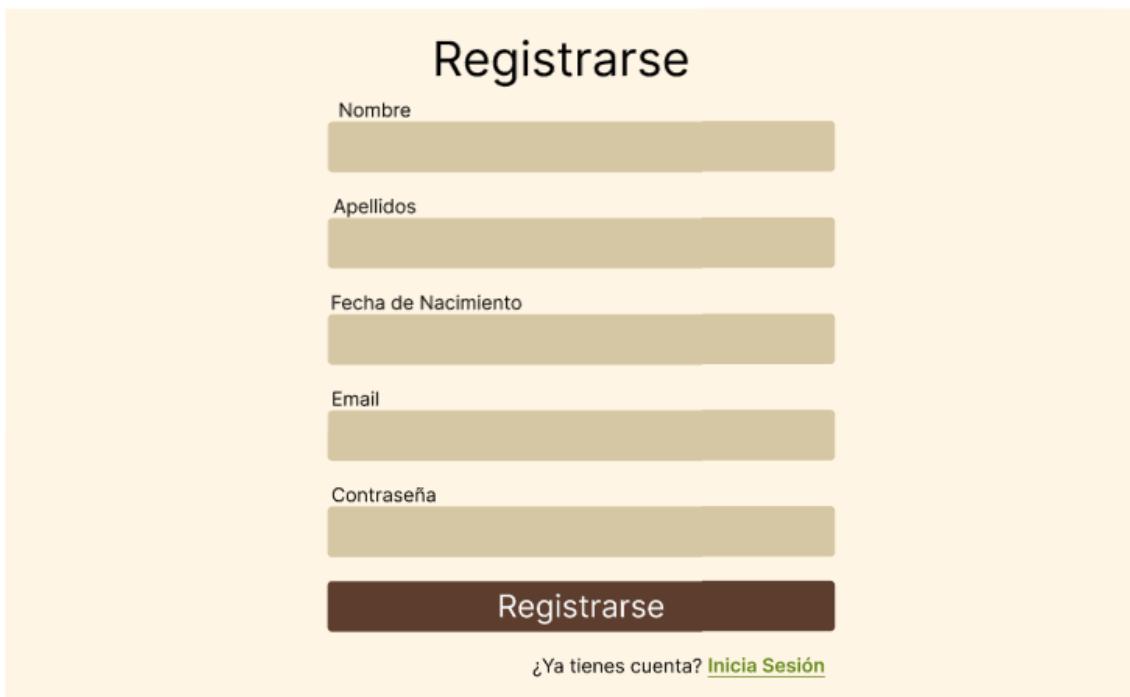


Ilustración 8 Página de Register



Ilustración 9 Página de Olvido de Contraseña

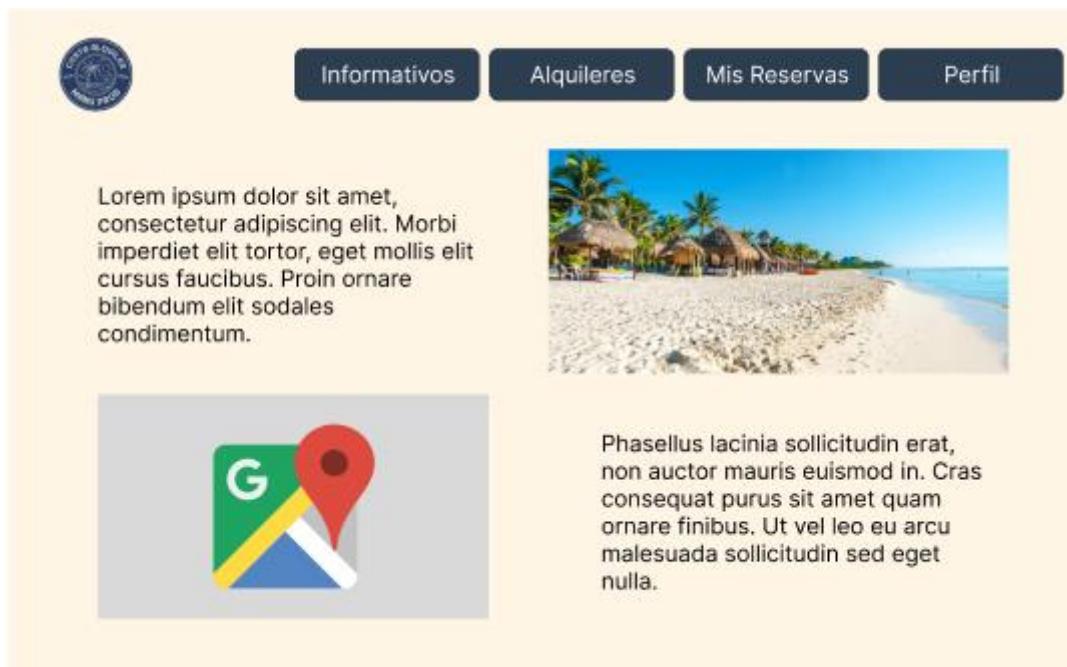


Ilustración 10 Página de Inicio

The screenshot shows a user interface for a rental service. At the top, there is a logo for 'COSTA ALQUILERES MARÍNU PROD' and four navigation buttons: 'Informativos', 'Alquileres', 'Mis Reservas', and 'Perfil'. Below this, a section titled 'Filtrar Resultados Por:' has two dropdown menus. The main content area displays two items, each consisting of a green vertical bar on the left and a white box on the right. The first item has a title 'Titulo' and a short description: 'Phasellus lacinia sollicitudin erat, non auctor mauris euismod in. Cras consequat purus sit amet quam ornare finibus.' It includes a 'Ver Más' button. The second item also has a title 'Titulo' and a similar description. Both items have a 'Reservar' button at the bottom.

Ilustración 11 Página de Informativos/Alquilables

This screenshot shows a detailed view of a rental or information item. At the top, there is a logo for 'COSTA ALQUILERES MARÍNU PROD' and a 'Volver' button. The main content area features a large green placeholder image on the left and a white box on the right. The white box contains a title 'Titulo' and a long, repetitive description: 'Phasellus lacinia sollicitudin erat, non auctor mauris euismod in. Cras consequat purus sit amet quam ornare finibus. Phasellus lacinia sollicitudin erat, non auctor mauris euismod in. Cras consequat purus sit amet quam ornare finibus. Phasellus lacinia sollicitudin erat.' It includes a 'Reservar' button with a calendar icon.

Ilustración 12 Página vista Alquiler/Informativo.



Ilustración 13 Mis Reservas



Ilustración 14 Mi Perfil

6. Implementación

Servidor

ENRUTADOR

-*Api.php*

Contiene los endpoints que van a ser usados por el cliente.

Esta captura muestra los endpoints protegidos con Sanctum “tokens”, ya que para poder hacer peticiones a estos , necesitamos un token de inicio de sesión que se consiguen al iniciar sesión con valores válidos.

Dentro podemos ver que tenemos un middleware que se llama isAdmin el cual se encarga de controlar el acceso a esos endpoints para solo los usuarios administradores.

```
Route::middleware('auth:sanctum')->group(function () {
    // Rutas solo para admin
    Route::middleware('isAdmin')->group(function () {
        // Rutas completas (CRUD) para usuarios, publicaciones y reservas (solo para admin)
        Route::apiResource('usuarios', UsuarioController::class)->only(['destroy', 'show']);
        Route::apiResource('publicaciones', PublicacionController::class)->only(['update', 'store', 'destroy']);
        Route::apiResource('reservas', ReservaController::class)->only(['update', 'destroy']);

        Route::post('usuariosPaginados', [UsuarioController::class, 'paginarUsuarios']);
        Route::post('publicacionesPaginadas', [PublicacionController::class, 'obtenerPaginadas']);
        Route::post('reservasPaginadas', [ReservaController::class, 'paginarReservas']);

        // Rutas específicas de actualización que solo puede usar admin
        Route::post('actualizar', [PublicacionController::class, 'update']);
        Route::post('actualizarReservas', [ReservaController::class, 'update']);
        Route::post('actualizarFechaPublicacion', [ReservaController::class, 'actualizarFechaPublicacionYReservas']);
        Route::post('intercambiarFechas', [ReservaController::class, 'intercambiarReserva']);

        // Subida de imágenes
        Route::post('upload', [ImageController::class, 'upload']);
    });
    // Rutas para administrador
    Route::apiResource('usuarios', UsuarioController::class)->only(['index', 'store', 'update']);
    Route::apiResource('publicaciones', PublicacionController::class)->only(['index', 'show']);
    Route::apiResource('reservas', ReservaController::class)->except(['update']);

    // Publicaciones informativas y alquilables
    Route::get('informativos', [PublicacionController::class, 'obtenerInformativos']);
    Route::get('alquilables', [PublicacionController::class, 'obtenerAlquilables']);
    Route::get('publicacionesAleatorias', [PublicacionController::class, 'obtenerPublicacionesAleatorias']);
});
```

Ilustración 15 EndPoints Sanctum I

Este es el middleware llamado isAdmin que se encarga de controlar el acceso a los endpoints de tipo admin.

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

1 reference | 0 overrides
class isAdmin
{
    0 references | 0 overrides
    public function handle(Request $request, Closure $next): mixed
    {
        if ($auth()->check() && $auth()->user()->Tipo === 'admin') {
            return $next($request);
        }

        return $response()->json([
            'message' => 'Acceso denegado. Solo los administradores pueden realizar esta acción.'
        ], 403);
    }
}
```

Ilustración 16 Middleware isAdmin

Fuera de el middleware tenemos mas métodos que estos pueden ser usados por todos los usuarios que no sean admin.

```
// Usuario autenticado
Route::get('usuario/getId', [UsuarioController::class, 'obtenerUsuarioAutenticado']);
Route::get('getData', [UsuarioController::class, 'obtenerDatosUsuarioAutenticado']);
Route::post('mailTo', [UsuarioController::class, 'enviarCorreoPersonalizado']);
Route::get('isAdmin', [UsuarioController::class, 'obtenerAdmin']);
Route::post('usuarios/actualizar', [UsuarioController::class, 'actualizar']);

// Reservas extras
Route::post('disponibilidadReserva', [ReservaController::class, 'getDisponibilidad']);
Route::post('capacidadDisponible', [ReservaController::class, 'getCapacidadDisponible']);
Route::get('obtenerReservasUsuario', [ReservaController::class, 'getReservasPorUsuario']);
Route::get('obtenerReservasUsuario/{id}', [ReservaController::class, 'getReservasPorIdUsuario']);
Route::get('obtenerReservasDetalladas', [ReservaController::class, 'obtenerReservasDetalladas']);

// Página
Route::post('informativosPaginados', [PublicacionController::class, 'obtenerInformativosPaginados']);
Route::post('alquilablesPaginados', [PublicacionController::class, 'obtenerAlquilablesPaginados']);
```

Ilustración 17 EndPoints Santum II

Estas capturas muestran los endpoints que pueden ser usados sin necesidad de token de inicio de sesion, algunos de ellos son horaFecha , enviar-restablecimiento y todos sus iguales , register y login.

```
//Funcion para obtener la fecha y la hora del Server
Route::get('horafecha', function () : mixed {
    $now = Carbon::now('Europe/Madrid'); // Establece la zona horaria manualmente

    return response()->json([
        'fecha' => $now->format('Y-m-d'),
        'hora' => $now->format('H:i:s')
    ]);
});

// Esto es para restablecer la contraseña
// envia al correo el enlace para restablecer la contraseña
Route::post('enviar-restablecimiento', [RestablecerPasswordController::class, 'enviarRestablecimiento'])->name('enviar.restablecimiento');
//Muestra el formulario al hacer click en el enlace del correo
Route::get('restablecer-password/{email}', [RestablecerPasswordEmailController::class, 'mostrarFormulario'])->name('mostrar.formulario.restablecer');
// Restablece la contraseña al dar click en restablecer contraseña en la pagina de restablecimiento de contraseña
Route::post('restablecer-password', [RestablecerPasswordEmailController::class, 'restablecer'])->name('restablecer.password');
//Retorna la vista de success
Route::get('password-reset-success', function () : mixed {
    return view('password.reset-success');
})->name('password.reset.success');
```

Ilustración 18 Endpoints Públicos I

```
////Register
Route::post('register', function (Request $request) : mixed {
    $validated = $request->validate([
        'Nombre' => 'required|string|max:255',
        'Apellidos' => 'required|string|max:255',
        'Fecha_nacimiento' => 'required|date',
        'Email' => 'required|email|unique:usuarios,Email',
        'Password' => 'required|string|min:6|confirmed',
    ]);

    // Crear un nuevo usuario
    $usuario = Usuario::create([
        'Nombre' => $validated['Nombre'],
        'Apellidos' => $validated['Apellidos'],
        'Fecha_nacimiento' => $validated['Fecha_nacimiento'],
        'Email' => $validated['Email'],
        'Password' => Hash::make($validated['Password']),
        'Tipo' => 'usuario'
    ]);

    // Crear un token para el nuevo usuario
    $token = $usuario->createToken('token-api')->plainTextToken;

    // Devolver el token al cliente
    return response()->json(['token' => $token], 201);
});

//Metodo de Login
Route::post('login', function (Request $request) : mixed {
    $usuario = Usuario::where('Email', $request->email)->first();

    if (!$usuario || !Hash::check($request->password, $usuario->Password)) {
        return response()->json(['message' => 'Credenciales incorrectas'], 401);
    }

    $token = $usuario->createToken('token-api')->plainTextToken;

    return response()->json(['token' => $token], 200);
});
```

Ilustración 19 Endpoints Públicos II

MODELOS

Los modelos definen la manera en la que se van a interpretar los datos de la base de datos.

- Modelo usuario.php

Contiene los campos y las relaciones de la tabla usuarios de la base de datos.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

23 references | 0 implementations
class Usuario extends Authenticatable
{
    use HasApiTokens, Notifiable, HasFactory;
    0 references
    protected $table = 'usuarios';

    0 references
    protected $fillable = [
        'Nombre',
        'Apellidos',
        'Fecha_nacimiento',
        'Email',
        'Tipo',
        'Password'
    ];

    0 references | 0 overrides
    public function publicaciones(): mixed
    {
        return $this->hasMany(Publicacion::class);
    }

    // Relación uno a muchos con reservas
    0 references | 0 overrides
    public function reservas(): mixed
    {
        return $this->hasMany(Reserva::class);
    }
}
```

Ilustración 20 Modelo usuario.php

- *Modelo publicacion.php*

Contiene los campos y las relaciones de la tabla de publicaciones en la base de datos.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

25 references | 0 implementations
class Publicacion extends Model
{
    use HasFactory;
    0 references
    protected $table = 'publicaciones';
    0 references
    protected $fillable = [
        'Titulo',
        'Descripcion',
        'Fecha_evento',
        "Tipo",
        "Precio",
        "Imagen",
        "Aforo_maximo",
    ];
    0 references | 0 overrides
    public function usuario(): mixed
    {
        return $this->belongsTo(Usuario::class);
    }
}
```

Ilustración 21 Modelo publicacion.php

- *Modelo reservas.php*

Contiene los campos y las relaciones de la tabla de reservas en la base de datos.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

19 references | 0 implementations
class Reserva extends Model
{
    use HasFactory;
    0 references
    protected $table = 'reservas';

    0 references
    protected $fillable = [
        'usuario_id',
        'publicacion_id',
        'fecha_reserva',
        'total_pagar',
        'personas',
    ];

    0 references | 0 overrides
    public function usuario(): mixed
    {
        return $this->belongsTo(Usuario::class);
    }
    0 references | 0 overrides
    public function publicacion(): mixed
    {
        return $this->belongsTo(Publicacion::class);
    }
}
```

Ilustración 22 Modelo reservas.php

MIGRATIONS

Las migraciones se encargan de crear las tablas, relaciones y campos necesarios para la base de datos.

- Migration usuario.php

Contiene la definición de los campos y las relaciones de la tabla de usuarios de la base de datos.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
|
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('usuarios', function (Blueprint $table): void {
            $table->id();
            $table->string('Nombre');
            $table->string('Apellidos');
            $table->date('Fecha_nacimiento');
            $table->string('Email')->unique();
            $table->enum('Tipo', ['usuario', 'admin']);
            $table->string('Password');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('usuarios');
    }
};
```

Ilustración 23 Migracion usuario.php

- *Migration publicacion.php*

Contiene la definición de los campos y las relaciones de la tabla de publicación de la base de datos.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('publicaciones', function (Blueprint $table): void {
            $table->id();
            $table->string('titulo');
            $table->string('descripcion');
            $table->datetime('fecha_evento');
            $table->enum('tipo', ['alquilable', 'informativo']);
            $table->integer('precio')->nullable();
            $table->text('imagen')->nullable();
            $table->integer('aforo_maximo')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('publicaciones');
    }
};
```

Ilustración 24 Migracion publicación.php

- *Migration reservas.php*

Contiene la definición de los campos y las relaciones de la tabla de reservas de la base de datos.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('reservas', function (Blueprint $table): void {
            $table->id();
            $table->unsignedBigInteger("usuario_id");
            $table->unsignedBigInteger("publicacion_id");
            $table->datetime('fecha_reserva');
            $table->integer('total_pagar')->nullable();
            $table->integer('personas')->nullable();
            $table->timestamps();

            // CORREGIDO aqui
            $table->foreign('usuario_id')->references('id')->on('usuarios')->onDelete('cascade');
            $table->foreign('publicacion_id')->references('id')->on('publicaciones')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('reservas');
    }
};
```

Ilustración 25 Migracion reservas.php

FACTORIES

Los Factory son usados para generar registros de datos aleatorios en la base de datos.

- Factory usuario.php

Factory de usuario para generar datos fake o aleatorios

```
<?php

namespace Database\Factories;

use App\Models\Usuario;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;

0 references | 0 implementations
class UsuarioFactory extends Factory
{
    0 references
    protected $model = Usuario::class;

    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'Nombre' => $this->faker->firstName, // Nombre del usuario
            'Apellidos' => $this->faker->lastName, // Apellidos del usuario
            'Fecha nacimiento' => $this->faker->date, // Fecha de nacimiento (formato yyyy-mm-dd)
            'Email' => $this->faker->unique()->safeEmail, // Correo electrónico único
            'Tipo' => $this->faker->randomElement(['admin', 'usuario']), // Tipo de usuario (admin o usuario)
            'Password' => bcrypt('password'), // Contraseña encriptada
        ];
    }
}
```

Ilustración 26 Factory de usuario.php

- Factory publicacion.php

Factory de publicación para generar datos fake o aleatorios

```
namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Publicacion>
 *
 * Esta clase define una fábrica para el modelo Publicacion.
 * Las fábricas en Laravel permiten generar datos de prueba de manera sencilla.
 */
0 references | 0 implementations
class PublicacionFactory extends Factory
{
    /**
     * Define el estado por defecto del modelo.
     *
     * @return array<string, mixed> Un array con los valores por defecto para cada campo.
     */
    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'Titulo' => $this->faker->sentence, // Genera un título aleatorio
            'Descripcion' => $this->faker->paragraph, // Genera una descripción aleatoria
            'Fecha_evento' => $this->faker->date, // Genera una fecha aleatoria para el evento
            'Tipo' => $this->faker->randomElement(['alquilable', 'informativo']), // Selecciona aleatoriamente entre 'alquilable' o 'informativo'
            'Precio' => $this->faker->numberBetween(100, 10000), // Genera un precio aleatorio entre 100 y 10000
            'Imagen' => $this->faker->imageUrl(640, 480, 'business', true), // Genera una URL de imagen aleatoria
            'Aforo_maximo' => $this->faker->numberBetween(100, 500), // Genera un aforo máximo aleatorio entre 100 y 500
        ];
    }
}
```

Ilustración 27 Factory de publicación.php

- Factory reserva.php

Factory de reserva para generar datos fake o aleatorios

```
<?php

namespace Database\Factories;

use App\Models\Publicacion;
use App\Models\Usuario;
use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Reserva>
 */
0 references | 0 implementations
class ReservaFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
0 references | 0 overrides
    public function definition(): array
    {
        return [
            'usuario_id' => Usuario::factory(), // Relacionamos con un usuario aleatorio
            'publicacion_id' => Publicacion::factory(), // Relacionamos con una publicación aleatoria
            'fecha_reserva' => $this->faker->dateTimeBetween('now', '+1 year'), // Fecha de reserva en el futuro, hasta un año
            'total_pagar' => $this->faker->numberBetween(100, 10000), // Monto a pagar aleatorio
            'personas' => $this->faker->numberBetween(1, 4), // Número de personas en la reserva
        ];
    }
}
```

Ilustración 28 Factory de reserva.php

CONTROLLERS

Los controladores son aquellos que contienen los métodos necesarios para el funcionamiento de los endpoints

-ImageController.php

Controlador de imagen el cual solo contiene un método uploadImage el cual se encarga de guardar imágenes en el servidor.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

5 references | 0 implementations
class ImageController extends Controller
{
    /**
     * Upload image
     */
    public function uploadImage(Request $file): array
    {
        // Validar archivo directamente aquí
        if (!$file->isValid() || !in_array($file->extension(), ['jpg', 'jpeg', 'png', 'gif', 'webp'])) {
            return [
                'success' => false,
                'message' => 'Archivo no válido'
            ];
        }

        $path = $file->store('photos', 'public');

        return [
            'success' => true,
            'path' => asset('storage/' . $path)
        ];
    }
}
```

Ilustración 29 Controller ImageController.php

-UserController.php

Controller de Usuario el cual contiene los métodos necesarios para las solicitudes de los enpoints.

-Index (Protegido)

El método index se encarga de retornar todos los usuarios.

GET /usuarios

```
0 references | 0 overrides
public function index(): mixed
{
    $usuarios = Usuario::all();
    return response()->json(data: $usuarios);
}
```

Ilustración 30 Index de usuarios

-Paginar Usuarios (Protegido)

El método paginarUsuarios se encarga de retornar todos los usuarios de manera paginada.

GET /usuariosPaginados

```
/*
 * POST /paginarusuarios
 * Devuelve usuarios paginados. Recibe 'pagina' en el body.
 */
1 reference | 0 overrides
public function paginarUsuarios(Request $request): mixed
{
    // Número de elementos por página
    $perPage = 10;

    // Página solicitada (por defecto 1)
    $page = (int) $request->input('pagina', 1);
    $page = max(value: $page, values: 1);

    // Total de usuarios y páginas
    $total = Usuario::count();
    $totalPages = (int) ceil(num: $total / $perPage);

    // Usuarios de la página solicitada
    $usuarios = Usuario::skip($page - 1) * $perPage
        ->take($perPage)
        ->get()
        ->map(function ($usuario): array {
            return [
                'id' => $usuario->id,
                'Nombre' => $usuario->Nombre,
                'Apellidos' => $usuario->Apellidos,
                'Email' => $usuario->Email,
                'Fecha_nacimiento' => $usuario->Fecha_nacimiento,
                'Tipo' => $usuario->Tipo,
                'updated_at' => $usuario->updated_at,
            ];
        });

    return response()->json([
        'success' => true,
        'data' => $usuarios,
        'page' => $page,
        'totalPages' => $totalPages
    ]);
}
```

Ilustración 31 Paginar Usuarios

-Is Admin

El método is Admin se encarga de comprobar si un usuario es administrador o no.

GET /isAdmin

```
/**
 * GET /isAdmin|
 * Devuelve si el usuario autenticado es admin.
 */
1 reference | 0 overrides
public function obtenerAdmin(): mixed
{
    $usuario = auth()->user();

    return response()->json([
        'is_admin' => $usuario->Tipo === 'admin'
    ]);
}
```

Ilustración 32 Is Admin

-Obtener Usuario Autenticado

El método obtenerUsuarioAutenticado se encarga de retornar solo el id del Usuario.

GET /usuario/getId

```
/**
 * GET /isAdmin|
 * Devuelve si el usuario autenticado es admin.
 */
1 reference | 0 overrides
public function obtenerAdmin(): mixed
{
    $usuario = auth()->user();

    return response()->json([
        'is_admin' => $usuario->Tipo === 'admin'
    ]);
}
```

Ilustración 33 Obtener Usuario Autenticado

-Store de Usuarios

El método store se encarga de guardar un Usuario.

POST /usuarios

```
/*
 * POST /usuarios
 * Crea un nuevo usuario.
 */
0 references | 0 overrides
public function store(Request $request): mixed
{
    // Validar los datos recibidos
    $validator = Validator::make($request->all(), [
        'Nombre' => 'required|string|max:255',
        'Apellidos' => 'required|string|max:255',
        'Email' => 'required|email|max:255|unique:usuarios,Email',
        'Fecha_nacimiento' => 'required|date',
        'Tipo' => 'required|string|in:admin,usuario',
        'Password' => 'required|string|min:8',
    ]);

    if ($validator->fails()) {
        return response()->json(['errors' => $validator->errors()], 422);
    }

    // Crear el usuario
    $usuario = Usuario::create([
        'Nombre' => $request->input('Nombre'),
        'Apellidos' => $request->input('Apellidos'),
        'Email' => $request->input('Email'),
        'Fecha_nacimiento' => $request->input('Fecha_nacimiento'),
        'Tipo' => $request->input('Tipo'),
        'Password' => bcrypt($request->input('Password')),
    ]);

    return response()->json(['usuario' => $usuario], 201);
}
```

Ilustración 34 Store de Usuarios

-Obtener Datos de los Usuarios Autenticados

El método getData se encarga de retornar los datos del usuario autenticado

GET /getData

```
/*
 * GET /getData
 * Devuelve los datos del usuario autenticado.
 */
1 reference | 0 overrides
public function obtenerDatosUsuarioAutenticado(): mixed
{
    $usuario = auth()->user();

    if (!$usuario) {
        return response()->json(['error' => 'Usuario no autenticado'], 401);
    }

    return response()->json([
        'id' => $usuario->id,
        'Nombre' => $usuario->Nombre,
        'Apellidos' => $usuario->Apellidos,
        'Email' => $usuario->Email,
        'Fecha_nacimiento' => $usuario->Fecha_nacimiento,
        'Tipo' => $usuario->Tipo,
        "updated_at" => $usuario->updated_at,
        "created_at" => $usuario->created_at,
    ]);
}
```

Ilustración 35 Obtener Datos Usuario Autenticado

-Enviar Correo Personalizado

El método enviarCorreoPersonalizado se encarga de retornar los datos del usuario autenticado

GET /mailTo

```
/*
 * POST /mailTo
 * Envía un correo personalizado al usuario.
 */
1 reference | 0 overrides
public function enviarCorreoPersonalizado(Request $request): mixed
{
    $validator = Validator::make($request->all(), [
        'email' => 'required|email',
        'mensaje' => 'required|string|max:1000',
    ]);

    if ($validator->fails()) {
        return response()->json([
            'error' => 'Datos inválidos',
            'messages' => $validator->errors()
        ], 422);
    }

    Mail::to($request->email)->send(new CorreoPersonalizado(mensaje: $request->mensaje));

    return response()->json(['success' => 'Correo enviado correctamente con vista personalizada'], 200);
}
```

Ilustración 36 Enviar correo personalizado

-Show Usuario(Protegido)

El método show se encarga de retornar los datos del usuario por id

GET /usuarios/{id}

```
/*
 * GET /usuarios/{id}
 * Devuelve los datos de un usuario por su id.
 */
0 references | 0 overrides
public function show(int $id): mixed
{
    $usuario = Usuario::find($id);

    if (!$usuario) {
        return response()->json(['error' => 'Usuario no encontrado'], 404);
    }

    return response([
        'id' => $usuario->id,
        'Nombre' => $usuario->Nombre,
        'Apellidos' => $usuario->Apellidos,
        'Email' => $usuario->Email,
        'Fecha_nacimiento' => $usuario->Fecha_nacimiento,
        'Tipo' => $usuario->Tipo,
    ]);
}
```

Ilustración 37 retornar usuario por id

-Actualizar Usuario

El método actualizar se encarga de actualizar los datos del usuario

PUT /usuarios/actualizar

```
public function actualizar(Request $request): mixed
{
    // Validar los datos recibidos
    $validator = Validator::make($request->all(), [
        'id_usuario' => 'required|exists:usuarios,id',
        'Nombre' => 'required|string|max:255',
        'Apellidos' => 'required|string|max:255',
        'Email' => 'required|email|max:255|unique:usuarios,Email,' . $request->input('id_usuario'),
        'Fecha_nacimiento' => 'required|date',
        'Tipo' => 'required|string|in:admin,usuario',
        'Password' => 'nullable|string|min:8',
    ]);

    if ($validator->fails()) {
        return response()->json(['errors' => $validator->errors()], 422);
    }

    // Buscar usuario y actualizar campos
    $usuario = Usuario::find($request->input('id_usuario'));

    if (!$usuario) {
        return response()->json(['error' => 'Usuario no encontrado'], 404);
    }

    $usuario->Nombre = $request->input('Nombre');
    $usuario->Apellidos = $request->input('Apellidos');
    $usuario->Email = $request->input('Email');
    $usuario->Fecha_nacimiento = $request->input('Fecha_nacimiento');
    $usuario->Tipo = $request->input('Tipo');

    // Actualizar contraseña si se proporciona
    if ($request->has('Password') && $request->input('Password') != null) {
        $usuario->Password = bcrypt($request->input('Password'));
    }

    $usuario->save();

    return response()->json(['usuario' => $usuario], 200);
}
```

Ilustración 38 actualizar usuario

-Delete Usuario

El método de Delete usuario se encarga de eliminar un usuario

DELETE /usuarios/{id}

```
/**  
 * DELETE /usuarios/{id}  
 * Elimina un usuario por su id.  
 */  
0 references | 0 overrides  
public function destroy(string $id): mixed  
{  
    $usuario = Usuario::find($id);  
  
    if (!$usuario) {  
        return response()->json(['error' => 'Usuario no encontrado'], 404);  
    }  
  
    $usuario->delete();  
  
    return response()->json(['message' => 'Usuario eliminado correctamente'], 200);  
}
```

Ilustración 39 Delete usuario

-PublicacionController.php

Controller de publicación el cual contiene los métodos necesarios para las solicitudes de los endpoints.

-Show Publicaciones

El método de show publicaciones se encarga de mostrar todos los datos de las publicaciones.

GET /usuarios

```
/*
 * GET /publicaciones
 * Devuelve todas las publicaciones.
 */
0 references | 0 overrides
public function index(): mixed
{
    $publicaciones = Publicacion::all();
    return response()->json(data: $publicaciones);
}
```

Ilustración 40 Show Publicaciones

-Obtener Paginadas (Protegido)

El método de obtener paginadas se encarga de retornar todas las publicaciones de forma paginada.

POST /publicacionesPaginadas

```
/*
 * POST /publicaciones/paginar
 * Devuelve publicaciones paginadas. Recibe 'pagina' en el body.
 */
1 reference | 0 overrides
public function obtenerPaginadas(Request $request): mixed
{
    // Constante de elementos por página
    $ELEMENTOS POR PAGINA = 8;

    // Obtener número de página desde la query (por defecto 1)
    $pagina = (int) $request->input('pagina', 1);

    try {
        // Obtener publicaciones con paginación
        $publicaciones = Publicacion::paginate($ELEMENTOS POR PAGINA, [ '*' ], 'page', $pagina);

        // Preparar respuesta con metadatos personalizados
        return response()->json([
            'success' => true,
            'data' => $publicaciones->items(),
            'page' => $publicaciones->currentPage(),
            'totalPages' => $publicaciones->lastPage()
        ]);
    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => 'Error al obtener publicaciones paginadas.',
            'error' => $e->getMessage()
        ], 500);
    }
}
```

Help us!

Ilustración 41 Obtener Paginadas

-Obtener Alquilables Paginados

El método de obtener alquilables paginados se encarga de retornar todas las publicaciones de tipo alquilable de forma paginada.

POST /alquilablesPaginados

```
/***
 * POST /alquilablesPaginados
 * Devuelve publicaciones de tipo 'alquilable' paginadas y con fecha futura.
 */
reference|0 overrides
public function obtenerAlquilablesPaginados(Request $request): mixed
{
    $ELEMENTOS POR PAGINA = 8;
    $pagina = (int) $request->input('pagina', 1);

    try {
        $publicaciones = Publicacion::where('tipo', 'alquilable')
            ->whereDate('fecha_evento', '>=', Carbon::today())
            ->paginate($ELEMENTOS POR PAGINA, ['*'], 'page', $pagina);

        return response()->json([
            'success' => true,
            'data' => $publicaciones->items(),
            'page' => $publicaciones->currentPage(),
            'totalPages' => $publicaciones->lastPage()
        ]);
    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => 'Error al obtener publicaciones alquilables paginadas.',
            'error' => $e->getMessage()
        ], 500);
    }
}
```

Ilustración 42 Obtener Alquilables Paginadas

-Obtener Informativos Paginados

El método de obtener informativos paginados se encarga de retornar todas las publicaciones de tipo informativo de forma paginada.

POST /informativosPaginados

```
/***
 * POST /informativosPaginados
 * Devuelve publicaciones de tipo 'informativo' paginadas y con fecha futura.
 */
reference|0 overrides
public function obtenerInformativosPaginados(Request $request): mixed
{
    $ELEMENTOS POR PAGINA = 8;
    $pagina = (int) $request->input('pagina', 1);

    try {
        $publicaciones = Publicacion::where('tipo', 'informativo')
            ->whereDate('fecha_evento', '>=', Carbon::today())
            ->paginate($ELEMENTOS POR PAGINA, ['*'], 'page', $pagina);

        return response()->json([
            'success' => true,
            'data' => $publicaciones->items(),
            'page' => $publicaciones->currentPage(),
            'totalPages' => $publicaciones->lastPage()
        ]);
    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => 'Error al obtener publicaciones informativas paginadas.',
            'error' => $e->getMessage()
        ], 500);
    }
}
```

Ilustración 43 Obtener Informativos Paginadas

-Obtener Informativos

El método de obtener informativos se encarga de retornar todas las publicaciones de tipo informativo.

GET /informativos

```
/**
 * GET /informativos
 * Devuelve todas las publicaciones de tipo 'informativo'.
 */
1 reference | 0 overrides
public function obtenerInformativos(): mixed
{
    $publicaciones = Publicacion::where('tipo', 'informativo')->get();

    if ($publicaciones->isEmpty()) {
        return response()->json([], 404);
    }

    return response()->json($publicaciones);
}
```

Ilustración 44 Obtener Informativos

-Obtener Alquilables

El método de obtener alquilables se encarga de retornar todas las publicaciones de tipo alquilable.

GET /alquilables

```
/**
 * GET /alquilables
 * Devuelve todas las publicaciones de tipo 'alquilable'.
 */
1 reference | 0 overrides
public function obtenerAlquilables(): mixed
{
    $publicaciones = Publicacion::where('tipo', 'alquilable')->get();

    if ($publicaciones->isEmpty()) {
        return response()->json([], 404);
    }

    return response()->json($publicaciones);
}
```

Ilustración 45 Obtener Alquilables

-Store Publicacion (Protegido)

El método de store se encarga de almacenar publicaciones.

POST /publicaciones

```
/*
 * POST /publicaciones
 * crea una nueva publicación con imágenes.
 */
0 references|0 overrides
public function store(Request $request): mixed
{
    log::info('Iniciando método store para crear publicación');

    try {
        log::info('Validando la solicitud...', ['request' => $request->all()]);
        $request->validate([
            'titulo' => 'required|string|max:255',
            'Descripción' => 'required|string|max:1000',
            'Fecha_evento' => 'required|date',
            'Tipo' => 'required|string|max:255',
            'Precio' => 'required|numeric',
            'Aforo_maximo' => 'required|numeric',
            'Imagenes' => 'required|array|max:4',
            'Imagenes.*' => 'image|mimes:jpg,jpeg,png,gif,webp|max:2048'
        ]);
        $publicacion = new Publicacion([
            'titulo' => $request->input('titulo'),
            'Descripción' => $request->input('Descripción'),
            'Fecha_evento' => $request->input('Fecha evento'),
            'Aforo_maximo' => $request->input('Aforo_maximo'),
            'Tipo' => $request->input('Tipo'),
            'Precio' => $request->input('Precio')
        ]);
        log::info('Datos de la publicación preparados.', ['publicacion' => $publicacion]);
        $imageController = new ImageController();
    }
}
```

Ilustración 46 Store de Publicación I

```
$imagePaths = [];
$uploadFailed = false;
$errorMessages = [];

foreach ($request->file('Imagenes') as $image) {
    log::info('Subiendo imagen...', ['Imagen' => $image->getClientOriginalName()]);
    $responseData = $imageController->uploadImage(file: $image);

    if (!$responseData['success']) {
        $uploadFailed = true;
        $errorMessages[] = 'No se pudo subir la imagen: ' . $image->getClientOriginalName();
        log::error('Fallo al subir imagen.', ['Imagen' => $image->getClientOriginalName()]);
    } else {
        $imagePaths[] = basename($responseData['path']);
        log::info('Imagen subida exitosamente.', ['Ruta' => $responseData['path']]);
    }
}

if ($uploadFailed) {
    log::warning('Error en la subida de imágenes.', ['errores' => $errorMessages]);
    return response()->json([
        'success' => false,
        'message' => 'Algunas imágenes no se pudieron subir.',
        'errors' => $errorMessages
    ], 400);
}

$publicacion->Imagen = implode(separator: ';', array: $imagePaths);
$publicacion->save();

log::info('Publicación guardada exitosamente.', ['id' => $publicacion->id]);
```

Ilustración 47 Store de Publicación II

```
return response()->json([
    'success' => true,
    'message' => 'Publicación y imágenes guardadas exitosamente.',
    'data' => $publicacion
]);
} catch (\Exception $e) {
    log::error('Error inesperado al guardar la publicación.', [
        'error' => $e->getMessage(),
        'trace' => $e->getTraceAsString()
    ]);

    return response()->json([
        'success' => false,
        'message' => 'Ocurrió un error inesperado.',
        'error' => $e->getMessage()
    ], 500);
}
```

Ilustración 48 Store de Publicación III

-Publicaciones Aleatorias

El método de store se encarga de mostrar publicaciones aleatoriamente mezclando informativas y reservables.

POST /publicaciones

```
/**
 * GET /publicacionesAleatorias
 * Devuelve 6 publicaciones aleatorias.
 */
1 reference | 0 overrides
public function obtenerPublicacionesAleatorias(): mixed
{
    // Retorna 6 publicaciones aleatorias
    $publicaciones = Publicacion::inRandomOrder()->limit(6)->get();

    return response()->json($publicaciones);
}
```

Ilustración 49 Publicaciones Aleatorias

-Update Publicaciones (Protegido)

El método de store se encarga de actualizar las publicaciones.

PUT /publicaciones

```
/*
 * PUT /publicaciones/actualizar
 * Actualiza una publicación existente y sus imágenes.
 */
1 reference | 0 overrides
public function update(Request $request): mixed
{
    log::info('Iniciando método update para actualizar publicación', ['request' => $request->all()]);

    try {
        // Validación de los datos
        $request->validate([
            'id_publicacion' => 'required|integer|exists:publicaciones,id',
            'Titulo' => 'required|string|max:255',
            'Descripcion' => 'required|string|max:1000',
            'Fecha_evento' => 'required|date',
            'Tipo' => 'required|string|max:255',
            'Precio' => 'required|numeric',
            'Aforo_maximo' => 'required|numeric',
            'Imagenes' => 'sometimes|array|max:4',
            'Imagenes.*' => 'image|mimes:jpg,jpeg,png,gif,webp|max:2048',
            'Imagenes_existentes' => 'sometimes|array',
            'Imagenes_existentes.*' => 'string'
        ]);
        $id = $request->input('id_publicacion');
        $publicacion = Publicacion::find($id);

        if (!$publicacion) {
            return response()->json([
                'success' => false,
                'message' => 'Publicación no encontrada.'
            ], 404);
        }
    }
```

Ilustración 50 Update de Publicaciones I

```

// obtener imágenes antiguas y existentes a conservar
$imagenesAntiguas = $publicacion->Imagen ? explode(separator: ';', string: $publicacion->Imagen) : [];
$imagenesExistentes = $request->input('imagenes_existentes', []);

// Asegurarse de que es array
if (is_string(value: $imagenesExistentes)) {
    $imagenesExistentes = explode(separator: ';', string: $imagenesExistentes);
}

// Determinar imágenes a eliminar
$imagenesAEliminar = array_diff(array: $imagenesAntiguas, arrays: $imagenesExistentes);

foreach ($imagenesAEliminar as $imagen) {
    $rutaImagen = public_path('storage/photos/' . basename(path: $imagen));
    if (file_exists(filename: $rutaImagen)) {
        unlink(filename: $rutaImagen);
    }
}

// Actualizar datos principales
$publicacion->titulo = $request->input('Titulo');
$publicacion->descripcion = $request->input('Descripcion');
$publicacion->fecha_evento = $request->input('fecha_evento');
$publicacion->tipo = $request->input('Tipo');
$publicacion->precio = $request->input('Precio');
$publicacion->foro_maximo = $request->input('Foro_maximo');

// Inicializar lista con las imágenes que se conservan
$imagePaths = $imagenesExistentes;

```

Ilustración 51 Update de Publicaciones II

```

// Subir nuevas imágenes si existen
if ($request->hasFile('imagenes')) {
    $imageController = new ImageController();
    $uploadedFailed = false;
    $serverMessages = [];

    foreach ($request->file('imagenes') as $image) {
        $responseData = $imageController->uploadImage(file: $image);

        if (!($responseData['success'])) {
            $uploadedFailed = true;
            $serverMessages[] = "No se pudo subir la imagen: " . $image->getClientOriginalName();
            $logError("Fallo al subir imagen.", ['Imagen' => $image->getClientOriginalName()]);
        } else {
            $imagePaths[] = basename(path: $responseData['path']);
            $logInfo("Imagen subida exitosamente.", ['ruta' => $responseData['path']]);
        }
    }
    if ($uploadedFailed)
        return response()->json(['success' => false, 'message' => 'Algunas imágenes no se pudieron subir.', 'errors' => $serverMessages], 400);
}

// Guardar ruta actualizada de imágenes
$publicacion->Imagen = implode(separator: ';', array: $imagePaths);
$publicacion->save();

return response()->json([
    'success' => true,
    'message' => 'Publicación actualizada exitosamente.',
    'data' => $publicacion
]);

} catch (\Exception $e) {
    return response()->json([
        'success' => false,
        'message' => 'Ocurrió un error inesperado.',
        'error' => $e->getMessage()
    ], 500);
}

```

Ilustración 52 Update de Publicaciones III

-Show Publicaciones (Protegido)

El método de show publicaciones se encarga de mostrar las publicaciones da igual el tipo por id.

GET /publicaciones/{id}

```

/**
 * GET /publicaciones/{id}
 * Devuelve los datos de una publicación por su id.
 */
0 references | 0 overrides
public function show(string $id): mixed
{
    $publicacion = Publicacion::find($id);

    if (!$publicacion) {
        return response()->json(['message' => 'Publicación no encontrada'], 404);
    }

    return response()->json($publicacion);
}

```

Ilustración 53 Show de Publicaciones

-Delete Publicaciones (Protegido)

El método de delete publicaciones se encarga de borrar las publicaciones por id.

DELETE /publicaciones/{id}

```
/*
 * DELETE /publicaciones/{id}
 * Elimina una publicación y sus imágenes asociadas.
 */
0 references|0 overrides
public function destroy(string $id): mixed
{
    $publicacion = Publicacion::find($id);

    if (!$publicacion)
        return response()->json(['message' => 'Publicación no encontrada.'], 404);

    try {
        // Eliminar imágenes asociadas
        if ($publicacion->imagen) {
            $imagenes = explode(separador: ';', string: $publicacion->imagen);

            foreach ($imagenes as $imagen) {
                $rutaImagen = public_path('storage/photos/' . basename(path: $imagen));
                if (file_exists(filename: $rutaImagen)) {
                    unlink(filename: $rutaImagen);
                }
            }
        }

        // Eliminar la publicación
        $publicacion->delete();

        return response()->json(['message' => 'Publicación e imágenes eliminadas correctamente.'], 200);
    } catch (\Exception $e) {
        return response()->json([
            'message' => 'Error al eliminar la publicación.',
            'error' => $e->getMessage()
        ], 500);
    }
}
```

Ilustración 54 Delete de Publicaciones

-ReservaController.php

Controller de reservas el cual contiene los métodos necesarios para las solicitudes de los enpoints.

-Show Reservas

El método de show reservas se encarga de mostrar todos los datos de las reservas.

GET /reservas

```
/** 
 * GET /reservas
 * Devuelve todas las reservas.
 */
0 references | 0 overrides
public function index(): mixed
{
    $reservas = Reserva::all();
    return response()->json($reservas);
}
```

Ilustración 55 Show Reservas

Paginar Reservas (Protegido)

El método de paginar reservas sirve para retornar todas las reservas pero de forma paginada.

POST /reservasPaginadas

```
/*
 * POST /reservasPaginadas
 * Devuelve reservas paginadas. Recibe 'pagina' en el body.
 */
1 reference | 0 overrides
public function paginarReservas(Request $request): mixed
{
    // Definir el número de elementos por página
    $perPage = 10;

    // Obtener la página desde el cuerpo de la solicitud, por defecto 1
    $page = (int) $request->input('pagina', 1);
    $page = max(value: $page, values: 1); // Evitar páginas menores a 1

    // Obtener el total de reservas
    $total = Reserva::count();

    // Calcular total de páginas
    $totalPages = (int) ceil(num: $total / $perPage);

    // Obtener las reservas de la página solicitada
    $reservas = Reserva::with(['usuario:id,nombre,Apellidos', 'publicacion:id,titulo'])
        ->skip(($page - 1) * $perPage)
        ->take($perPage)
        ->get()
        ->map(function ($reserva): array {
            return [
                'id' => $reserva->id,
                'nombre_usuario' => $reserva->usuario ? $reserva->usuario->Nombre . ' ' . $reserva->usuario->Apellidos : 'Desconocido',
                'titulo_publicacion' => $reserva->publicacion->titulo ?? 'Sin título',
                'fecha_reserva' => $reserva->fecha_reserva,
                'total_pagar' => $reserva->total_pagar ?? 0,
                'personas' => $reserva->personas,
            ];
        });
}
```

Ilustración 56 Reservas Paginadas

Obtener Reservas por Usuario

El método de obtener reservas por usuario retorna las reservas del usuario autenticado.

GET /obtenerReservasUsuario

```
/*
 * GET /obtenerReservasUsuario
 * Devuelve todas las reservas del usuario autenticado.
 */
1 reference | 0 overrides
public function getReservasPorUsuario(Request $request): mixed
{
    $usuario = Auth::guard('sanctum')->user();
    if (!$usuario) {
        return response()->json(['error' => 'Usuario no autenticado'], 401);
    }

    // Cargar reservas junto con el título de la publicación
    $reservas = $usuario->reservas()->with(['publicacion:id,titulo,imagen'])->get();

    return response()->json($reservas);
}
```

Ilustración 57 Reservas por Usuario

Actualizar Fecha Publicación (Protegido)

El método de actualizarFechaPublicacion actualiza la fecha de publicación y de todas sus reservas asociadas notificando al usuario por email.

POST /actualizarFechaPublicacion

```
/*
 * POST /actualizarFechaPublicacion
 * Actualiza la fecha de una publicación y de todas sus reservas asociadas, notificando a los usuarios.
 */
1 reference | 0 overrides
public function actualizarFechaPublicacionYReservas(Request $request): mixed
{
    $validated = $request->validate([
        'publicacion_id' => 'required|exists:publicaciones,id',
        'nueva_fecha_evento' => 'required|date_format:y-m-d', // Solo la fecha, sin hora
    ]);

    $publicacion = Publicacion::findOrFail($validated['publicacion_id']);
    $fechaAnterior = $publicacion->fecha_evento;
    $nuevaFechaEvento = $validated['nueva_fecha_evento'];

    // Actualizar la fecha del evento de la publicación
    $publicacion->fecha_evento = $nuevaFechaEvento;
    $publicacion->save();

    // Obtener todas las reservas relacionadas con esta publicación
    $reservas = Reserva::with('usuario')
        ->where('publicacion_id', $publicacion->id)
        ->get();

    $reservasActualizadas = [];
    foreach ($reservas as $reserva) {
        // Extraer la hora original de la reserva
        $horaOriginal = Carbon::parse($reserva->fecha_reserva)->format('H:i:s');

        // Formar la nueva fecha completa
        $nuevaFechaCompleta = $nuevaFechaEvento . ' ' . $horaOriginal;

        // Actualizar la reserva
        $reserva->fecha_reserva = $nuevaFechaCompleta;
        $reserva->save();
    }
}
```

Ilustración 58 Actualizar Fecha Publicación I

```

    // Guardar para notificación
    $reservasActualizadas[] = [
        'usuario' => $reserva->usuario,
        'nueva_fecha' => $nuevaFechaCompleta
    ];
}

// Enviar correos a los usuarios afectados
foreach ($reservasActualizadas as $item) {
    $usuario = $item['usuario'];
    $nuevaFecha = $item['nueva_fecha'];

    if (!empty($usuario->Email)) {
        $mensaje = "Hola {$usuario->Nombre}, la fecha de tu reserva ha cambiado a {$nuevaFecha}. Gracias por tu comprensión.";
        Mail::to($usuario->Email)->send(new CorreoPersonalizado(mensaje: $mensaje));
    }
}

return response()->json([
    'message' => 'Fecha del evento y reservas actualizadas correctamente',
    'nueva_fecha_evento' => $nuevaFechaEvento,
    'total_reservas_afectadas' => count(value: $reservasActualizadas)
]);
}

```

Ilustración 59 Actualizar Fecha Publicación II

Obtener reservas por usuario

El método obtener reservas por usuario retorna las reservas por id de usuario.

GET /obtenerReservasUsuario/{id}

```

/**
 * GET /obtenerReservasUsuario/{id}
 * Devuelve todas las reservas de un usuario por su ID.
 */
1 reference | 0 overrides
public function getReservasPorIdUsuario($id): mixed
{
    $usuario = \App\Models\Usuario::find($id);

    if (!$usuario) {
        return response()->json(['error' => 'Usuario no encontrado'], 404);
    }

    // Cargar reservas junto con el título de la publicación
    $reservas = $usuario->reservas()->with(['publicacion:id,titulo'])->get();

    return response()->json($reservas);
}

```

Ilustración 60 Obtener reservas por usuario

Store de Reservas

El método store de Reservas sirve para almacenar reservas.

POST /reservas

```
/**
 * POST /reservas
 * Crea una nueva reserva para el usuario autenticado.
 */
0 references | 0 overrides
public function store(Request $request): mixed
{
    // Validar sin usuario_id
    $validated = $request->validate([
        'publicacion_id' => 'required|exists:publicaciones,id',
        'hora_reserva' => 'required|date_format:H:i', // por ejemplo '16:00'
        'total_pagar' => 'nullable|integer',
        'personas' => 'nullable|integer|min:1|max:4',
    ]);

    // Obtener el usuario autenticado
    $usuario = auth()->user();

    if (!$usuario) {
        return response()->json(['error' => 'Usuario no autenticado'], 401);
    }

    // Obtener la publicación
    $publicacion = Publicacion::findOrFail($validated['publicacion_id']);

    // Formar fecha completa con fecha_evento + hora_reserva
    $fechaEvento = \Illuminate\Support\Carbon::parse($publicacion->fecha_evento)->format('Y-m-d');
    $hora = str_pad(string: substr(string: $validated['hora_reserva'], offset: 0, length: 2), length: 2, pad: '0', STR_PAD_LEFT) . ':00:00';
    $fechaReserva = $fechaEvento . ' ' . $hora;

    // Crear reserva
    $reserva = Reserva::create([
        'usuario_id' => $usuario->id,
        'publicacion_id' => $validated['publicacion_id'],
        'fecha_reserva' => $fechaReserva,
        'total_pagar' => $validated['total_pagar'] ?? null,
        'personas' => $validated['personas'] ?? null,
    ]);
}

return response()->json([
    'message' => 'Reserva creada correctamente',
    'reserva' => $reserva
], 201);
```

Ilustración 61 Store de Reservas I

```
return response()->json([
    'message' => 'Reserva creada correctamente',
    'reserva' => $reserva
], 201);
```

Ilustración 62 Store de Reservas II

Capacidad disponible de las reservas

El método que sirve para comprobar la capacidad disponible de las reservas en todo momento.

POST /capacidadDisponible

```
/**
 * POST /capacidadDisponible
 * Devuelve la capacidad disponible para una publicación.
 */
reference|0 overrides
public function getCapacidadDisponible(Request $request): mixed
{
    $idPublicacion = $request->input('idPublicacion');

    if (!$idPublicacion) {
        return response()->json(['error' => 'ID de publicación requerido'], 400);
    }

    $publicacion = Publicacion::find($idPublicacion);

    if (!$publicacion) {
        return response()->json(['error' => 'Publicación no encontrada'], 404);
    }

    $aforoMaximo = $publicacion->aforo_maximo;

    // Suma de todas las personas ya reservadas
    $personasReservadas = Reserva::where('publicacion_id', $idPublicacion)
        ->sum('personas');

    // Cálculo de personas disponibles
    $personasDisponibles = max(value: $aforoMaximo - $personasReservadas, values: 0);

    // El usuario puede reservar entre 1 y 4 personas, sin superar la capacidad restante
    $maxReservables = min(value: 4, values: $personasDisponibles);
    $minReservables = $maxReservables > 0 ? 1 : 0;

    return response()->json([
        'aforo_maximo' => $aforoMaximo,
        'personas_reservadas' => $personasReservadas,
        'personas_disponibles' => $personasDisponibles,
        'min_reservables' => $minReservables,
        'max_reservables' => $maxReservables
    ]);
}
```

Ilustración 63 Capacidad Disponible de las Publicaciones

Disponibilidad de las Reservas

El método que sirve para comprobar la disponibilidad de las reservas en todo momento.

POST /disponibilidadReserva

```
/**
 * POST /disponibilidadReserva
 * Comprueba si una hora está disponible para reservar en una publicación.
 */
reference|0 overrides
public function getDisponibilidad(Request $request): mixed
{
    $idPublicacion = $request->input('idPublicacion');
    $horaReserva = $request->input('horaReserva'); // Por ejemplo "23"

    // Validación básica
    if ($idPublicacion == null) {
        return response()->json(['error' => 'Datos incompletos'], 400);
    }

    // Buscar publicación
    $publicacion = Publicacion::find($idPublicacion);

    if (!$publicacion) {
        return response()->json(['error' => 'Publicación no encontrada'], 404);
    }

    // Obtener fecha del evento de la publicación
    $fechaEvento = Carbon::parse($publicacion->fecha_evento)->format('Y-m-d');

    // Formatear hora
    $horaFormateada = str_pad(string: $horaReserva, length: 2, pad_string: '0', pad_type: STR_PAD_LEFT) . ':00:00';

    // Componer fecha y hora completa
    $fechaHoraCompleta = $fechaEvento . ' ' . $horaFormateada;

    // Verificar si existe reserva exacta en ese datetime
    $existeReserva = Reserva::where('publicacion_id', $idPublicacion)
        ->where('fecha_reserva', $fechaHoraCompleta)
        ->exists();

    return response()->json([
        'fecha_reserva_comprobada' => $fechaHoraCompleta,
        'disponible' => !$existeReserva,
    ]);
}
```

Ilustración 64 Disponibilidad de las Reservas

Obtener Reservas por Id

El método que sirve para obtener las reservas por Id.

GET /reservas/{id}

```
/** 
 * GET /reservas/{id}
 * Devuelve los datos de una reserva por su ID.
 */
0 references|0 overrides
public function show(string $id): mixed
{
    $reserva = Reserva::with(['usuario:id,Nombre,Apellidos', 'publicacion:id,titulo'])
        ->find($id);

    if (!$reserva) {
        return response()->json(['error' => 'Reserva no encontrada'], 404);
    }

    $nombreCompleto = $reserva->usuario
        ? $reserva->usuario->Nombre . ' ' . $reserva->usuario->Apellidos
        : 'Desconocido';

    return response()->json([
        'id' => $reserva->id,
        'usuario_id' => $reserva->usuario->id ?? null,
        'publicacion_id' => $reserva->publicacion->id ?? null,
        'nombre_usuario' => $nombreCompleto,
        'titulo_publicacion' => $reserva->publicacion->titulo ?? 'Sin título',
        'fecha_reserva' => $reserva->fecha_reserva,
        'total_pagar' => $reserva->total_pagar ?? 0,
        'personas' => $reserva->personas,
    ]);
}
```

Ilustración 65 Show de Reservas

Update de Reservas

El método que sirve para actualizar las Reservas.

PUT /reservas

```
/** 
 * PUT /reservas
 * Actualiza los datos de una reserva.
 */
1 reference|0 overrides
public function update(Request $request): mixed
{
    // Validar los datos, incluyendo el ID de la reserva
    $validated = $request->validate([
        'id' => 'required|exists:reservas,id',
        'usuario_id' => 'required|exists:usuarios,id',
        'publicacion_id' => 'required|exists:publicaciones,id',
        'fecha_reserva' => 'required|date_format:Y-m-d H:i:s',
        'total_pagar' => 'nullable|numeric',
        'personas' => 'nullable|integer|min:1|max:4',
    ]);

    // Buscar la reserva
    $reserva = Reserva::find($validated['id']);

    if (!$reserva) {
        return response()->json(['error' => 'Reserva no encontrada'], 404);
    }

    // Actualizar campos
    $reserva->usuario_id = $validated['usuario_id'];
    $reserva->publicacion_id = $validated['publicacion_id'];
    $reserva->fecha_reserva = $validated['fecha_reserva'];
    $reserva->total_pagar = $validated['total_pagar'] ?? null;
    $reserva->personas = $validated['personas'] ?? null;

    $reserva->save();

    return response()->json([
        'message' => 'Reserva actualizada correctamente',
        'reserva' => $reserva
    ]);
}
```

Ilustración 66 Update de Reservas

Intercambiar Reservas (Protegido)

El método que sirve para intercambiar la fecha entre dos reservas.

POST /intercambiarFechas

```
/*
 * POST /intercambiarFechas
 * Intercambia la fecha de dos reservas y notifica a los usuarios si se solicita.
 */
1 reference | 0 overrides
public function intercambiarReserva(Request $request): mixed
{
    // Validar los datos del request
    $validated = $request->validate([
        'id' => 'required|exists:reservas,id', // ID de la reserva que se quiere cambiar
        'nueva_fecha_reserva' => 'required|date_format:Y-m-d H:i:s', // Nueva fecha de reserva
        'correo' => 'nullable|boolean' // Si se debe enviar correo
    ]);

    // Obtener la reserva original
    $reservaOriginal = Reserva::with('usuario')->findOrFail($validated['id']);

    // Buscar una reserva para intercambiar (misma publicación y fecha nueva)
    $reservaDestino = Reserva::with('usuario')
        ->where('fecha_reserva', $validated['nueva_fecha_reserva'])
        ->where('publicacion_id', $reservaOriginal->publicacion_id)
        ->first();

    // Si no existe la reserva destino con la nueva fecha, devolver error
    if (!$reservaDestino) {
        return response()->json(['error' => 'No existe una reserva con esa fecha para intercambiar'], 404);
    }

    // Intercambiar las fechas de las reservas
    $fechaTemp = $reservaOriginal->fecha_reserva;
    $reservaOriginal->fecha_reserva = $reservaDestino->fecha_reserva;
    $reservaDestino->fecha_reserva = $fechaTemp;

    // Guardar las reservas con las nuevas fechas
    $reservaOriginal->save();
    $reservaDestino->save();
}
```

Ilustración 67 Intercambiar Fechas I

```
// Si el parámetro 'correo' es true, enviar correos a los usuarios
if ($request->has('correo') && $request->correo) {
    $usuario1 = $reservaOriginal->usuario;
    $usuario2 = $reservaDestino->usuario;

    // Verificar si ambos usuarios tienen correo
    if (!empty($usuario1->Email) && !empty($usuario2->Email)) {
        $mensaje1 = "Hola {$usuario1->Nombre}, tu reserva ha sido cambiada al horario {$reservaOriginal->fecha_reserva}." ;
        $mensaje2 = "Hola {$usuario2->Nombre}, tu reserva ha sido cambiada al horario {$reservaDestino->fecha_reserva}.";

        // Enviar los correos a ambos usuarios
        Mail::to($usuario1->Email)->send(new CorreoPersonalizado(mensaje: $mensaje1));
        Mail::to($usuario2->Email)->send(new CorreoPersonalizado(mensaje: $mensaje2));
    } else {
        return response()->json(['error' => 'Uno o ambos usuarios no tienen un Email asignado.'], 422);
    }
}

// Respuesta exitosa
return response()->json([
    'message' => 'Reserva intercambiada correctamente',
    'reserva_1' => $reservaOriginal,
    'reserva_2' => $reservaDestino
]);
```

Ilustración 68 Intercambiar Fechas II

Obtener Reservas Detalladas (Protegido)

El método que sirve para obtener las reservas de forma detallada, obteniendo el usuario y la publicación.

POST /obtenerReservasDetalladas

```
/**
 * GET /obtenerReservasDetalladas
 * Devuelve todas las reservas con información detallada de usuario y publicación.
 */
0 references | 0 overrides
public function obtenerReservasDetalladas(): mixed
{
    $reservas = Reserva::with(['usuario:id,Nombre,Apellidos', 'publicacion:id,titulo'])
        ->get()
        ->map(function ($reserva): array {
            $nombreCompleto = $reserva->usuario
                ? $reserva->usuario->Nombre . ' ' . $reserva->usuario->Apellidos
                : 'Desconocido';

            return [
                'id' => $reserva->id,
                'nombre_usuario' => $nombreCompleto,
                'titulo_publicacion' => $reserva->publicacion->título ?? 'Sin título',
                'fecha_reserva' => $reserva->fecha_reserva,
                'estado' => Carbon::parse($reserva->fecha_reserva)->isFuture() ? 'pendiente' : 'pasada',
                'total_pagar' => $reserva->total_pagar ?? 0,
            ];
        });

    return response()->json($reservas);
}
/**
```

Ilustración 69 Obtener Reservas Detalladas

Delete de Reservas

Este método sirve para eliminar una reserva por id.

DELETE /reservas/{id}

```
/**
 * DELETE /reservas/{id}
 * Elimina (cancela) una reserva por su ID.
 */
0 references | 0 overrides
public function destroy(string $id): mixed
{
    $reserva = Reserva::find($id);

    if (!$reserva) {
        return response()->json(['error' => 'Reserva no encontrada'], 404);
    }

    $reserva->delete();

    return response()->json(['mensaje' => 'Reserva cancelada correctamente']);
}
```

Ilustración 70 Delete de Reservas

VISTAS

Las vistas son usadas para enviar mails y mostrar páginas desde el servidor

-Vista Email Personalizado

Es una vista de blade usada para enviar mails.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Correo Personalizado</title>
</head>
<body style="margin: 0; padding: 0; background-color: #f9fafb; font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;">
    <div style="max-width: 600px; margin: 40px auto; background-color: white; border-radius: 8px; overflow: hidden; box-shadow: 0 10px 15px rgba(0,0,0,0.1); padding: 20px 30px;">
        <div style="background-color: #1d4ed8; padding: 20px 30px;">
            <h2 style="color: white; font-size: 24px; margin: 0;">Mensaje desde Marina Rent</h2>
        </div>

        <div style="padding: 30px;">
            <p style="font-size: 16px; color: #111827; margin-bottom: 20px;">
                Hola 🌟,
            </p>

            <p style="font-size: 16px; color: #374151; margin-bottom: 30px;">
                {{ $mensaje }}
            </p>
        </div>

        <div style="background-color: #f3f4f6; padding: 20px 30px; text-align: center; font-size: 12px; color: #6b7280;">
            Este mensaje fue generado automáticamente por el sistema de Marina Rent.
        </div>
    </div>
</body>
</html>
```

Ilustración 71 emailpersonalizado.blade.php

-Vista Password Reset Success

Es una vista de blade usada para mostrar que la contraseña del email ha sido restablecida correctamente.

```
<!DOCTYPE html>
<html lang="es" class="h-full bg-indigo-50">
<head>
    <meta charset="UTF-8">
    <title>Éxito</title>
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="h-full flex items-center justify-center px-4 py-12">
    <div class="w-full max-w-md bg-white rounded-2xl shadow-xl overflow-hidden">
        <div class="bg-gray-300 px-6 py-8 flex flex-col items-center">
            
            <h2 class="text-2xl font-bold text-green-600">Contraseña restablecida!</h2>
            <p class="mt-2 text-gray-700 text-center">Puedes cerrar esta pestaña</p>
        </div>
    </div>
</body>
</html>
```

Ilustración 72 password-reset-success.blade.php

-Vista Reset Password

Es la vista que se muestra a la hora de restablecer contraseñas.

```
<!DOCTYPE html>
<html lang="es" class="h-full bg-indigo-50">
  <head>
    <meta charset="UTF-8">
    <title>Restablecer Contraseña</title>
    <script src="https://cdn.tailwindcss.com"></script>
  </head>
  <body class="h-full flex items-center justify-center px-4 py-12">
    <!-- Card -->
    <div class="w-full max-w-md bg-white rounded-2xl shadow-xl overflow-hidden">
      <!-- Header -->
      <div class="bg-gray-30 px-6 py-8 flex flex-col items-center">
        
        <h2 class="text-2xl font-bold text-gray-800">Restablecer contraseña</h2>
      </div>

      <!-- Form -->
      <div class="px-6 py-8">
        <form method="POST" action="{{ route('restablecer.password') }}" class="space-y-6">
          @csrf
          <input type="hidden" name="token" value="{{ $token }}>
          <input type="hidden" name="email" value="{{ $email }}>

          <div>
            <label for="password" class="block text-sm font-medium text-gray-700">Nueva Contraseña</label>
            <div class="mt-1">
              <input
                type="password"
                name="password"
                id="password"
                required
                class="block w-full bg-gray-50 border border-gray-200 rounded-lg px-3 py-2 text-gray-900 placeholder-gray-400 focus:outline-none focus:ring-gray-300">
            </div>
          </div>
        </form>
      </div>
    </div>
  </body>
</html>
```

Ilustración 73 reset.blade.php I

```
<div>
  <label for="password_confirmation" class="block text-sm font-medium text-gray-700">Confirmar Contraseña</label>
  <div class="mt-1">
    <input
      type="password"
      name="password_confirmation"
      id="password_confirmation"
      required
      class="block w-full bg-gray-50 border border-gray-200 rounded-lg px-3 py-2 text-gray-900 placeholder-gray-400 focus:outline-none focus:ring-gray-300">
  </div>
</div>

<div>
  <button
    type="submit"
    class="w-full flex justify-center bg-indigo-600 hover:bg-indigo-700 text-white font-semibold py-2 rounded-lg shadow-sm focus:outline-none focus:ring-gray-300">
    Restablecer Contraseña
  </button>
</div>
</form>

</div>
</div>
</body>
</html>
```

Ilustración 74 reset.blade.php II

ENV

Los .env son ficheros de configuración de variables de entorno que se encuentran ocultos en la estructura del proyecto con un (.) delante de su nombre.

```

LOG_DEPRECATED_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=marina
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME="manuel5365.mgl@gmail.com"
MAIL_PASSWORD="qqxt exwz cagi zkht"
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="marinaRent@gmail.com"
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

```

Ilustración 75 Fichero Env I

```

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1

VITE_APP_NAME="${APP_NAME}"
VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
VITE_PUSHER_HOST="${PUSHER_HOST}"
VITE_PUSHER_PORT="${PUSHER_PORT}"
VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

```

Ilustración 76 Fichero Env II

Cliente

ENRUTADOR – App.jsx

Esta página es la encargada de hacer posible la navegación por la página ya que es aquella que decide en qué página vamos a estar y en qué momento.

Para ello usa <Router> con sus <Routes> en la que cada <Route> es una página diferente es por ello que podemos cambiar entre diferentes vistas sin tener que hacer cargas de redirección con el navegador.

Además se encarga de comprobar si el usuario es Administrador o incluso si el authToken existe.

```
import { BrowserRouter as Router, Route, Routes, useLocation, Navigate } from 'react-router-dom';
import { useState, useEffect } from 'react';
import Navbar from './components/Navbar';
import Footer from './components/Footer';
import Home from './pages/Home';
import Login from './pages/Login';
import Informativos from './pages/Informativos';
import Alquilables from './pages/Alquilables';
import Mostrador from './pages/Mostrador';
import Carrito from './pages/Carrito';
import NotFound from './pages/NotFound';
import Reservas from './pages/Reservas';
import Perfil from './pages/Perfil';
import Admin from './pages/Admin';
import Registro from './pages/Register';
import CohereChat from './pages/ai';
import RecuperarPassword from './pages/ResetPass';
import { API_URL } from './utilities/apirest';
import axios from 'axios';

/**
 * Componente principal de la aplicación.
 * Configura el router, la autenticación, el carrito y la lógica de rutas protegidas.
 * @returns {JSX.Element}
 */
function App() {
  return (
    <Router>
      <AppContent />
    </Router>
  );
}
```

Ilustración 77 App.jsx /

```

/**
 * Componente que contiene la lógica de rutas y estado global.
 * Maneja autenticación, rutas protegidas, estado de admin y carrito.
 * @returns {JSX.Element}
 */
function AppContent() {
  const location = useLocation();
  // Determina si el usuario está autenticado
  const isAuthenticated = localStorage.getItem("authToken");
  // Estado del carrito de compras
  const [cart, setCart] = useState([]);
  // Estado para saber si el usuario es admin
  const [admin, setAdmin] = useState();

  /**
   * Carga el carrito desde localStorage al montar el componente.
   */
  useEffect(() => {
    const storedCart = localStorage.getItem("cart");
    if (storedCart) {
      setCart(JSON.parse(storedCart));
    }
  }, []);

  /**
   * Guarda el carrito en localStorage cada vez que cambia.
   */
  useEffect(() => {
    if (cart.length > 0) {
      localStorage.setItem("cart", JSON.stringify(cart));
      console.log("Carrito actualizado:", cart);
    }
  }, [cart]); // Se activa cada vez que el carrito cambia
}

```

Ilustración 78 App.jsx II

```

/**
 * Comprueba si el usuario autenticado es admin.
 * Actualiza el estado 'admin' según la respuesta de la API.
 */
useEffect(() => {
  const token = localStorage.getItem("authToken");
  if (token != null) {
    const headers = token ? { Authorization: `Bearer ${token}` } : {};
    axios.get(API_URL + "api/isAdmin", { headers })
      .then((response) => {
        console.log(response)
        if (response.status === 200)
          setAdmin(response.data.is_admin)
        else setAdmin(false)
      })
      .catch((error) => {
        console.error("Error al cargar los informativos:", error);
        setAdmin(false)
      })
  }
}, [isAuthenticated])

```

Ilustración 79 App.jsx III

```

return (
  <>
    {/* Navbar solo si no estamos en login, registro o recuperar */}
    {location.pathname !== "/" && location.pathname !== "/registro" && location.pathname !== "/recuperar" && <Navbar admin={admin} />}

  <Routes>
    {/* Ruta de login */}
    <Route
      path="/"
      element={!isAuthenticated ? <Login /> : <Navigate to="/home" replace />}
    />
    {/* Ruta de registro */}
    <Route
      path="/registro"
      element={isAuthenticated ? <Registro /> : <Navigate to="/home" replace />}
    />
    {/* Ruta de recuperación de contraseña */}
    <Route
      path="/recuperar"
      element={!isAuthenticated ? <RecuperarPassword /> : <Navigate to="/home" replace />}
    />
    {/* Ruta principal (home) */}
    <Route
      path="/home"
      element={isAuthenticated ? <Home /> : <Navigate to="/" replace />}
    />
    {/* Ruta de página no encontrada */}
    <Route
      path="/NotFound"
      element={isAuthenticated ? <NotFound /> : <Navigate to="/" replace />}
    />
    {/* Ruta de informativos */}
    <Route
      path="/informativos"
      element={isAuthenticated ? <Informativos /> : <Navigate to="/" replace />}
    />
)

```

Ilustración 80 App.jsx IV

```

  {/* Ruta de productos alquilables */}
  <Route
    path="/alquilables"
    element={isAuthenticated ? <Alquilables /> : <Navigate to="/" replace />}
  />
  {/* Ruta de mostrador, requiere autenticación y pasa el carrito */}
  <Route
    path="/mostrador"
    element={isAuthenticated ? <Mostrador cart={cart} setCart={setCart} /> : <Navigate to="/" replace />}
  />
  {/* Ruta del carrito */}
  <Route
    path="/carrito"
    element={isAuthenticated ? <Carrito cart={cart} setCart={setCart} /> : <Navigate to="/" replace />}
  />
  {/* Ruta de reservas */}
  <Route
    path="/reservas"
    element={isAuthenticated ? <Reservas cart={cart} setCart={setCart} /> : <Navigate to="/" replace />}
  />
  {/* Ruta de perfil */}
  <Route
    path="/perfil"
    element={isAuthenticated ? <Perfil /> : <Navigate to="/" replace />}
  />
  {/* Ruta de chat IA */}
  <Route
    path="/iacohere"
    element={isAuthenticated ? <CohereChat /> : <Navigate to="/" replace />}
  />
  {/* Ruta de administración, solo para admins */}
  <Route
    path="/admin"
    element={
      isAuthenticated
        ? admin === undefined
          ? <div className="flex justify-center items-center"><div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full a

```

Ilustración 81 App.jsx V

```

    {/* Footer solo si no estamos en login, registro o recuperar */}
    {location.pathname !== "/" && location.pathname !== "/registro" && location.pathname !== "/recuperar" && <Footer />}
  );
}

export default App;

```

Ilustración 82 App jsx VI

VISTAS / PÁGINAS

Login.jsx

El login es aquella ventana definida en el enrutador que solo se muestra en caso de que no haya authtoken existente o no se haya iniciado sesión.

Esta ventana realiza una petición al server a la hora de dar en iniciar sesión , cuando lo logra hacer redirige a la página de Login.

En caso de 3 intentos fallidos de peticiones de inicio de sesión nos muestra un mensaje de ¿olvidaste la contraseña?, si hacemos click nos redirigirá a recuperar contraseña.

Esta página también nos permite navegar a la ventana del register en caso de que no tengamos cuenta.

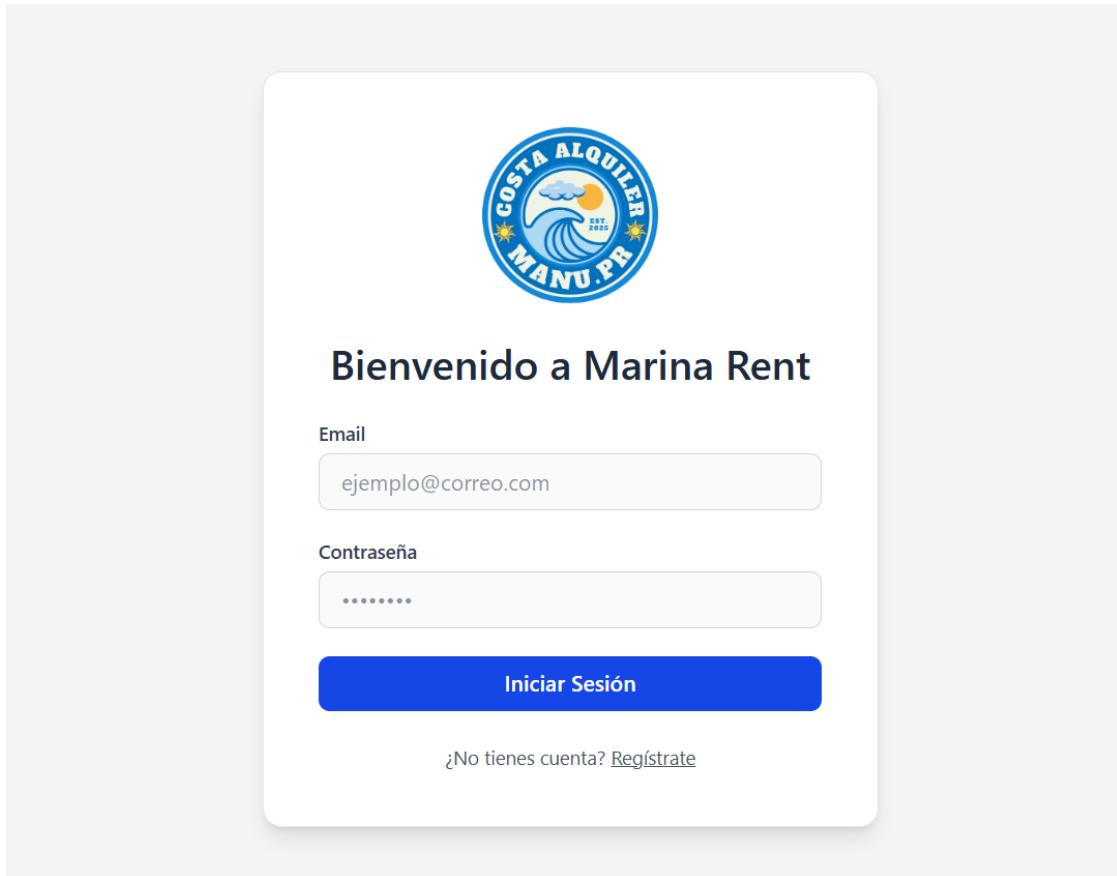


Ilustración 83 Vista de Login

Código – Login

```
/*
 * @file Login.jsx
 * @description Página de inicio de sesión para usuarios de Marina Rent.
 * Permite autenticarse, muestra errores y ofrece enlace de recuperación tras varios intentos fallidos.
 */

import { useState } from "react"
import { useNavigate } from "react-router-dom"
import { API_URL } from "../utilities/apirest"
import axios from "axios"

/**
 * Componente de formulario de login.
 * Gestiona el estado del usuario, errores, intentos fallidos y navegación.
 * @component
 * @returns {JSX.Element}
 */
function Login() {
  /**
   * Email introducido por el usuario.
   * ((string|Function)[])
   */
  const [email, setEmail] = useState("")
  /**
   * Contraseña introducida por el usuario.
   * ((string|Function)[])
   */
  const [password, setPassword] = useState("")
  /**
   * Mensaje de error a mostrar.
   * {[string|null, Function]}
   */
  const [errorMessage, setErrorMessage] = useState(null)
  /**
   * Hook de navegación de React Router.
   */
  const navigate = useNavigate()
  /**
   * Estado de carga mientras se realiza la petición.
   * {[boolean, Function]}
   */
}
```

Ilustración 84 Login.jsx I

```
const [isLoading, setIsLoading] = useState(false)
/**
 * Número de intentos fallidos de login.
 * {[number, Function]}
 */
const [failedAttempts, setFailedAttempts] = useState(0)
/**
 * Muestra el enlace de recuperación tras varios intentos fallidos.
 * {[boolean, Function]}
 */
const [showRecoveryLink, setShowRecoveryLink] = useState(false)
```

Ilustración 85 Login.jsx II

```
/** 
 * Envía la solicitud de login a la API.
 * Si el login es correcto, guarda el token y navega a /home.
 * Si falla, muestra mensaje de error y, tras 3 intentos, muestra enlace de recuperación.
 * @param {React.FormEvent} event
 */
function realizarSolicitud(event) {
  setIsLoading(true)
  event.preventDefault()
  const url = API_URL + "api/login"
  axios
    .post(url, { email: email, password: password })
    .then((response) => {
      if (response.status == 200) {
        const token = response.data.token.split("|")[1]
        localStorage.setItem("authToken", token)
        setErrorMessage(null)
        setFailedAttempts(0)
        navigate("/home")
      }
    })
    .catch((error) => {
      if (error.response) {
        const newFailedAttempts = failedAttempts + 1
        setFailedAttempts(newFailedAttempts)

        if (newFailedAttempts >= 3) {
          setShowRecoveryLink(true)
        }

        setErrorMessage("Credenciales incorrectas. Inténtalo de nuevo.")
      } else {
        setErrorMessage("Error de conexión. Por favor, intenta más tarde.")
      }
    })
    .finally(() => {
      setIsLoading(false)
    })
}

```

Ilustración 86 Login.jsx III

```
/** 
 * Navega a la página de recuperación de contraseña.
 */
function handleRecoveryClick() {
  navigate("/recuperar")
}

return (
  <div className="min-h-screen flex items-center justify-center bg-gray-100 px-4">
    <div className="w-full max-w-md p-10 bg-white rounded-lg shadow-lg border border-gray-200">
      {/* Logo de la aplicación */}
      <div className="flex justify-center mb-6">
        
      </div>

      <h2 className="text-center text-3xl font-semibold text-gray-800 mb-6">Bienvenido a Marina Rent</h2>

      {/* Mensaje de error si existe */}
      {errorMessage && (
        <div className="text-sm text-center text-red-700 bg-red-100 rounded py-2 mb-4">{errorMessage}</div>
      )}

      {/* Enlace de recuperación tras varios intentos fallidos */}
      {showRecoveryLink && (
        <div className="text-sm text-center text-blue-700 bg-blue-50 rounded py-2 mb-4">
          ¿Olvidaste tu contraseña?
          <span onClick={handleRecoveryClick}>
            Recuperar contraseña
          </span>
        </div>
      )}
    </div>
  </div>
)

```

Ilustración 87 Login.jsx IV

```
/* Formulario de login */
<form onSubmit={realizarSolicitud} className="space-y-5">
  <div>
    <label className="block text-sm font-medium text-gray-700 mb-1">Email</label>
    <input
      type="email"
      className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 transition-colors"
      placeholder="ejemplo@correo.com"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
    />
  </div>

  <div>
    <label className="block text-sm font-medium text-gray-700 mb-1">Contraseña</label>
    <input
      type="password"
      className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 transition-colors"
      placeholder="*****"
      value={password}
      onChange={(e) => setPassword(e.target.value)}
    />
  </div>

  <button
    type="submit"
    className="w-full py-2 bg-blue-700 text-white font-semibold rounded-md hover:bg-blue-800 transition-colors">
    {isLoading ? (
      <>
        <div className="flex justify-center items-center">
          <span className="h-6 w-6 border-2 border-white border-t-transparent rounded-full animate-spin" style={{ border: `2px solid ${isSubmitting ? '#000' : '#fff'}` }}></span>
        </div>
      </>
    ) : (
      <span>Iniciar Sesión</span>
    )}
  </button>
</form>
```

Ilustración 88 Login.jsx V

```
</form>

/* Enlace para registro de nuevos usuarios */
<p className="text-center text-sm text-gray-600 mt-6">
  ¿No tienes cuenta?
  <span onClick={() => navigate("/registro")}>Regístrate</span>
</p>
</div>
</div>
}

export default Login
```

Ilustración 89 Login.jsx VI

ResetPass.jsx

Esta clase es la encargada de enviar un mail a la dirección de correo indicada en el input text, al hacerlo nos muestra un mensaje de que el correo ha sido enviado correctamente.

Para acceder a esta vista tendremos que fallar 3 veces la contraseña en la ventana del Login.

Bienvenido a Marina Rent

Credenciales incorrectas. Inténtalo de nuevo.

¿Olvidaste tu contraseña? [Recuperar contraseña](#)

Email
manuel5365.mgl@gmail.com

Contraseña
...

Iniciar Sesión

¿No tienes cuenta? [Regístrate](#)

Recuperar Contraseña

Ingresá tu correo electrónico y te enviaremos un enlace para restablecer tu contraseña.

Email
manuel5365.mgl@gmail.com

Enviar Instrucciones

¿Ya recordaste tu contraseña? [Iniciar Sesión](#)

Ilustración 90 Reset Pass I

Ilustración 91 Vista Reset Pass II

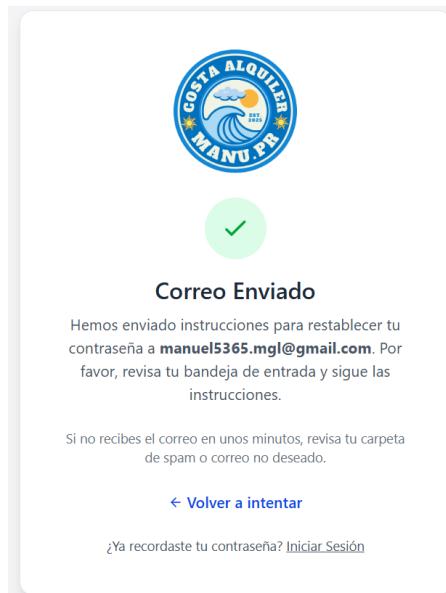


Ilustración 92 Reset Pass III

Código - ResetPass

```
/**  
 * @file ResetPass.jsx  
 * @description Página para recuperación de contraseña. Permite al usuario solicitar el envío de un correo para restablecer su contraseña.  
 */  
  
import { useState } from "react"  
import { ArrowLeft, Send } from "lucide-react"  
import { useNavigate } from "react-router-dom"  
import axios from "axios"  
import { API_URL } from "../utilities/apirest"  
  
/**  
 * Componente de recuperación de contraseña.  
 * Permite al usuario solicitar instrucciones para restablecer su contraseña.  
 * @component  
 * @returns {JSX.Element}  
 */  
export default function RecuperarPassword() {  
    /**  
     * Hook de navegación de React Router.  
     */  
    const navigate = useNavigate()  
    /**  
     * Email introducido por el usuario.  
     * [(string|Function)[]]  
     */  
    const [email, setEmail] = useState("")  
    /**  
     * Paso actual del proceso (1: formulario, 2: mensaje de éxito).  
     * [[number, Function]]  
     */  
    const [step, setStep] = useState(1)  
    /**  
     * Mensaje de error a mostrar.  
     * [[string|null, Function]]  
     */
```

Ilustración 93 Reset Pass jsx I

```

const [errorMessage, setErrorMessage] = useState(null)
/**
 * Estado de carga mientras se realiza la petición.
 * {[boolean, Function]}
 */
const [isLoading, setIsLoading] = useState(false)

/**
 * Envía la solicitud de recuperación de contraseña a la API.
 * Si es exitosa, muestra mensaje de éxito. Si falla, muestra error.
 * @param {React.FormEvent} event
 */
async function enviarSolicitud(event) {
    event.preventDefault()
    setIsLoading(true)

    const url = API_URL + "api/enviar-restablecimiento"
    // No es necesario el token para recuperación
    axios
        .post(url, { email: email })
        .then((response) => {
            setStep(2)
            setErrorMessage(null)
            // setDisponibilidadHora(response.data.disponible) // No es necesario aquí
        })
        .catch((error) => {
            console.log(error)
            setErrorMessage("Error al enviar el correo. Por favor, intenta más tarde.")
        }).finally(() => {
            setIsLoading(false)
        })
}

```

Ilustración 94 Reset Pass jsx II

```

/**
 * Navega a la página de login.
 * @param {React.MouseEvent} e
 */
const handleNavigateToLogin = (e) => {
    e.preventDefault()
    navigate("/")
}

return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100 px-4">
        <div className="w-full max-w-md p-10 my-3 bg-white rounded-xl shadow-lg border border-gray-200">
            /* Logo de la aplicación */
            <div className="flex justify-center mb-6">
                
            </div>

            {/* Paso 1: Formulario para introducir el email */}
            {step === 1 ? (
                <>
                    <h2 className="text-center text-3xl font-semibold text-gray-800 mb-6">Recuperar Contraseña</h2>
                    <p className="text-center text-gray-600 mb-6">
                        Ingresá tu correo electrónico y te enviaremos un enlace para restablecer tu contraseña.
                    </p>
                    {/* Mensaje de error si existe */}
                    {errorMessage && (
                        <div className="text-sm text-center text-red-700 bg-red-100 rounded py-2 mb-4">{errorMessage}</div>
                    )}
                </>
            ) : null}
        </div>
    </div>
)

```

Ilustración 95 Reset Pass jsx III

```

    /* Formulario de recuperación */
    <form onSubmit={enviarSolicitud} className="space-y-5">
      <div>
        <label className="block text-sm font-medium text-gray-700 mb-1">Email</label>
        <input
          type="email"
          className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:border-blue-800 transition-[border-color, outline] duration-150"
          placeholder="ejemplo@correo.com"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
      </div>

      <button
        type="submit"
        className="w-full py-2 bg-blue-700 text-white font-semibold rounded-md hover:bg-blue-800 transition-[background-color] duration-150"
      >
        {isLoading ? (
          <div className="flex justify-center items-center">
            <span className="h-6 w-6 border-2 border-white border-t-transparent rounded-full animate-spin" style={{ border: `2px solid ${isSubmitting ? '#007bff' : 'white'}` }}></span>
          </div>
        ) : (
          <span className="flex items-center justify-center gap-2">
            <Send size={16} />
            Enviar Instrucciones
          </span>
        )}
      </button>
    </form>
  </>
) : (

```

Ilustración 96 Reset Pass IV

```

    // Paso 2: Mensaje de éxito tras enviar el correo
    <>
      <div className="text-center">
        <div className="flex justify-center mb-4">
          <div className="w-16 h-16 bg-green-100 rounded-full flex items-center justify-center">
            <svg
              xmlns="http://www.w3.org/2000/svg"
              className="h-8 w-8 text-green-600"
              fill="none"
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
              <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M5 13l4 4L19 7" />
            </svg>
          </div>
        </div>
        <h2 className="text-2xl font-semibold text-gray-800 mb-2">Correo Enviado</h2>
        <p className="text-gray-600 mb-6">
          Hemos enviado instrucciones para restablecer tu contraseña a <strong>{email}</strong>. Por favor, revisa tu bandeja de entrada y sigue las instrucciones.
        </p>
        <p className="text-sm text-gray-500 mb-6">
          Si no recibes el correo en unos minutos, revisa tu carpeta de spam o correo no deseado.
        </p>
        <button
          onClick={() => setStep(1)}
          className="text-blue-700 hover:text-blue-800 font-medium flex items-center justify-center mx-auto"
        >
          <ArrowLeft size={16} className="mr-1" />
          Volver a intentar
        </button>
      </div>
    </>
  )

```

Ilustración 97 Reset Pass V

```

    /* Enlace para volver al login */
    <p className="text-center text-sm text-gray-600 mt-6">
      ¿Ya recordaste tu contraseña? " "
      <button
        onClick={handleNavigateToLogin}
        className="underline cursor-pointer hover:text-blue-700 font-normal text-sm"
      >
        Iniciar Sesión
      </button>
    </p>
  </div>
)

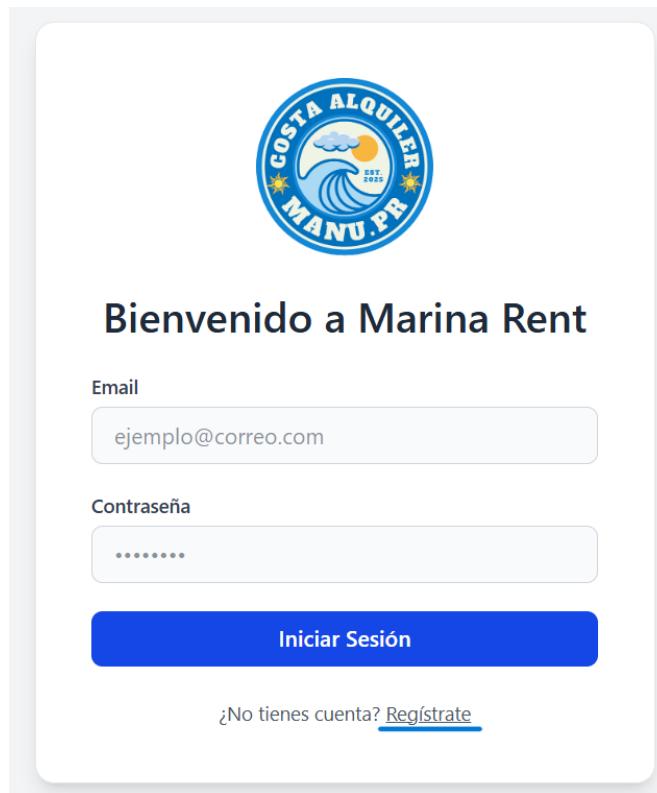
```

Ilustración 98 Reset Pass VI

Register.jsx

Esta clase es usada para registrarte y darte de alta como usuario en la página, para acceder a ella simplemente no tendremos que tener la sesión iniciada y acceder al loggin, después de esto daremos click en no tienes cuenta **regístrate**.

Al dar en regístrate si los datos son válidos nos redireccionará al home y guardará el token para poder hacer peticiones sobre toda la página.

*Ilustración 99 Register I*



Regístrate en Marina Rent

Nombre
prueba

Apellidos
prueba1

Email
prueba@gmail.com

Fecha de Nacimiento
05/06/2025

Ilustración 100 Register II

Contraseña
.....

Seguridad: Fuerte

- ✓ Mínimo 6 caracteres
- ✓ Al menos una mayúscula
- ✓ Al menos una minúscula
- ✓ Al menos un número
- ✓ Al menos un carácter especial

Confirmar Contraseña
.....

Regístrate

¿Ya tienes cuenta? [Iniciar Sesión](#)

Ilustración 101 Register III

Código – Register

```
/**
 * @file Register.jsx
 * @description Página de registro de usuario para Marina Rent.
 * Permite crear una cuenta nueva, valida la seguridad de la contraseña y muestra mensajes de error.
 */

import { useState } from "react"
import { useNavigate } from "react-router-dom"
import axios from "axios"
import { API_URL } from "../utilities/apirest"

/**
 * Componente de formulario de registro.
 * Gestiona el estado del usuario, validaciones, errores y navegación.
 * @component
 * @returns {JSX.Element}
 */
export default function Registro() {
    /**
     * Nombre del usuario.
     * {string|Function}[]
     */
    const [nombre, setNombre] = useState("")
    /**
     * Apellidos del usuario.
     * {string|Function}[]
     */
    const [apellidos, setApellidos] = useState("")
    /**
     * Email del usuario.
     * {string|Function}[]
     */
    const [email, setEmail] = useState("")
    /**
     * Fecha de nacimiento del usuario.
     * {string|Function}[]
     */
}
```

Ilustración 102 Register jsx I

```
/**
 * Fecha de nacimiento del usuario.
 * {string|Function}[]
 */
const [fecha_nacimiento, setFechaNacimiento] = useState("")
/**
 * Contraseña introducida por el usuario.
 * {string|Function}[]
 */
const [password, setPassword] = useState("")
/**
 * Confirmación de la contraseña.
 * {string|Function}[]
 */
const [password_confirmation, setPasswordConfirmation] = useState("")
/**
 * Mensaje de error a mostrar.
 * {[string|null, Function]}
 */
const [errorMessage, setErrorMessage] = useState(null)
/**
 * Estado de carga mientras se realiza la petición.
 * {[boolean, Function]}
 */
const [isLoading, setIsLoading] = useState(false)
/**
 * Hook de navegación de React Router.
 */
const navigate = useNavigate();
```

Ilustración 103 Register jsx II

```

/**
 * Maneja el envío del formulario de registro.
 * Realiza validaciones de contraseña y envía los datos a la API.
 * @param {React.FormEvent} event
 */
function realizarRegistro(event) {
  setIsLoading(true);
  event.preventDefault()

  // Validación de coincidencia de contraseñas
  if (password !== password_confirmation) {
    setIsLoading(false);
    setErrorMessage("Las contraseñas no coinciden. Inténtalo de nuevo.")
    return
  }

  // Validar longitud mínima
  if (password.length < 6) {
    setIsLoading(false);
    setErrorMessage("La contraseña debe tener al menos 6 caracteres.")
    return
  }

  // Validar seguridad de la contraseña
  const hasUpperCase = /[A-Z]/.test(password)
  const hasLowerCase = /[a-z]/.test(password)
  const hasNumbers = /\d/.test(password)
  const hasSpecialChar = /[^#\$%^&*()_-+=\{\};':"\\".,.<>/?]/.test(password)

  if (!(hasUpperCase && hasLowerCase && hasNumbers && hasSpecialChar)) {
    setIsLoading(false);
    setErrorMessage("La contraseña debe incluir mayúsculas, minúsculas, números y caracteres especiales.")
    return
  }
}

```

Ilustración 104 Register jsx III

```

// Enviar datos a la API de registro
const url = API_URL + "api/register"
axios
  .post(url, {
    Nombre: nombre,
    Apellidos: apellidos,
    Fecha_nacimiento: fecha_nacimiento,
    Email: email,
    Password: password,
    Password_confirmation: password_confirmation,
  })
  .then((response) => {
    if (response.status === 201 || response.status === 200) {
      // Si el registro es exitoso, guarda el token si la API lo devuelve
      if (response.data.token) {
        const token = response.data.token.split("|")[1]
        localStorage.setItem("authToken", token)
      }
      setErrorMessage(null)
      navigate("/home"); // Redirigir al home después del registro exitoso
    }
  })
  .catch((error) => {
    console.log(error)
    setIsLoading(false);
    if (error.response) {
      if (error.response.status === 422) {
        setErrorMessage("El email ya está registrado o los datos son inválidos.")
      } else {
        setErrorMessage("Error al registrar. Inténtalo de nuevo.")
      }
    } else {
      setErrorMessage("Error de conexión. Por favor, intenta más tarde.")
    }
  })
  .finally(() => {
    setIsLoading(false);
  });
}

```

Ilustración 105 Register jsx IV

```

/**
 * Calcula la fortaleza de la contraseña para mostrar feedback visual.
 * @param {string} password
 * @returns {{text: string, color: string}}
 */
function getPasswordStrength(password) {
  let strength = 0

  if (password.length >= 6) strength += 1
  if (password.length >= 8) strength += 1
  if (/^[A-Z]/.test(password)) strength += 1
  if (/^[a-z]/.test(password)) strength += 1
  if (/^\d/.test(password)) strength += 1
  if (/^[@#%&*()_-=\[\]{};':\"\\|,<>/?]/.test(password)) strength += 1

  if (strength <= 2) return { text: "Débil", color: "bg-red-500" }
  if (strength <= 4) return { text: "Media", color: "bg-yellow-500" }
  return { text: "Fuerte", color: "bg-green-500" }
}

```

Ilustración 106 Register.jsx V

```

return (
  <div className="min-h-screen flex items-center justify-center bg-gray-100 px-4">
    <div className="w-full max-w-md p-10 my-3 bg-white rounded-xl shadow-lg border border-gray-200">
      {/* Logo de la aplicación */}
      <div className="flex justify-center mb-6">
        
      </div>

      <h2 className="text-center text-3xl font-semibold text-gray-800 mb-6">Regístrate en Marina Rent</h2>

      {/* Mensaje de error si existe */}
      {errorMessage && (
        <div className="text-sm text-center text-red-700 bg-red-100 rounded py-2 mb-4">{errorMessage}</div>
      )}

      {/* Formulario de registro */}
      <form onSubmit={realizarRegistro} className="space-y-5">
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-1">Nombre</label>
          <input
            type="text"
            className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 text-gray-700 placeholder-Tu nombre"
            value={nombre}
            onChange={(e) => setNombre(e.target.value)}
            required
          />
        </div>

        <div>
          <label className="block text-sm font-medium text-gray-700 mb-1">Apellidos</label>
          <input
            type="text"
            className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 text-gray-700 placeholder-Tus apellidos"
            value={apellidos}
            onChange={(e) => setApellidos(e.target.value)}
            required
          />
        </div>
      </form>
    </div>
  </div>
)

```

Ilustración 107 Register.jsx VI

```

<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">Email</label>
  <input
    type="email"
    className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 text-gray-900 placeholder=ejemplo@correo.com"
    value={email}
    onChange={(e) => setEmail(e.target.value)}
    required
  />
</div>

<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">Fecha de Nacimiento</label>
  <input
    type="date"
    className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 text-gray-900 placeholder=fecha_nacimiento"
    value={fecha_nacimiento}
    onChange={(e) => setFechaNacimiento(e.target.value)}
    required
  />
</div>

<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">Contraseña</label>
  <input
    type="password"
    className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50 text-gray-900 placeholder=*****"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
    required
  />
</div>

```

Ilustración 108 Register jsx VII

```

/* Indicador visual de fortaleza de contraseña */
{password && (
  <div className="mt-2">
    <div className="flex items-center space-x-2">
      <div className="text-xs">Seguridad:</div>
      <div className={`${h-2 w-full rounded ${getPasswordStrength(password).color}`}>${getPasswordStrength(password).text}</div>
    </div>
    <ul className="text-xs text-gray-600 mt-1 space-y-1">
      <li className={password.length >= 6 ? "text-green-600" : "text-gray-600"}>✓ Mínimo 6 caracteres</li>
      <li className={/[A-Z]/.test(password) ? "text-green-600" : "text-gray-600"}>
        ✓ Al menos una mayúscula
      </li>
      <li className={/[a-z]/.test(password) ? "text-green-600" : "text-gray-600"}>
        ✓ Al menos una minúscula
      </li>
      <li className={/\d/.test(password) ? "text-green-600" : "text-gray-600"}>✓ Al menos un número</li>
      <li
        className={
          /[!@#$%^&*()_+\-=\[\]{};':"\\\|,.<>/?]/.test(password) ? "text-green-600" : "text-gray-600"
        }
      >
        ✓ Al menos un carácter especial
      </li>
    </ul>
  </div>
)
}

```

Ilustración 109 Register jsx VIII

```

<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">Confirmar Contraseña</label>
  <input
    type="password"
    className="w-full px-4 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500 bg-gray-50"
    placeholder="*****"
    value={password_confirmation}
    onChange={(e) => setPasswordConfirmation(e.target.value)}
    required
  />
</div>

<button
  type="submit"
  className="w-full py-2 bg-blue-700 text-white font-semibold rounded-md hover:bg-blue-800 transition-colors"
>
  {isLoading ? (
    <>
      <div className="flex justify-center items-center">
        | <span className="h-6 w-6 border-2 border-white border-t-transparent rounded-full animate-spin"></span>
      </div>
    </>
  ) : (
    <span>Registrate</span>
  )}
</button>
</form>

```

Ilustración 110 Register IX

```

/* Enlace para usuarios ya registrados */
<p className="text-center text-sm text-gray-600 mt-6">
  ¿Ya tienes cuenta?{" "}
  <span className="underline cursor-pointer hover:text-blue-700">
    <span className="underline cursor-pointer hover:text-blue-700">
      <span
        onClick={() => navigate("")}
        className="underline cursor-pointer hover:text-blue-700"
      >
        Iniciar Sesión
      </span>
    </span>
  </span>
</p>
</div>
</div>
)
}

```

Ilustración 111 Register X

Home.jsx

Esta es la sección principal de la página web, ella contiene información en html sobre la página web, esta página es la primera que vemos cuando conseguimos iniciar sesión.

Esta página muestra los componentes de Navbar.jsx y Footer.jsx los cuales van a ser explicados en los siguientes apartados.

Además esta página incluye el componente de MarbellaMap.jsx el cual también va a ser explicado en los siguientes puntos.



Ilustración 112 Home I



Ilustración 113 Home II



Ilustración 114 Home III



Ilustración 115 Home IV

Código – Home

```

import { useNavigate } from "react-router-dom"
import { CheckCircleIcon } from "@heroicons/react/24/solid"
import MarbellaMap from "../components/MarbellaMap"
import { useEffect, useState } from "react"
import { API_URL, IMAGE_URL } from "../utilities/apirest"
import axios from "axios"

/**
 * Página principal (Home) de la aplicación.
 * Muestra un video de portada, razones para elegir MarinaRent, publicaciones destacadas y un mapa.
 * @component
 * @returns {JSX.Element}
 */
export default function Home() {
  /**
   * Hook de navegación de React Router.
   */
  const navigate = useNavigate()
  /**
   * Lista de publicaciones destacadas obtenidas de la API.
   * {[Array, Function]}
   */
  const [publicaciones, setPublicaciones] = useState([])
  /**
   * Estado de carga para mostrar loader mientras se obtienen las publicaciones.
   * {[boolean, Function]}
   */
  const [loading, setLoading] = useState(true)

```

Ilustración 116 Home.jsx I

```

/**
 * Efecto para obtener publicaciones aleatorias al montar el componente.
 */
useEffect(() => {
  const obtenerPublicaciones = async () => {
    try {
      const url = API_URL + "api/publicacionesAleatorias"
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}
      setLoading(true)
      const response = await axios.get(url, { headers })
      setPublicaciones(response.data)
    } catch (error) {
      console.error("Error al obtener publicaciones:", error)
    } finally {
      setLoading(false)
    }
  }
  obtenerPublicaciones()
}, [])

/**
 * Obtiene la primera imagen de la cadena de imágenes separadas por punto y coma.
 * @param {string} imagenes - Cadena de imágenes separadas por ';'
 * @returns {string} - URL de la primera imagen
 */
const obtenerPrimeraImagen = (imagenes) => {
  if (!imagenes) return ""
  const primeraImagen = imagenes.split(";")[0]
  return `${IMAGE_URL}${primeraImagen}`
}

```

Ilustración 117 Home.jsx II

```

return (
  <>
  <main>
    /* Sección de video de portada con texto superpuesto */
    <div className="relative w-full aspect-video">
      /* Texto encima del video */
      <div className="text-4xl sm:text-5xl md:text-6xl lg:text-7xl xl:text-8xl font-bold text-white text-center">
        Bienvenidos a Marina Rent
      </h1>
      <h2 className="text-base sm:text-lg md:text-xl font-semibold text-white mt-2 md:mt-4 text-center">
        Diversión desde el Principio
      </h2>
    </div>
    /* iframe de YouTube como fondo de video */
    <iframe
      id="player"
      className="absolute inset-0 w-full h-full border-0"
      allowFullScreen
      allow="autoplay encrypted-media"
      referrerPolicy="strict-origin-when-cross-origin"
      title="ScubaCaribe"
      src="https://www.youtube.com/embed/tX785Tzm4UE?autoplay=1&mute=1&loop=1&controls=0&modestbranding=1&rel=0&disablekb=1&iv_load_policy=3"
    />
  </div>

  /* Banner de TripAdvisor */
  <div className="w-full justify-center flex bg-green-400 py-2 md:py-0">
    
  </div>
)

```

Ilustración 118 Home jsx III

```

/* Sección de 10 razones para elegir MarinaRent */
<section className="flex flex-col md:flex-row">
  <div className="w-full md:w-1/2 flex flex-col items-start py-8 md:py-16 px-4 md:px-5">
    <h2 className="text-3xl sm:text-4xl md:text-5xl">
      <span className="font-bold">10 RAZONES </span>POR QUÉ MARINARENT
    </h2>
    <p className="py-3 md:py-5 text-base md:text-xl">
      Desde 1991, Marinarent ha brindado una gama completa de buceo, snorkel, tours y deportes acuáticos para
      los mejores destinos de playa del mundo.
    </p>
    /* Lista de razones con icono */
    <ul className="flex flex-row flex-wrap font-bold text-sm sm:text-base md:text-lg text-gray-700">
      [
        "Más de 30 años de experiencia",
        "Embajadores ambientales",
        "Expansión global",
        "Familia y grupos",
        "Altos estándares de seguridad",
        "Amplia cartera de experiencias",
        "Calidad y atención al cliente",
        "Equipo de primera categoría",
        "Flota totalmente equipada",
        "Instructores plurilingües",
      ].map((text, index) => (
        <li key={index} className="w-full sm:w-1/2 py-2 md:py-3 flex items-center gap-2">
          <checkCircleIcon className="h-5 w-5 md:h-6 md:w-6 text-blue-600 shrink-0" />
          <text>
            {text}
          </text>
        </li>
      ))
    </ul>
  </div>
  /* Imagen de barco a la derecha */
  <div className="w-full md:w-1/2">
    
  </div>
</section>

```

Ilustración 119 Home jsx IV

```

/* Sección de publicaciones destacadas */
<section className="py-8 md:py-16 bg-gray-50">
  <div className="mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
    <h2 className="text-3xl sm:text-4xl md:text-5xl font-semibold text-center text-blue-900">
      Publicaciones Destacadas
    </h2>
    {/* Loader mientras se cargan las publicaciones */}
    {loading ? (
      <div className="flex justify-center items-center py-20">
        <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-blue-700"></div>
      </div>
    ) : (
      // Grid de publicaciones destacadas
      <div className="mt-6 md:mt-10 grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4 md:gap-8">
        {publicaciones.map((publicacion) => (
          <div key={publicacion.id}
            className="flex flex-col items-center rounded-lg bg-white p-4 md:p-6 shadow-lg hover:shadow-xl transition-shadow duration-300"
            onClick={() => (navigate("/mostrador", { state: { elemento: publicacion } }))}
            style={{ cursor: "pointer" }}>
            <div className="relative w-full">
              <img
                src={obtenerPrimerImagen(publicacion.imagen) || "/placeholder.svg"}
                alt={publicacion.titulo}
                className="w-full h-48 sm:h-50 object-cover rounded mb-4 md:mb-6 hover:blur-sm scale-[1.01] transition duration-300"
                onError={(e) => [
                  // Imagen de respaldo si falla la carga
                  e.target.src = "placeholder.jpg"
                ]}
              />
            </div>
            <h3 className="text-lg md:text-xl font-semibold text-blue-700 line-clamp-1">
              {publicacion.titulo}
            </h3>
            <p className="text-center text-sm md:text-base text-gray-600 mt-2 line-clamp-3">
              {publicacion.descripcion}
            </p>
          </div>
        ))
      </div>
    )}
  </div>
</section>

```

Ilustración 120 Home jsx V

```

    </div>
  )
)
</div>
</section>

/* Mapa de Marbella */
<MarbellaMap />
</main>
</>
)
}
}

```

Ilustración 121 Home jsx VI

Informativos.jsx

La página de Informativos se encarga de mostrar aquellas publicaciones que son de tipo informativo, además tiene un buscador.jsx el cual explicaremos en los siguientes apartados.

Esta página realiza peticiones al servidor obteniendo directamente las publicaciones de tipo informativo además retornando estas de forma paginada.

El buscador actúa correctamente, además si estamos en la página 2 y hacemos una petición de búsqueda, el buscador buscará en todas las páginas retornadas por el servidor.



Publicaciones de Tipo Informativo

Encuentra información actualizada sobre eventos y noticias gratuitos

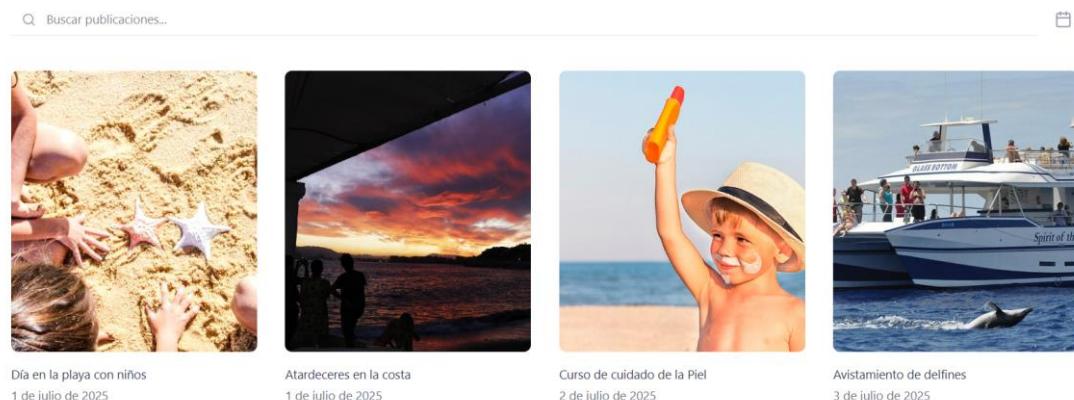


Ilustración 122 Informativo I



Ilustración 123 Informativo II

Publicaciones de Tipo Informativo

Encuentra información actualizada sobre eventos y noticias gratuitos

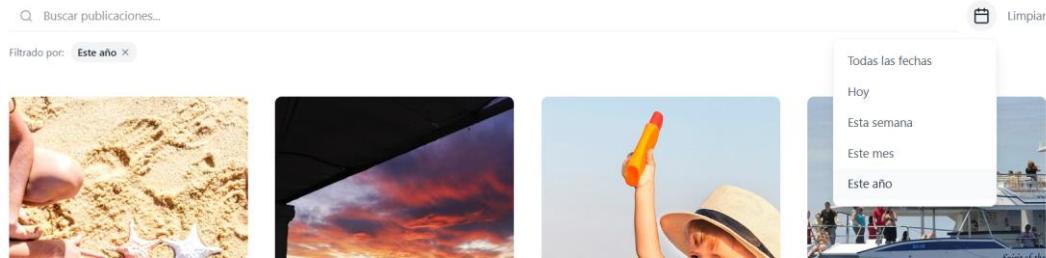


Ilustración 124 Informativo III

Publicaciones de Tipo Informativo

Encuentra información actualizada sobre eventos y noticias gratuitos

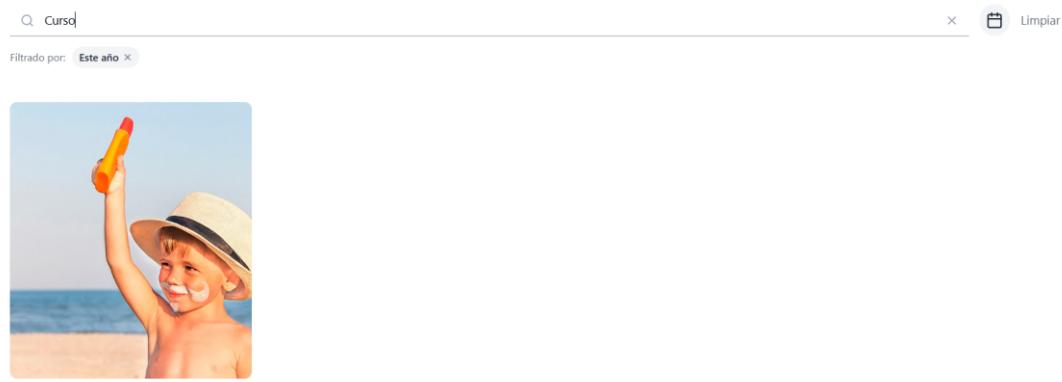


Ilustración 125 Informativo IV

Código – Informativos

```
/*
 * #file Informativos.jsx
 * @description Página para mostrar publicaciones de tipo informativo con búsqueda, filtrado y paginación.
 */

import { useState, useEffect } from "react"
import { API_URL, IMAGE_URL } from "../../utilities/apiRest"
import axios from "axios"
import { useNavigate } from "react-router-dom"
import Buscador from "./components/buscador"
import { ChevronLeft, ChevronRight } from "lucide-react"

/** 
 * Página de publicaciones informativas.
 * Permite buscar, filtrar por fecha, pagination y navegar a la vista de detalle.
 * @component
 * @returns {JSX.Element}
 */
export default function Informativos() {
    /**
     * Lista de elementos informativos cargados desde la API.
     * [{Array, Function}]
     */
    const [elementos, setElementos] = useState([])
    /**
     * Estado de carga de la página.
     * [{boolean, Function}]
     */
    const [loading, setLoading] = useState(true)
    /**
     * Término de búsqueda.
     * [(string|Function)[]]
     */
    const [searchTerm, setSearchTerm] = useState("")
    /**
     * Filtro de fecha seleccionado.
     * [(string|Function)[]]
     */
}
```

Ilustración 126 Informativos jsx I

```

    /**
     * Filtro de fecha seleccionado.
     *  [(string|Function)[]]
     */
    const [filterDate, setFilterDate] = useState("")
    /**
     * Página actual de la paginación.
     *  [[number, Function]]
     */
    const [currentPage, setCurrentPage] = useState(1)
    /**
     * Total de páginas disponibles.
     *  [[number, Function]]
     */
    const [totalPages, setTotalPages] = useState(1)
    /**
     * Hook de navegación de React Router.
     */
    const navigate = useNavigate()

    /**
     * Efecto para cargar elementos al cambiar de página.
     */
    useEffect(() => {
        cargarElementos(currentPage)
    }, [currentPage])

```

Ilustración 127 Informativos.jsx II

```

    /**
     * Carga los elementos informativos desde la API.
     * @param {number} pagina - Página a cargar.
     */
    function cargarElementos(pagina) {
        setLoading(true)
        const url = API_URL + "api/informativosPaginados"
        const token = localStorage.getItem("authToken")
        const headers = token ? { Authorization: `Bearer ${token}` } : {}

        axios
            .post(url, { pagina }, { headers })
            .then((response) => {
                if (response.data.success) {
                    setElementos(response.data.data)
                    setTotalPages(response.data.totalPages)
                }
            })
            .catch((error) => {
                console.error("Error al cargar los informativos:", error)
            })
            .finally(() => {
                setLoading(false)
            })
    }

    /**
     * Navega a la página de mostrador con el elemento seleccionado.
     * @param {Object} elemento
     */
    const handleClick = (elemento) => {
        navigate("/mostrador", { state: { elemento } })
    }

```

Ilustración 128 Informativos.jsx III

```

    /**
     * Formatea la fecha a formato legible en español.
     * @param {string} dateString
     * @returns {string}
     */
    function formatDate(dateString) {
        const date = new Date(dateString)
        const options = { year: "numeric", month: "long", day: "numeric" }
        return date.toLocaleDateString("es-ES", options)
    }

    /**
     * Maneja la búsqueda por término.
     * @param {string} term
     */
    const handleSearch = (term) => {
        setSearchTerm(term)
        setCurrentPage(1) // Reinicia a la primera página al buscar
    }

    /**
     * Maneja el filtrado por fecha.
     * @param {string} datefilter
     */
    const handleFilter = (dateFilter) => {
        setFilterDate(dateFilter)
        setCurrentPage(1) // Reinicia a la primera página al filtrar
    }

    /**
     * Resetea los filtros y búsqueda.
     */
    const handleReset = () => {
        setSearchTerm("")
        setFilterDate("")
        setCurrentPage(1) // Reinicia a la primera página al limpiar filtros
        cargarElementos(1)
    }
}

```

Ilustración 129 Informativos jsx IV

```

    /**
     * Cambia la página actual.
     * @param {number} newPassword
     */
    const handlePageChange = (newPage) => {
        if (newPage >= 1 && newPassword <= totalPages) {
            setCurrentPage(newPage)
        }
    }

    /**
     * Filtra los elementos por fecha según el filtro seleccionado.
     * @param {string} fechaStr
     * @returns {boolean}
     */
    const filtrarPorFecha = (fechaStr) => {
        if (!filterDate) return true

        const fecha = new Date(fechaStr)
        const hoy = new Date()

        switch (filterDate) {
            case "today":
                return fecha.toDateString() === hoy.toDateString()
            case "week":
                const primerDiaSemana = new Date(hoy)
                primerDiaSemana.setDate(hoy.getDate() - hoy.getDay())
                const ultimoDiaSemana = new Date(primerDiaSemana)
                ultimoDiaSemana.setDate(primerDiaSemana.getDate() + 6)
                return fecha >= primerDiaSemana && fecha <= ultimoDiaSemana
            }
            case "month":
                return fecha.getMonth() === hoy.getMonth() && fecha.getFullYear() === hoy.getFullYear()
            case "year":
                return fecha.getFullYear() === hoy.getFullYear()
            default:
                return true
        }
    }
}

```

Ilustración 130 Informativos jsx V

```

    /**
     * Elementos filtrados por fecha y búsqueda.
     * Solo muestra eventos futuros o actuales.
     */
    const elementosFiltrados = elementos
      // Primero: filtrar publicaciones futuras o actuales
      .filter((elemento) => {
        const fechaEvento = new Date(elemento.fecha_evento)
        const hoy = new Date()
        hoy.setHours(0, 0, 0, 0) // para ignorar la hora
        return fechaEvento >= hoy
      })
      // Segundo: aplicar filtro de búsqueda y fecha
      .filter((elemento) => {
        const coincideBusqueda = elemento.titulo.toLowerCase().includes(searchTerm.toLowerCase())
        const coincideFecha = filtrarPorFecha(elemento.fecha_evento)
        return coincideBusqueda && coincideFecha
      })

    /**
     * Efecto para intentar cargar más datos si el filtrado no arroja resultados y hay más páginas.
     */
    useEffect(() => {
      if (searchTerm || filterDate) {
        if (elementosFiltrados.length === 0 && currentPage < totalPages) {
          setCurrentPage(currentPage + 1)
        }
      }
    }, [elementosFiltrados.length])
  
```

Ilustración 131 Informativos jsx VI

```

    /**
     * Renderiza los botones de paginación.
     * @returns {JSX.Element}
     */
    const renderPaginationButtons = () => {
      return (
        <div className="flex justify-center items-center mt-8 gap-2">
          <button
            onClick={() => handlePageChange(currentPage - 1)}
            disabled={currentPage === 1}
            className="relative inline-flex items-center px-2 py-2 text-sm font-medium text-gray-500 bg-white border border-gray-300 rounded-md h-8 w-4" />
          <chevronLeft className="h-4 w-4" />
          <span className="text-sm">
            Página {currentPage} de {totalPages}
          </span>
          <button
            onClick={() => handlePageChange(currentPage + 1)}
            disabled={currentPage === totalPages}
            className="relative inline-flex items-center px-2 py-2 text-sm font-medium text-gray-500 bg-white border border-gray-300 rounded-md h-8 w-4" />
          <chevronRight className="h-4 w-4" />
        </div>
      )
    }

    return (
      <div className="bg-white">
        <div className="mx-auto max-w-2xl px-4 py-16 sm:px-6 sm:py-12 lg:max-w-7xl lg:px-8">
          <div className="mb-8">
            <h1 className="text-3xl font-bold text-gray-900 mb-1">Publicaciones de Tipo Informativo</h1>
            <p className="text-gray-500 text-sm">Encuentra información actualizada sobre eventos y noticias gratuitos</p>
          </div>
        </div>
      </div>
    )
  
```

Ilustración 132 Informativos jsx VII

```

    /* Buscador de publicaciones y filtros */
    <div className="mb-10">
      <Buscador onSearch={handleSearch} onFilter={handleFilter} onReset={handleReset} />
    </div>

    /* Loader de carga */
    <div className="flex justify-center items-center h-64">
      <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
    </div>
  ) : (
    <div>
      /* Grid de elementos filtrados */
      <div className="mt-6 grid grid-cols-1 gap-x-6 gap-y-10 sm:grid-cols-2 lg:grid-cols-4 xl:gap-x-8">
        {elementosFiltrados.length > 0 ? (
          elementosFiltrados.map((elemento) => (
            <div key={elemento.id}>
              <div className="group relative cursor-pointer" onClick={() => handleClick(elemento)}>
                /* Imagen principal del elemento */
                <img src={IMAGE_URL + elemento.imagen.split(";")[0] || "/placeholder.svg"} alt={elemento.titulo} className="aspect-square w-full rounded-md bg-gray-200 object-cover group-hover:opacity-75 lg:aspect-auto" />
                <div className="mt-4 flex justify-between">
                  <div>
                    /* Título y fecha del evento */
                    <h3 className="text-sm text-gray-700">{elemento.titulo}</h3>
                    <p className="mt-1 text-sm text-gray-500">{formatDate(elemento.fecha_evento)}</p>
                  </div>
                </div>
              </div>
            ) : (
              <div>
                // Mensaje cuando no hay resultados
                <div className="col-span-full text-center py-10">
                  <p className="text-gray-500">
                    No se encontraron resultados que coincidan con los criterios de búsqueda.
                  </p>
                </div>
              ) )
            </div>
          ) : (
            /* Controles de paginación */
            {loading && totalPages > 1 && <div className="mt-10">{renderPaginationButtons()}</div>}
          )
        ) )
      </div>
    </div>
  )
)

```

Ilustración 133 Informativos jsx VIII

```

    ) : (
      // Mensaje cuando no hay resultados
      <div className="col-span-full text-center py-10">
        <p className="text-gray-500">
          No se encontraron resultados que coincidan con los criterios de búsqueda.
        </p>
      </div>
    )
  )
)
</div>

```

Ilustración 134 Informativos jsx IX

Alquilables.jsx

La página de Alquilables se encarga de mostrar aquellas publicaciones que son de tipo alquilable, además tiene un buscador.jsx el cual explicaremos en los siguientes apartados.

Esta página realiza peticiones al servidor obteniendo directamente las publicaciones de tipo alquilable además retornando estas de forma paginada.

El buscador actúa correctamente, además si estamos en la página 2 y hacemos una petición de búsqueda, el buscador buscará en todas las páginas retornadas por el servidor.

 Inicio Informativos Reservables Mis Reservas Carrito IA Chatbox Admin 

Publicaciones de Tipo Reservable

Encuentra información actualizada sobre eventos con reserva previa

Buscar publicaciones... 

				
Tour en barco por la bahía 1 de julio de 2025	Moto de agua 30 min 2 de julio de 2025	45 €	Paseo en velero privado 3 de julio de 2025	150 €
			Pesca deportiva con patrón 4 de julio de 2025	90 €

Ilustración 135 Alquilables I

				
Clase de paddle surf 4 de julio de 2025	Kayak por acantilados 5 de julio de 2025	30 €	Snorkel guiado 5 de julio de 2025	35 €
			Bautismo de buceo 6 de julio de 2025	70 €

< Página 1 de 2 >

Enlaces rápidos	Contacto	Síguenos
Inicio	manuels365@gmail.com	
Informativos	Teléfono: +34 641 130 893	
Reservables		

Ilustración 136 Alquilables II

Publicaciones de Tipo Reservable

Encuentra información actualizada sobre eventos con reserva previa

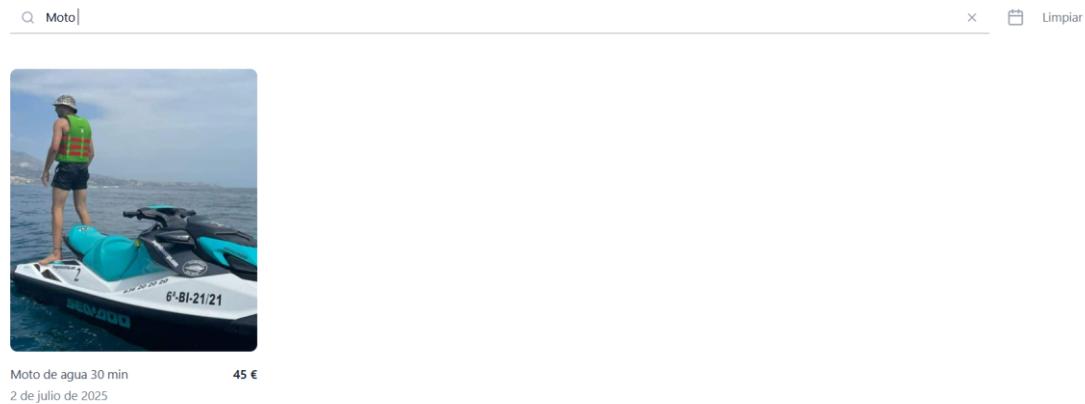


Ilustración 137 Alquilables III

Código – Alquilables

```
/**
 * @file Alquilables.jsx
 * @description Página para mostrar publicaciones de tipo reservable (alquilables) con búsqueda, filtrado y paginación.
 */

import { useState, useEffect } from "react"
import { API_URL, IMAGE_URL } from "../utilities/apirest"
import axios from "axios"
import { useNavigate } from "react-router-dom"
import Buscador from "../components/buscador"
import { ChevronLeft, ChevronRight } from "lucide-react"

/**
 * Página de publicaciones reservables.
 * Permite buscar, filtrar por fecha, paginar y navegar a la vista de detalle.
 * @component
 * @returns {JSX.Element}
 */
export default function Informativos() {
    /**
     * Lista de elementos reservables cargados desde la API.
     * {[Array, Function]}
     */
    const [elementos, setElementos] = useState([])
    /**
     * Estado de carga de la página.
     * {[boolean, Function]}
     */
    const [loading, setLoading] = useState(true)
    /**
     * Término de búsqueda.
     * {(string|Function)[]}
     */
    const [searchTerm, setSearchTerm] = useState("")
    /**
     * Filtro de fecha seleccionado.
     * {(string|Function)[]}
     */
    const [filterDate, setFilterDate] = useState("")
```

Ilustración 138 Alquilables jsx I

```

    /**
     * Página actual de la paginación.
     * {[number, Function]}
     */
    const [currentPage, setCurrentPage] = useState(1)
    /**
     * Total de páginas disponibles.
     * {[number, Function]}
     */
    const [totalPages, setTotalPages] = useState(1)
    /**
     * Hook de navegación de React Router.
     */
    const navigate = useNavigate()

    /**
     * Efecto para cargar elementos al cambiar de página.
     */
    useEffect(() => {
        cargarElementos(currentPage)
    }, [currentPage])
}

```

Ilustración 139 Alquilables.jsx II

```

    /**
     * Carga los elementos reservables desde la API.
     * @param {number} pagina - Página a cargar.
     */
    function cargarElementos(pagina) {
        setLoading(true)
        const url = API_URL + "api/alquilablesPaginados"
        const token = localStorage.getItem("authToken")
        const headers = token ? { Authorization: `Bearer ${token}` } : {}
        axios
            .post(url, { pagina }, { headers })
            .then((response) => {
                if (response.data.success) {
                    setElementos(response.data.data)
                    setTotalPages(response.data.totalPages)
                }
            })
            .catch((error) => {
                console.error("Error al cargar los alquilables:", error)
            })
            .finally(() => {
                setLoading(false)
            })
    }

    /**
     * Maneja la búsqueda por término.
     * @param {string} term
     */
    const handleSearch = (term) => {
        setSearchTerm(term)
        setCurrentPage(1) // Reinicia a la primera página al buscar
    }
}

```

Ilustración 140 Alquilables.jsx III

```

    /**
     * Maneja el filtrado por fecha.
     * @param {string} dateFilter
     */
    const handleFilter = (dateFilter) => {
      setFilterDate(dateFilter)
      setCurrentPage(1) // Reinicia a la primera página al filtrar
    }

    /**
     * Resetea los filtros y búsqueda.
     */
    const handleReset = () => {
      setSearchTerm("")
      setFilterDate("")
      setCurrentPage(1) // Reinicia a la primera página al limpiar filtros
      cargarElementos(1)
    }

    /**
     * Cambia la página actual.
     * @param {number} newPassword
     */
    const handlePageChange = (newPage) => {
      if (newPage >= 1 && newPassword <= totalPages) {
        setCurrentPage(newPage)
      }
    }
  
```

Ilustración 141 Alquilables.jsx IV

```

    /**
     * Filtra los elementos por fecha según el filtro seleccionado.
     * @param {string} fechastr
     * @returns {boolean}
     */
    const filtrarPorFecha = (fechastr) => {
      if (!filterDate) return true

      const fecha = new Date(fechastr)
      const hoy = new Date()

      switch (filterDate) {
        case "today":
          return fecha.toDateString() === hoy.toDateString()
        case "week":
          const primerDiaSemana = new Date(hoy)
          primerDiaSemana.setDate(hoy.getDate() - hoy.getDay())
          const ultimoDiaSemana = new Date(primerDiaSemana)
          ultimoDiaSemana.setDate(primerDiaSemana.getDate() + 6)
          return fecha >= primerDiaSemana && fecha <= ultimoDiaSemana
        }
        case "month":
          return fecha.getMonth() === hoy.getMonth() && fecha.getFullYear() === hoy.getFullYear()
        case "year":
          return fecha.getFullYear() === hoy.getFullYear()
        default:
          return true
      }
    }
  
```

Ilustración 142 Alquilables.jsx V

```

    /**
     * Elementos filtrados por fecha y búsqueda.
     * Solo muestra eventos futuros o actuales.
     */
    const elementosFiltrados = elementos
      // Filtra publicaciones futuras o actuales
      .filter((elemento) => {
        const fechaEvento = new Date(elemento.fecha_evento)
        const hoy = new Date()
        hoy.setHours(0, 0, 0, 0) // Ignora la hora
        return fechaEvento >= hoy
      })
      // Aplica filtro de búsqueda y fecha
      .filter((elemento) => {
        const coincideBusqueda = elemento.titulo.toLowerCase().includes(searchTerm.toLowerCase())
        const coincideFecha = filtrarPorFecha(elemento.fecha_evento)
        return coincideBusqueda && coincideFecha
      })

    /**
     * Efecto para intentar cargar más datos si el filtrado no arroja resultados y hay más páginas.
     */
    useEffect(() => {
      if (searchTerm || filterDate) {
        if (elementosFiltrados.length === 0 && currentPage < totalPages) {
          setCurrentPage(currentPage + 1)
        }
      }
    }, [elementosFiltrados.length])

    /**
     * Navega a la página de mostrador con el elemento seleccionado.
     * @param {Object} elemento
     */
    const handleClick = (elemento) => {
      navigate("/mostrador", { state: { elemento } })
    }
  
```

Ilustración 143 Alquilables.jsx VI

```

    /**
     * Formatea la fecha a formato legible en español.
     * @param {String} dateString
     * @returns {string}
     */
    function formatDate(dateString) {
      const date = new Date(dateString)
      const options = { year: "numeric", month: "long", day: "numeric" }
      return date.toLocaleDateString("es-ES", options)
    }

    /**
     * Renderiza los botones de paginación.
     * @returns {JSX.Element}
     */
    const renderPaginationButtons = () => {
      return (
        <div className="flex justify-center items-center mt-8 gap-2">
          <button
            onClick={() => handlePageChange(currentPage - 1)}
            disabled={currentPage === 1}
            className="relative inline-flex items-center px-2 py-2 text-sm font-medium text-gray-500 bg-white border border-gray-300 rounded-md">
            <ChevronLeft className="h-4 w-4" />
          </button>
          <span className="text-sm">
            Página {currentPage} de {totalPages}
          </span>
          <button
            onClick={() => handlePageChange(currentPage + 1)}
            disabled={currentPage === totalPages}
            className="relative inline-flex items-center px-2 py-2 text-sm font-medium text-gray-500 bg-white border border-gray-300 rounded-md">
            <ChevronRight className="h-4 w-4" />
          </button>
        </div>
      )
    }
  
```

Ilustración 144 Alquilables.jsx V

```
return (
  <div className="bg-white">
    <div className="mx-auto max-w-2xl px-4 py-16 sm:px-6 sm:py-12 lg:max-w-7xl lg:px-8">
      <div className="mb-8">
        <h1>Publicaciones de Tipo Reservable</h1>
        <p>Encuentra información actualizada sobre eventos con reserva previa</p>
      </div>

      {/* Buscador de publicaciones y filtros */}
      <div className="mb-10">
        <Buscador onSearch={handleSearch} onFilter={handleFilter} onReset={handleReset} />
      </div>

      {/* Loader de carga */}
      {loading ? (
        <div className="flex justify-center items-center h-64">
          <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
        </div>
      ) : (
        <>
          {/* Grid de elementos filtrados */}
          <div className="mt-6 grid grid-cols-1 gap-x-6 gap-y-10 sm:grid-cols-2 lg:grid-cols-4 xl:gap-x-8">
            {elementosfiltrados.length > 0 ? (
              elementosfiltrados.map((elemento) => (
                <div
                  key={elemento.id}
                  className="group relative cursor-pointer"
                  onClick={() => handleClick(elemento)}
                >
                  {/* Imagen principal del elemento */}
                  <img
                    src={IMAGE_URL + elemento.imagen.split(";")[0] || "/placeholder.svg"}
                    alt={elemento.titulo}
                    className="aspect-square w-full rounded-md bg-gray-200 object-cover group-hover:opacity-75 lg:aspect-auto lg:h-80"
                  />
              
```

)));
 </div>
 </>
)}
 </div>
 </div>
)

Ilustración 145 Alquilables jsx VI

```
        <div className="mt-4 flex justify-between">
          <div>
            {/* Título y fecha del evento */}
            <h3 className="text-sm font-medium text-gray-700">{elemento.titulo}</h3>
            <p className="mt-1 text-sm text-gray-500">{formatDate(elemento.fecha_evento)}</p>
          </div>
          {/* Precio del evento */}
          <p className="text-sm font-medium text-gray-900">{elemento.precio} €</p>
        </div>
      </div>
    )
  ) : (
  // Mensaje cuando no hay resultados
  <div className="col-span-full text-center py-10">
    <p className="text-gray-500">
      No se encontraron resultados que coincidan con los criterios de búsqueda.
    </p>
  </div>
)
}
</div>

 {/* Controles de paginación */}
 {!loading && totalPages > 1 && <div className="mt-10">{renderPaginationButtons()}</div>
</div>
</div>
)
}
</div>
</div>
}
```

Ilustración 146 Alquilables jsx VII

Mis Reservas.jsx

La clase de Mis reservas a parte de ser la más compleja es la más completa, esta página muestra un calendario Calendar.jsx el cual interactua con el servidor para poder mostrar las reservas realizadas por el usuario en el que tenemos la sesión iniciada en todo momento.

Al hacer click en un dia de dicho mes se nos desplegará un modal en el que podremos cancelar la reserva de dicho día.

Este componente ha sido sacado desde la página de tailwindcss, los modales han sido anidados a dicho componente y dicho componente será explicado en los próximos apartados.

Ilustración 147 Mis Reservas I

Ilustración 148 Mis Reservas II

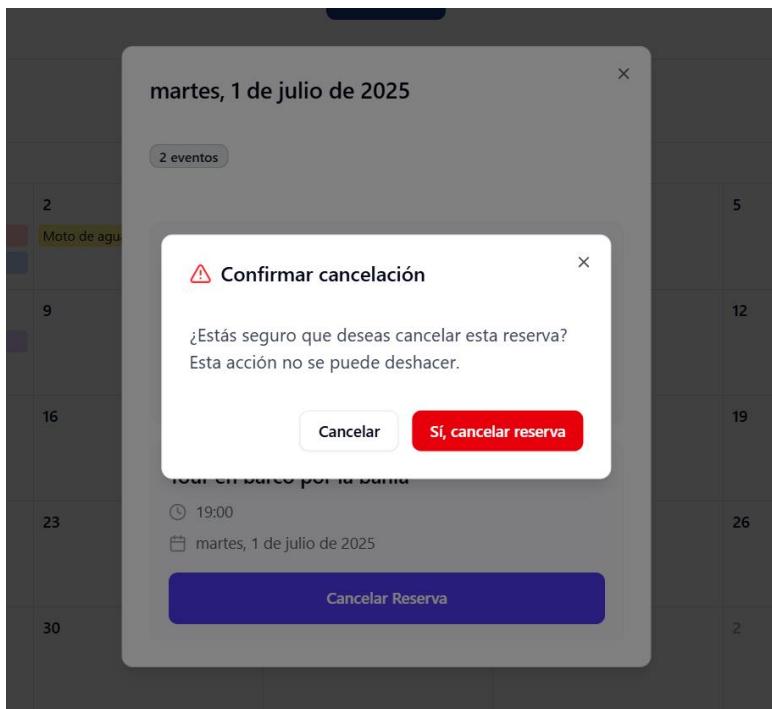


Ilustración 149 Mis Reservas III

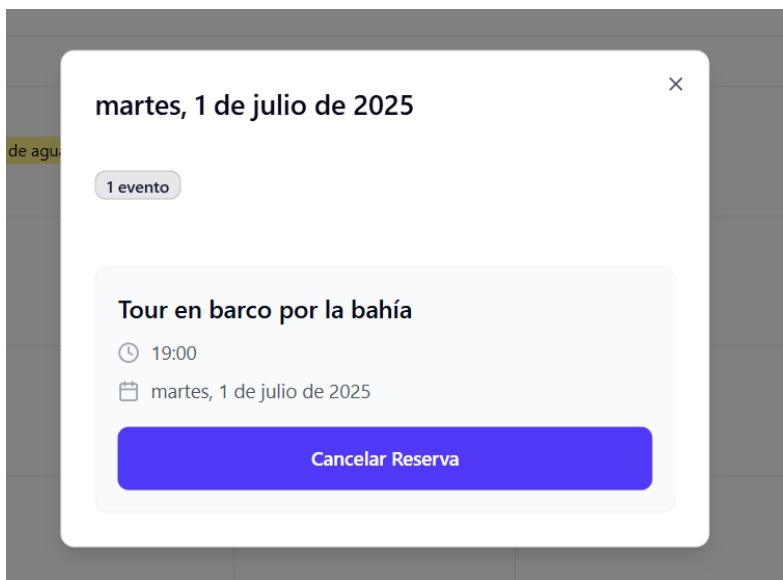


Ilustración 150 Mis reservas IV

Código – Mis Reservas

```

import React from 'react'
import Calendar from '../components/Calendar'

/**
 * Página de reservas.
 * Muestra el componente de calendario para gestionar o visualizar reservas.
 * @component
 * @returns {JSX.Element}
 */
export default function Reservas() {
  return (
    <>
      {/* Componente de calendario de reservas */}
      <Calendar />
    </>
  )
}

```

Ilustración 151 Reservas.jsx

Carrito.jsx

El carrito es el encargado de mostrar la información que ha sido almacenada en localStorage para ser almacenada como registro en la base de datos a través de una petición en el servidor.

No solo muestra los datos del carrito sino que también vuelve a realizar las peticiones pertinentes para comprobar que todos esos artículos pueden ser comprados.

Cuenta con comprobaciones como que la hora no esté pasada , es decir que la fecha de reserva sea mayor al día de hoy y la hora sea mayor que la actual en caso de que sea el día de hoy , además de esto también comprueba que la hora para dicho día siga estando disponible y que siga habiendo hueco para el número de personas indicado



Ilustración 152 Carrito I



Carrito

	Paseo en velero privado 1 Personas 16:00 h \$150	X
	Kayak por acantilados 2 Personas 13:00 h \$60	X

Order summary
 Seguro de Riesgos \$99.00
 Tasas estimadas \$8.32
Precio Reserva \$112.32
[Realizar Pedido](#)

Ilustración 153 Carrito II



Carrito

	Paseo en velero privado 1 Personas 16:00 h \$150	X
	Elemento Desfasado Hora seleccionada pasada, son las 09:36 Kayak por acantilados 2 Personas 8:00 h \$60	X

Order summary
 Seguro de Riesgos \$99.00
 Tasas estimadas \$8.32
Precio Reserva \$112.32
[Realizar Pedido](#)

Ilustración 154 Carrito III

Código – Cartito

```

import React, { useState, useEffect, useRef } from 'react';
import { API_URL, IMAGE_URL } from '../utilities/apirest';
import { motion } from "framer-motion"
import { Button } from "@/components/ui/button"
import axios from 'axios';
import { useNavigate } from "react-router-dom";

/**
 * Componente de carrito de reservas.
 * Permite ver, verificar disponibilidad y realizar reservas de productos seleccionados.
 * @component
 * @param {Object} props
 * @param {Array} props.cart - Array de productos en el carrito.
 * @param {Function} props.setCart - Función para actualizar el carrito.
 * @returns {JSX.Element}
 */
export default function Carrito({ cart, setCart }) {
  /**
   * Elementos del carrito que están disponibles.
   * {[Array, Function]}
   */
  const [carritoDesfase, setCarritoDesfase] = useState([]);
  /**
   * Elementos del carrito que no están disponibles o tienen problemas.
   * {[Array, Function]}
   */
  const [carritoNormal, setCarritoNormal] = useState([]);
  /**
   * Estado de carga.
   * {[boolean, Function]}
   */
  const [loading, setLoading] = useState(true);
  /**
   * Indica si todos los productos del carrito están disponibles para reservar.
   * {[boolean, Function]}
   */
  const [alquilerDisponible, setAlquilerDisponible] = useState(true);
  /**
   * Hook de navegación de React Router.
   */
  const navigate = useNavigate();
}

```

Ilustración 155 Carrito.jsx I

```

/**
 * Efecto para verificar disponibilidad cada vez que cambia el carrito.
 */
useEffect(() => {
  if (cart.length > 0) {
    verificarDisponibilidad();
  } else {
    setCarritoNormal([]);
    setCarritoDesfase([]);
    setLoading(false);
    setAlquilerDisponible(true);
  }
}, [cart]);

```

Ilustración 156 Carrito.jsx II

```

    /**
     * Verifica la disponibilidad de cada producto del carrito.
     * Clasifica los productos en disponibles y desfasados.
     */
const verificarDisponibilidad = async () => {
  setLoading(true);
  const token = localStorage.getItem("authToken");
  const headers = token ? { Authorization: `Bearer ${token}` } : {};

  const nuevosCarritoNormal = [];
  const nuevosCarritoDesfase = [];
  let disponible = true;

  await Promise.all(
    cart.map(async (item) => {
      try {
        // Verifica disponibilidad de hora
        const response1 = await axios.post(API_URL + "api/disponibilidadReserva", {
          idPublicacion: item.id,
          horaReserva: item.horaReserva + ":00"
        }, { headers });

        // Verifica capacidad disponible
        const response2 = await axios.post(API_URL + "api/capacidadDisponible", {
          idPublicacion: item.id
        }, { headers });

        // Obtiene fecha y hora actual del servidor
        const response3 = await axios.get(API_URL + "api/horaFecha", {}, { headers });

        const disponibilidad = response1.data.disponible;
        const maxReservables = response2.data.personas_disponibles;
        const mismaFecha = response3.data.fecha === item.fecha_evento.split(" ")[0];
        let horaFecha = true;

        if (mismaFecha) {
          const horaServ = parseInt(response3.data.hora.split(":")[0], 10);
          const horaItem = item.horaReserva;
          horaFecha = horaServ < horaItem;
        }
      }
    })
  );

  // Clasifica el producto según disponibilidad y restricciones
  if (maxReservables >= item.personas && disponibilidad && horaFecha) {
    nuevosCarritoNormal.push(item);
  } else {
    let causas = [];
    if (!disponibilidad) causas.push("No disponible en la hora seleccionada");
    if (item.personas > maxReservables) causas.push(`Solo hay ${maxReservables} plazas disponibles`);
    if (!horaFecha) causas.push(`Hora seleccionada pasada, son las ${response3.data.hora.split(":")[0]}:${response3.data.hora.split(":")[1]}`);
    item.causaDesfase = causas.join(" y ");
    nuevosCarritoDesfase.push(item);
    disponible = false;
  }
} catch (error) {
  console.error("Error al verificar disponibilidad para el ítem:", item.id, error);
}
);

setCarritoNormal(nuevosCarritoNormal);
setCarritoDesfase(nuevosCarritoDesfase);
setAlquilerDisponible(disponible);
setLoading(false);
};

/**
 * Elimina un producto del carrito.
 * @param {number} id - ID del producto a eliminar.
 */
const handleEliminarProducto = (id) => {
  const nuevoCart = cart.filter(item => item.id !== id);
  setCart(nuevoCart);
  localStorage.setItem("cart", JSON.stringify(nuevoCart));
};

```

Ilustración 157 Carrito jsx III

```

// clasifica el producto según disponibilidad y restricciones
if (maxReservables >= item.personas && disponibilidad && horaFecha) {
  nuevosCarritoNormal.push(item);
} else {
  let causas = [];
  if (!disponibilidad) causas.push("No disponible en la hora seleccionada");
  if (item.personas > maxReservables) causas.push(`Solo hay ${maxReservables} plazas disponibles`);
  if (!horaFecha) causas.push(`Hora seleccionada pasada, son las ${response3.data.hora.split(":")[0]}:${response3.data.hora.split(":")[1]}`);
  item.causaDesfase = causas.join(" y ");
  nuevosCarritoDesfase.push(item);
  disponible = false;
}
} catch (error) {
  console.error("Error al verificar disponibilidad para el ítem:", item.id, error);
}
);

setCarritoNormal(nuevosCarritoNormal);
setCarritoDesfase(nuevosCarritoDesfase);
setAlquilerDisponible(disponible);
setLoading(false);
};

/**
 * Elimina un producto del carrito.
 * @param {number} id - ID del producto a eliminar.
 */
const handleEliminarProducto = (id) => {
  const nuevoCart = cart.filter(item => item.id !== id);
  setCart(nuevoCart);
  localStorage.setItem("cart", JSON.stringify(nuevoCart));
};

```

Ilustración 158 Carrito jsx IV

```

/**
 * Realiza el pedido de todos los productos del carrito.
 * Envía las reservas a la API y limpia el carrito si tiene éxito.
 */
const handleRealizarPedido = async () => {
  const url = API_URL + "api/reservas";
  const token = localStorage.getItem("authToken");
  const headers = token ? { Authorization: `Bearer ${token}` } : {};

  setLoading(true);

  const peticiones = cart.map((elemento) => {
    return axios.post(url, {
      publicacion_id: elemento.id,
      hora_reserva: elemento.horaReserva.toString().padStart(2, "0") + ":00",
      total_pagar: elemento.precio,
      personas: elemento.personas
    }, { headers });
  });

  try {
    const respuestas = await Promise.all(peticiones);
    console.log("Reservas creadas:", respuestas.map(r => r.data.reserva));

    setCart([]);
    localStorage.setItem("cart", []);
    setCarritoDesfase([]);
    setCarritoNormal([]);
  } catch (error) {
    console.error("Error al crear una o más reservas:", error);
    alert("Hubo un problema al registrar alguna reserva.");
  } finally {
    setLoading(false);
  }
};


```

Ilustración 159 Carrito.jsx V

```

return (
  <div className="max-w-7xl mx-auto p-10">
    {/* Si el carrito está vacío, muestra mensaje y botón para ir a alquilables */}
    {carritoDesfase.length == 0 && carritoNormal.length == 0 && !loading ? (
      <div className="max-w-4xl mx-auto px-4 py-12">
        <motion.div
          initial={[ opacity: 0, y: 20 ]}
          animate={[ opacity: 1, y: 0 ]}
          transition={{ duration: 0.5 }}
          className="bg-blue-50 rounded-lg p-8 md:p-12 flex flex-col md:flex-row items-center justify-between">
          <div className="md:w-1/2 mb-8 md:mb-0 md:pr-8">
            <h2 className="text-2xl md:text-3xl font-bold tracking-tight uppercase text-gray-800 mb-2">Carrito Vacío</h2>
            <div className="w-20 h-1 bg-blue-600 mb-6"></div>
            <p className="text-gray-600 mb-8">
              Aún no has añadido ningún evento al carrito. Descubre nuestras sombrillas, tumbonas, paddle surf y mucho más para disfrutar del sol.
            </p>
            <Button
              onClick={() => navigate("/alquilables")}
              className="bg-blue-600 hover:bg-blue-700 text-white px-8 py-6 rounded-md font-medium text-lg"
            >
              ¡ALQUILA AHORA!
            </Button>
          </div>
        </motion.div>
        <motion.div
          className="md:w-1/2 relative"
          animate={{
            y: [0, -10, 0],
          }}
          transition={{
            repeat: Number.POSITIVE_INFINITY,
            duration: 3,
            ease: "easeInOut",
          }}
        >
          
        </motion.div>
      </motion.div>
    ) : (
      <div>
        <h2>Tu carrito</h2>
        <table border="1">
          <thead>
            <tr>
              <th>Artículo</th>
              <th>Descripción</th>
              <th>Precio</th>
              <th>Acciones</th>
            </tr>
          </thead>
          <tbody>
            {carritoNormal.map((item) => (
              <tr key={item.id}>
                <td>{item.nombre}</td>
                <td>{item.descripcion}</td>
                <td>${item.precio}</td>
                <td>
                  <button>Quitar</button>
                  <button>Actualizar</button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
        <div>
          <button>Vaciar carrito</button>
          <button>Continuar compras</button>
        </div>
      </div>
    )}
  </div>
);

```

Ilustración 160 Carrito.jsx VI

```

) : (
  <>

  {/* Título y contenedor principal */}
  <h1 className="text-4xl font-semibold mb-6">Carrito</h1>
  <div className="flex flex-col md:flex-row gap-8">

    {/* Lista de productos disponibles y desfasados */}
    <div className="flex-1 max-h-[600px] overflow-y-auto pr-2 space-y-6">
      {/* Productos disponibles */}
      {carritoNormal.map((producto) => (
        <div key={producto.id} className="flex items-center gap-6 border-b pb-6">
          <img
            | src={IMAGE_URL + producto.imagen.split(";")[0]}
            | alt={producto.titulo}
            | className="h-24 w-24 rounded"
          />
          <div className="flex-1">
            <h2 className="text-lg font-medium">{producto.titulo}</h2>
            <p className="text-gray-500">{producto.personas} Personas | {producto.horaReserva}:00 h</p>
            <p className="font-semibold mt-2">${producto.precio}</p>
          </div>
          <button
            | className="text-gray-400 hover:text-gray-600"
            | onClick={() => handleEliminarProducto(producto.id)}
          >
            X
          </button>
        </div>
      ))}
    </div>
  </div>
)

```

Ilustración 161 Carrito jsx VII

```

  {/* Resumen del pedido o loader */}
  {loading ? (
    <div className="flex justify-center items-center w-full md:w-1/3">
      <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
    </div>
  ) : (
    <div className="w-full md:w-1/3 bg-gray-50 p-6 rounded-lg h-fit md:top-10">
      <h2 className="text-xl font-semibold mb-4">Order summary</h2>
      <div className="flex justify-between text-sm mb-2">
        <span>Seguro de Riesgos</span>
        <span>$99.00</span>
      </div>
      <div className="flex justify-between text-sm mb-4">
        <span>Taxes estimadas</span>
        <span>$8.32</span>
      </div>
      <div className="flex justify-between font-semibold text-lg mb-6">
        <span>Precio Reserva</span>
        <span>$112.32</span>
      </div>
      {alquilerDisponible ? (
        <button className="w-full bg-indigo-600 text-white py-2 rounded hover:bg-indigo-700 transition"
          | onClick={handleRealizarPedido}
          | Realizar Pedido
        </button>
      ) : (
        <button className="w-full bg-indigo-300 text-white py-2 rounded cursor-not-allowed">
          | Realizar Pedido
        </button>
      )}
    </div>
  )
}

);

```

Ilustración 162 Carrito jsx VIII

ai.jsx

Además de todas las secciones mencionadas anteriormente, contamos con IA gracias a la API gratuita de [cohore](#), esto permite tener un chat como chatgpt en la página web.

Gracias a esto podremos chatear hasta que se nos acabe los prompts.

Este chat funciona haciendo peticiones POST a la API de cohore.

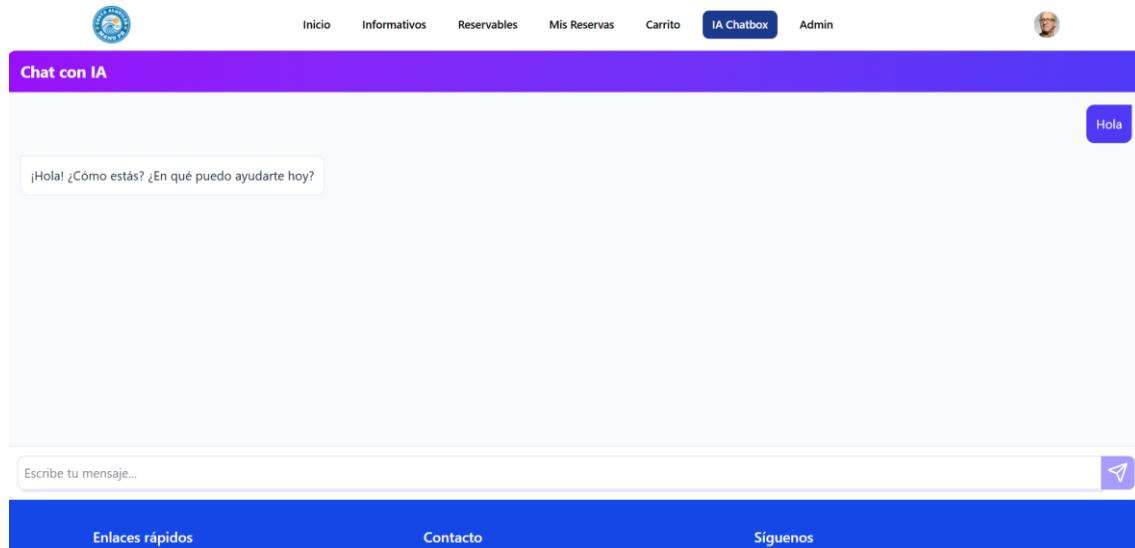


Ilustración 163 IA Chatbox

Código – ai

```
/* @file ai.jsx
 * @description Página de chat con IA usando Cohere. Permite enviar mensajes y recibir respuestas de la IA.
 */

import { useEffect, useState, useRef } from "react"
import { CohereClientV2 } from "cohere-ai"
import { Send } from "lucide-react"

/***
 * Componente de chat con IA Cohere.
 * Permite enviar mensajes y recibir respuestas en tiempo real.
 * @component
 * @returns {JSX.Element}
 */
const Coherechat = () => [
  /**
   * Lista de mensajes del chat.
   * {[Array, Function]}
   */
  const [messages, setMessages] = useState([])
  /**
   * Mensaje de entrada del usuario.
   * ((string|Function)[])
   */
  const [inputMessage, setInputMessage] = useState("")
  /**
   * Estado de carga mientras se espera la respuesta de la IA.
   * {[boolean, Function]}
   */
  const [isloading, setIsloading] = useState(false)
  /**
   * Referencia al final de la lista de mensajes para hacer scroll automático.
   * {React.RefObject}
   */
  const messagesEndref = useRef(null)
  // Instancia del cliente Cohere
  const cohore = new CohereClientV2({
    token: "b6oxyU9gisIciEy3l00hkZtpKJyGaX77cJC0qnW0",
  })
]
```

Ilustración 164 ai.jsx /

```
/**
 * Envía un mensaje al modelo Cohere y actualiza el chat.
 * @param {string} content - Mensaje del usuario.
 */
const sendMessage = async (content) => {
  if (!content.trim()) return;

  // Añade el mensaje del usuario al chat
  const userMessage = { role: "user", content };
  setMessages([...prev, userMessage]);

  setIsLoading(true);

  try {
    // Llama a la API de Cohere con el historial de mensajes
    const res = await cohere.chat({
      model: "command-a-03-2025",
      messages: [...messages.map((msg) => ({ role: msg.role, content: msg.content })), { role: "user", content }],
    });

    // Añade la respuesta de la IA al chat
    if (res.message) {
      const assistantMessage = {
        role: "assistant",
        content: res.message.content[0]?.text || "No response content",
      };
      setMessages([...prev, assistantMessage]);
    }
  } catch (error) {
    console.error("Error calling Cohere API:", error);
    setMessages([...prev, { role: "assistant", content: "Sorry, I encountered an error processing your request." }]);
  } finally {
    setIsLoading(false)
  }
}
```

Ilustración 165 ai.jsx II

```
/**
 * Maneja el envío del formulario (mensaje).
 * @param {React.FormEvent} e
 */
const handleSubmit = (e) => {
  e.preventDefault();
  sendMessage(inputMessage);
}

return (
  <div className="flex flex-col h-[600px] mx-auto border rounded-lg bg-white">
    /* Chat header */
    <div className="px-4 py-3 bg-gradient-to-r from-purple-600 to-indigo-600 text-white rounded-t-lg">
      <h1 className="text-xl font-bold">Chat con IA</h1>
    </div>

    /* Contenedor de mensajes */
    <div className="flex-1 p-4 overflow-y-auto bg-gray-50">
      {messages.length === 0 ? (
        <div className="flex items-center justify-center h-full text-gray-400">Empieza a conversar...</div>
      ) : (
        messages.map((message, index) => (
          <div key={index} className={mb-4 flex ${message.role === "user" ? "justify-end" : "justify-start"}}>
            <div className={`max-w-[80%] p-3 rounded-lg ${message.role === "user" ? "bg-indigo-600 text-white rounded-tr-none" : "bg-white border border-gray-200 text-gray-800 rounded-tl-none"}`}>
              {message.content}
            </div>
          </div>
        )));
    </div>
  </div>
)
```

Ilustración 166 ai.jsx III

```
/* Loader de la IA */
{isLoading && (
  <div className="flex justify-start mb-4">
    <div className="bg-white border border-gray-200 p-3 rounded-lg rounded-tl-none text-gray-800">
      <div className="flex space-x-2">
        <div>
          className="w-2 h-2 rounded-full bg-gray-300 animate-bounce"
          style={{ animationDelay: "0ms" }}
        </div>
        <div>
          className="w-2 h-2 rounded-full bg-gray-300 animate-bounce"
          style={{ animationDelay: "150ms" }}
        </div>
        <div>
          className="w-2 h-2 rounded-full bg-gray-300 animate-bounce"
          style={{ animationDelay: "300ms" }}
        </div>
      </div>
    </div>
  </div>
)
<div ref={messagesEndRef} />
</div>
```

Ilustración 167 ai.jsx IV

```
/* Area de entrada de texto */
<form onSubmit={handleSubmit} className="p-3 border-t border-gray-200 bg-white rounded-b-lg">
  <div className="flex items-center">
    <input
      type="text"
      value={inputMessage}
      onChange={(e) => setInputMessage(e.target.value)}
      placeholder="Escribe tu mensaje..."
      className="flex-1 p-2 border border-gray-300 rounded-l-md focus:outline-none focus:ring-2 focus:ring-indigo-500"
      disabled={isLoading}
    />
    <button
      type="submit"
      className="p-2 bg-indigo-600 text-white rounded-r-md hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-indigo-500 disabled={isLoading || !inputMessage.trim()}"
      >
      | <Send size={28} />
    </button>
  </div>
</form>
</div>
)
}

export default CohereChat
```

Ilustración 168 ai.jsx V

Admin.jsx

El apartado de Admin es aquel el cual es usado para administrar las publicaciones los usuarios y las reservas de la página web.

Esta se divide en 3 apartados: publicaciones, usuarios y reservas todas las secciones paginadas.

Estas tres secciones están compuestas por varios componentes , uno para cada uno de ellos por ejemplo para la parte de usuarios tengo usuarios.jsx que muestra todos los usuarios y para cuando abro el modal tengo modal-usuarios.jsx, lo mismo para publicaciones y reservas , estos serán explicados en los siguientes apartados.

Panel de Administración

Gestiona usuarios, reservas y anuncios desde este panel

Publicaciones Usuarios Reservas

Buscar publicaciones... Añadir nuevo producto

Día en la playa con niños 2025-07-01 00:00:00	Atardeceres en la costa 2025-07-01 00:00:00	Curso de cuidado de la Piel 2025-07-02 00:00:00	Avistamiento de delfines 2025-07-03 00:00:00
Gratis Editar Eliminar	Gratis Editar Eliminar	Gratis Editar Eliminar	Gratis Editar Eliminar

Ilustración 169 Admin Publicaciones I

Normas de seguridad en el mar 2025-07-03 00:00:00	Picnic playero 2025-07-04 00:00:00	Paddle surf para principiantes 2025-07-04 00:00:00	Evitar corrientes marinas 2025-07-05 00:00:00
Gratis Editar Eliminar	Gratis Editar Eliminar	Gratis Editar Eliminar	Gratis Editar Eliminar

< Página 1 de 4 >

Enlaces rápidos Contacto Síguenos

Ilustración 170 Admin Publicaciones II

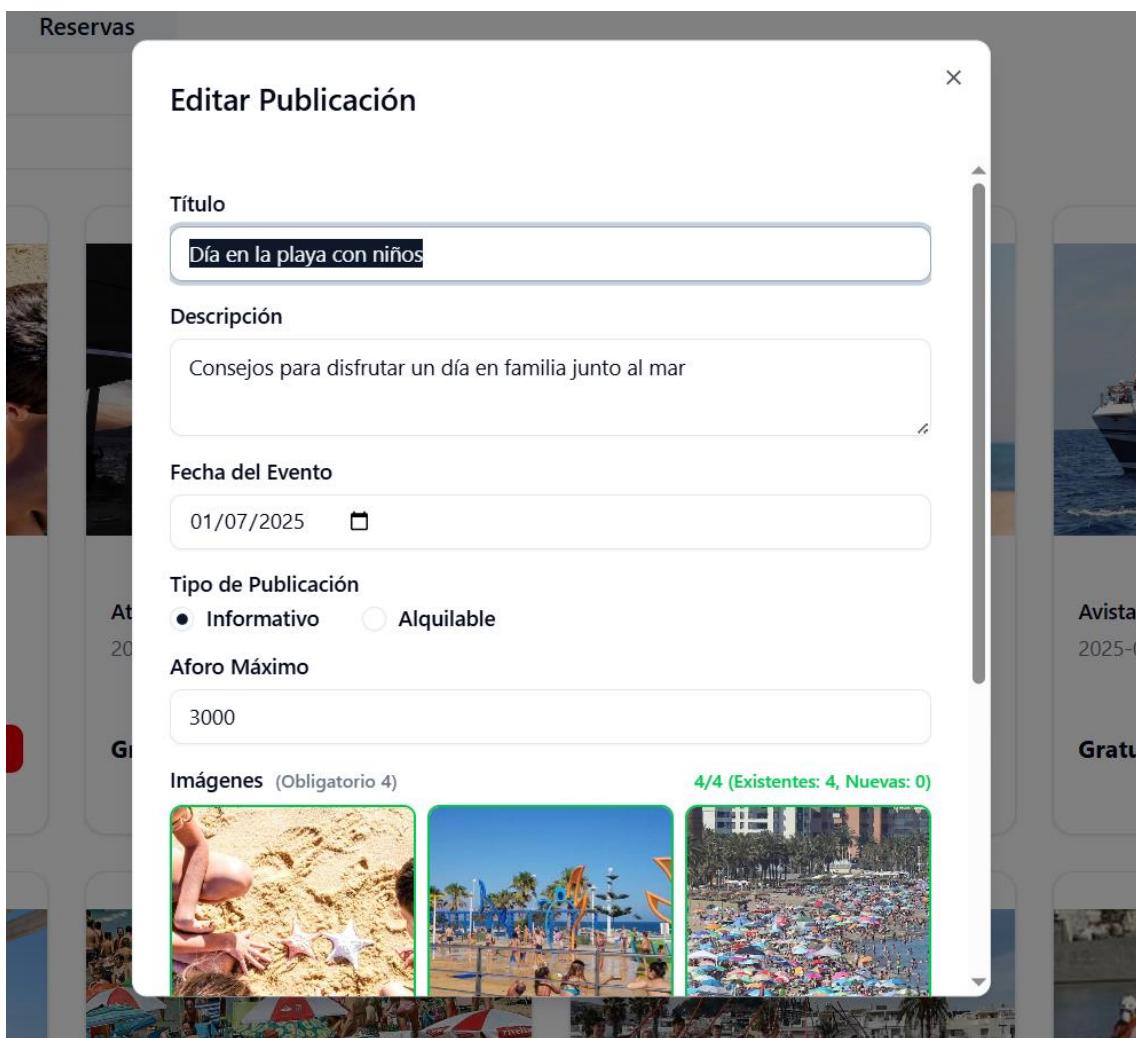


Ilustración 171 Admin Publicaciones III – Editar

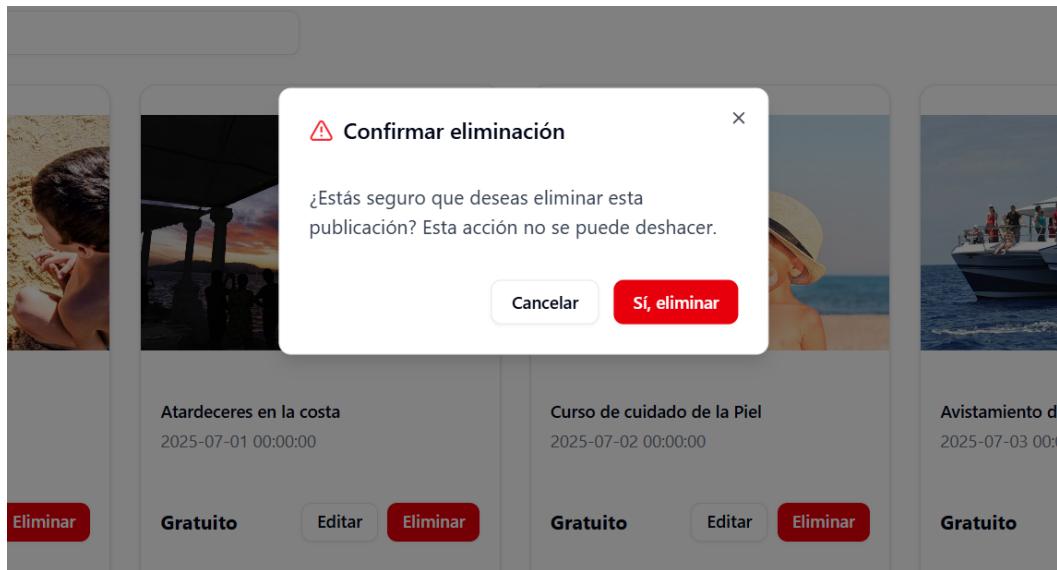


Ilustración 172 Admin Publicaciones IV – Eliminar

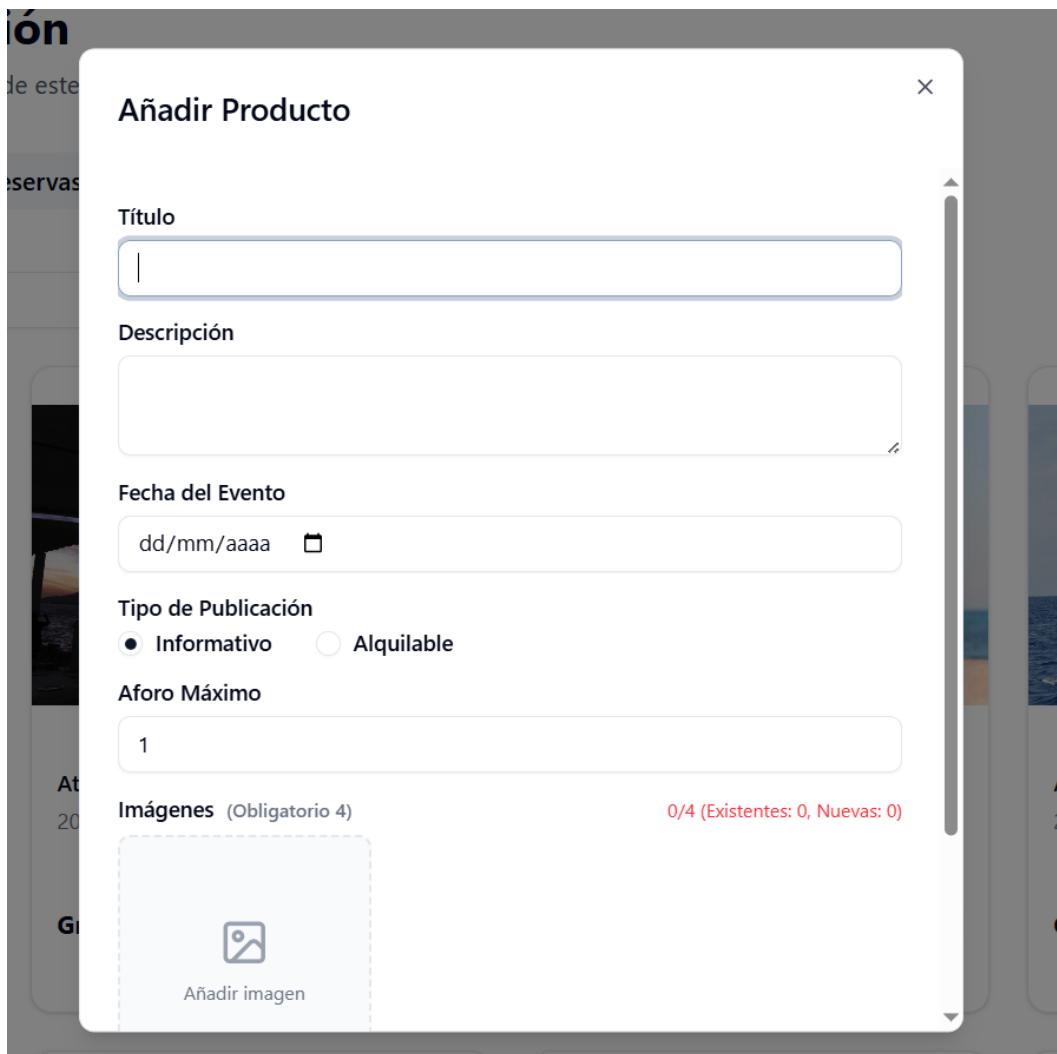


Ilustración 173 Admin Publicaciones V – Añadir



Panel de Administración

Gestiona usuarios, reservas y anuncios desde este panel

		Publicaciones	Usuarios	Reservas
<input type="text"/>	Buscar usuarios...			
				<button>Añadir Usuario</button>
USUARIO	EMAIL	FECHA NACIMIENTO	ROL	RESTABLECER CONTRASEÑA
MG Manuelg Gonzalez	manuel5365.mgl@gmail.com	2004-01-03	admin	<button>Enviar Restablecimiento</button>
MC Manuel Cuenta222	manuelgonzalezperez@iesflorenciopintado.es	2024-11-29	usuario	<button>Enviar Restablecimiento</button>
LK Lennie Kessler	pcmanuela3@gmail.com	2011-11-04	usuario	<button>Enviar Restablecimiento</button>
AB Axel Bins	gonzalezmolerot@gmail.com	1979-03-02	admin	<button>Enviar Restablecimiento</button>

Ilustración 174 Admin Usuarios I

EP	Elinore Pfannerstill	terry.elouise@example.org	1973-05-05	admin	Enviar Restablecimiento	<button>Editar</button> <button>Borrar</button>
LW	Louisa Waelchi	slegros@example.com	2015-01-11	usuario	Enviar Restablecimiento	<button>Editar</button> <button>Borrar</button>
GS	Graham Spinka	zskiles@example.net	1990-08-07	usuario	Enviar Restablecimiento	<button>Editar</button> <button>Borrar</button>
SR	Sim Rodriguez	stuart04@example.org	2008-02-21	admin	Enviar Restablecimiento	<button>Editar</button> <button>Borrar</button>
RA	River Adams	carol.quitzon@example.com	2008-09-09	admin	Enviar Restablecimiento	<button>Editar</button> <button>Borrar</button>

< Página 1 de 2 >



Ilustración 175 Admin Usuarios II

Editar Usuario

Modifica la información del usuario seleccionado.

Nombre

Apellidos

Email

Fecha de Nacimiento

Rol

Cancelar **Guardar cambios**

Ilustración 176 Admin Usuarios III – Editar

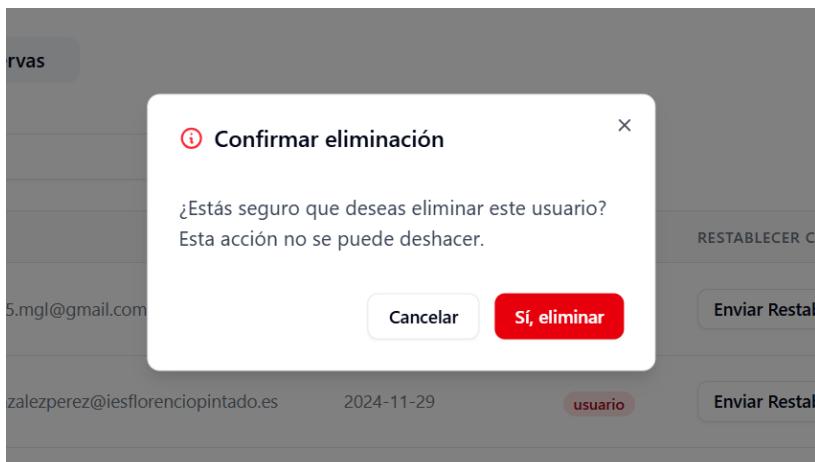


Ilustración 177 Admin Usuarios IV – Eliminar

Editar Usuario

Modifica la información del usuario seleccionado.

Nombre

Apellidos

Email

Fecha de Nacimiento

Contraseña

Rol

Selecciona un rol

Cancelar Guardar cambios

Ilustración 178 Admin Usuarios V – Añadir

The screenshot shows the 'Panel de Administración' (Administration Panel) with a sub-header 'Gestiona usuarios, reservas y anuncios desde este panel'. Below this are tabs for 'Publicaciones', 'Usuarios', and 'Reservas', with 'Reservas' being the active tab. A search bar labeled 'Buscar reservas...' is present. The main content is a table listing bookings:

ID	USUARIO	ANUNCIO	FECHA	ESTADO	ACCIONES
61	Cristian Redondo Martínez	Tour histórico en barco	2025-07-07	Reservada	<button>Editar</button> <button>Cancelar</button>
63	Cristian Redondo Martínez	Fiesta en catamarán	2025-07-02	Reservada	<button>Editar</button> <button>Cancelar</button>
64	Cristian Redondo Martínez	Pesca deportiva con patrón	2025-07-04	Reservada	<button>Editar</button> <button>Cancelar</button>
65	Manuelg Gonzalez	Alquiler equipo de pesca	2025-07-06	Reservada	<button>Editar</button> <button>Cancelar</button>

Ilustración 179 Admin Reservas I

The screenshot shows a continuation of the booking list from the previous interface:

69	Manuelg Gonzalez	Paseo en helicóptero	2025-07-03	<button>Editar</button> <button>Cancelar</button>
70	Manuelg Gonzalez	Paseo en velero privado	2025-07-03	<button>Editar</button> <button>Cancelar</button>

Below the table are navigation arrows and a page number 'Página 1 de 1'.

Ilustración 180 Admin Reservas II

The screenshot shows the 'Editar Reserva' (Edit Reservation) modal window. It contains fields for 'Usuario' (User), 'Publicación' (Publication), 'Fecha de Reserva' (Reservation Date), 'Total a Pagar' (Total to Pay), and a checkbox 'Notificar cambio al usuario' (Notify user of change). On the right, there are sections for 'Hora de Reserva' (Reservation Time) with a slider set at 21:00 PM, and 'Número de Personas' (Number of People) with a selection of 3 highlighted in blue. At the bottom are 'Cerrar' (Close) and 'Guardar cambios' (Save changes) buttons.

Ilustración 181 Admin Reservas III – Editar

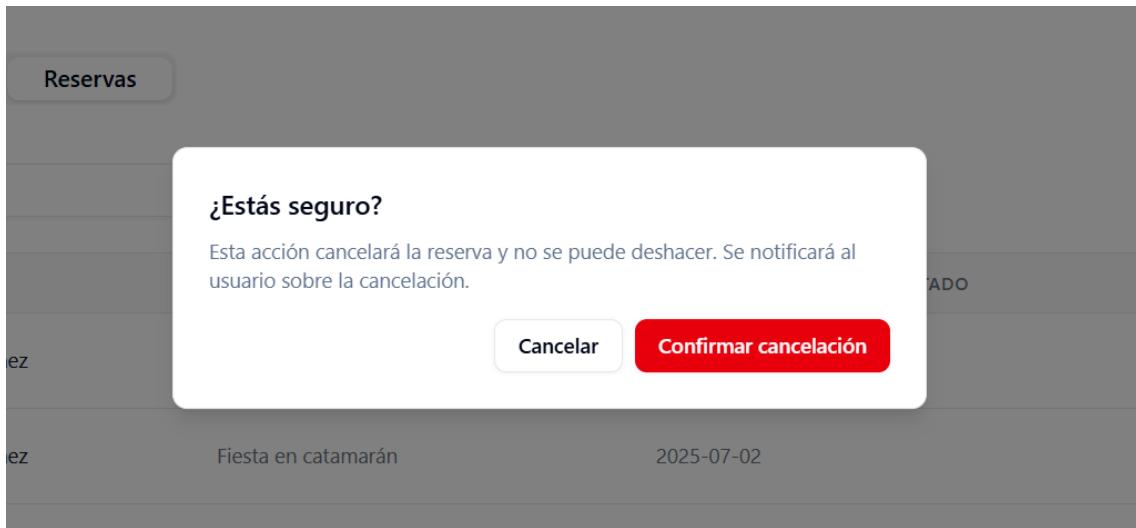


Ilustración 182 Admin Reservas IV - Eliminar

Código – Admin

```
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";

// Importa los componentes de administración
import Publicaciones from "../components/AdminTab/publicaciones";
import Reservas from "../components/AdminTab/reservas";
import Usuarios from "../components/AdminTab/usuarios";

/**
 * Panel de administración principal.
 * Permite gestionar usuarios, reservas y publicaciones mediante pestañas.
 * @component
 * @returns {JSX.Element}
 */
export default function AdminPanel() {
  return (
    <div className="min-h-screen bg-white">
      </> Contenedor principal del panel </>
      <main className="container mx-auto px-4 py-8">
        </> Título y descripción </>
        <h1 className="text-3xl font-bold mb-2">Panel de Administración</h1>
        <p className="text-gray-600 mb-8">
          Gestiona usuarios, reservas y anuncios desde este panel
        </p>

        </> Tabs para navegar entre secciones </>
        <Tabs defaultValue="anuncios" className="w-full">
          <TabsList className="mb-8">
            <TabsTrigger value="anuncios" className="text-base px-6">
              | Publicaciones
            </TabsTrigger>
            <TabsTrigger value="usuarios" className="text-base px-6">
              | Usuarios
            </TabsTrigger>
            <TabsTrigger value="reservas" className="text-base px-6">
              | Reservas
            </TabsTrigger>
          </TabsList>
        </Tabs>
      </main>
    </div>
  )
}
```

Ilustración 183 Admin jsx /

```

    /* Sección de publicaciones */
    <TabsContent value="anuncios">
        <Publicaciones />
    </TabsContent>
    /* Sección de usuarios */
    <TabsContent value="usuarios">
        <Usuarios />
    </TabsContent>
    /* Sección de reservas */
    <TabsContent value="reservas">
        <Reservas />
    </TabsContent>
</Tabs>
</main>
</div>
);
}

```

Ilustración 184 Admin jsx II

Perfil.jsx

La página de Perfil es aquella que se encarga de mostrar los datos del perfil del usuario, accediendo a ella a través de un click en el navbar y dando en mi perfil.

Esta parte hace peticiones al servidor para obtener los datos del perfil y para modificar los datos usa el componente modal-usuarios-perfil.jsx que es el encargado de modificar los datos.

The screenshot shows the Admin I profile page. At the top, there's a navigation bar with links: Inicio, Informativos, Reservables, Mis Reservas, Carrito, IA Chatbox, and Admin. On the far right is a user icon. Below the navigation is a banner featuring a blue sea and white waves. The banner displays the user's name, "Manuelg Gonzalez", and the text "Miembro desde 2025".

The main content area has two sections:

- Información Personal**: Details of your account. It includes a circular profile picture with initials "MG", the name "Manuelg Gonzalez", and the role "Administrador". Below this is an "Email" section with the address "manuel5365.mgl@gmail.com".
- Actividad Reciente**: History of your last activities. It shows two recent reservations:
 - Reserva: Alquiler equipo de pesca**: Total paid: 15 € | Persons: 1. Date: 6/7/2025.
 - Reserva: Moto de agua 30 min**: Total paid: 90 € | Persons: 2. Date: 2/7/2025.

Ilustración 185 Admin I

The screenshot shows a user profile with the following details:

- Fecha de nacimiento:** 3 de enero de 2004 (21 años)
- Tipo de cuenta:** Admin
- Última modificación del perfil:** 2025-06-03 11:26:46
- Editar perfil:** Button

Recent bookings listed:

- Reserva: Parasailing**
Total pagado: 660 € | Personas: 3
Fecha: 8/7/2025
- Reserva: Tour en barco por la bahía**
Total pagado: 180 € | Personas: 3
Fecha: 1/7/2025
- Reserva: Paseo en helicóptero**
Total pagado: 250 € | Personas: 1
Fecha: 3/7/2025
- Reserva: Paseo en velero privado**
Total pagado: 150 € | Personas: 1
Fecha: 3/7/2025

Footer navigation links:

- Enlaces rápidos**
- Contacto**: manuel5365@gmail.com
- Síguenos**: Icons for Instagram, Facebook, Twitter, and LinkedIn

Ilustración 186 Admin II

The screenshot shows a user profile with the following details:

- Nombre:** Manuelg Gonzalez
- Tipo de cuenta:** Administrador
- Última modificación del perfil:** 2025-06-03 11:26:46
- Editar perfil:** Button

A modal window titled "Editar Perfil" is open, showing fields for:

- Nombre:** Manuelg
- Apellidos:** Gonzalez
- Fecha de Nacimiento:** 03/01/2004
- Seguridad:** Restablecer contraseña (Forgot password)

Buttons at the bottom of the modal are "Cancelar" (Cancel) and "Guardar cambios" (Save changes).

Below the modal, a recent booking is listed:

- Reserva: Paseo en helicóptero**
Total pagado: 250 € | Personas: 1

Ilustración 187 Admin III

Código – Perfil

```
/** 
 * @file Perfil.jsx
 * @description Página de perfil de usuario. Permite ver y editar información personal, así como consultar la actividad reciente (reservas).
 */
import { useState, useEffect } from "react"
import { Card,CardContent,CardDescription,CardFooter,CardHeader,CardTitle } from "@/components/ui/card"
import { Avatar,AvatarFallback,AvatarImage } from "@/components/ui/avatar"
import { Badge } from "@/components/ui/badge"
import { Button } from "@/components/ui/button"
import axios from "axios"
import { API_URL, IMAGE_URL } from "../utilities/apiрест"
import { Tabs,TabsContent,TabsList,TabsTrigger } from "@/components/ui/tabs"
import { CalendarDays,MapPin,Mail,Phone,Shield,Clock,Anchor,Ship,Compass } from 'lucide-react'
import UsuariosModal from "../components/AdminTab/modal-usuarios-perfil" // Importamos el modal de usuarios

/**
 * Componente de perfil de usuario.
 * Permite ver y editar datos personales y consultar la actividad reciente.
 * @component
 * @returns {JSX.Element}
 */
export default function PerfilPage() {
    /**
     * Estado con los datos del perfil del usuario.
     * {[Object|null, Function]}
     */
    const [profile, setProfile] = useState(null);
    /**
     * Estado de carga de la página.
     * {[boolean, Function]}
     */
    const [loading, setLoading] = useState(true);
    /**
     * Pestaña activa en el perfil.
     * {string|Function}[]
     */
    const [activeTab, setActiveTab] = useState("perfil");
    /**
     */
}
```

Ilustración 188 Perfil.jsx I

```
const [activeTab, setActiveTab] = useState("perfil");
/**
 * Reservas del usuario.
 * {[Array, Function]}
 */
const [reservas, setReservas] = useState([]);
/**
 * Imágenes asociadas a las reservas.
 * {[Array, Function]}
 */
const [imagenes, setImagenes] = useState([]);
/**
 * Estado para controlar la apertura del modal de edición de usuario.
 * {[boolean, Function]}
 */
const [isModalOpen, setIsModalOpen] = useState(false);

/**
 * Efecto para cargar perfil y reservas al montar el componente o al cerrar el modal.
 */
useEffect(() => {
    fetchProfile();
    fetchReservas();
}, [isModalOpen]); // Actualiza cuando se cierra el modal
```

Ilustración 189 Perfil.jsx II

```

    /**
     * obtiene los datos del perfil del usuario desde la API.
     */
    const fetchProfile = async () => {
        const token = localStorage.getItem("authToken");
        const headers = token ? { Authorization: `Bearer ${token}` } : {};
        try {
            const response = await axios.get(API_URL + "api/getData", { headers });
            setProfile(response.data);
        } catch (error) {
            console.error("Error al cargar los datos del perfil:", error);
        } finally {
            setLoading(false);
        }
    };

    /**
     * obtiene las reservas del usuario desde la API.
     */
    const fetchReservas = async () => {
        const token = localStorage.getItem("authToken");
        const headers = token ? { Authorization: `Bearer ${token}` } : {};
        try {
            const response = await axios.get(API_URL + "api/obtenerReservasUsuario", { headers });
            setReservas(response.data);
        } catch (error) {
            console.error("Error al cargar los datos del perfil:", error);
        } finally {
            setLoading(false);
        }
    };
}

```

Ilustración 190 Perfil.jsx III

```

    /**
     * Calcula la edad a partir de la fecha de nacimiento.
     * @param {string} fechaNacimiento
     * @returns {number}
     */
    const calcularEdad = (fechaNacimiento) => {
        const fechaNac = new Date(fechaNacimiento);
        const hoy = new Date();
        let edad = hoy.getFullYear() - fechaNac.getFullYear();
        const mes = hoy.getMonth() - fechaNac.getMonth();
        if (mes < 0 || (mes === 0 && hoy.getDate() < fechaNac.getDate())) {
            edad--;
        }
        return edad;
    };

    /**
     * Formatea una fecha a formato legible en español.
     * @param {string} fecha
     * @returns {string}
     */
    const formatearFecha = (fecha) => {
        const options = { year: 'numeric', month: 'long', day: 'numeric' };
        return new Date(fecha).toLocaleDateString('es-ES', options);
    };

    /**
     * Abre el modal de edición de perfil.
     */
    const handleEditProfile = () => {
        setIsModalOpen(true);
    };
}

```

Ilustración 191 Perfil.jsx IV

```
// Loader mientras se cargan los datos
if (loading) {
    return (
        <div className="flex justify-center items-center h-64">
            <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
        </div>
    );
}

// Si no hay perfil, muestra mensaje de error
if (!profile) {
    return (
        <div className="flex flex-col items-center justify-center min-h-screen">
            <h1 className="text-2xl font-bold text-red-500">Error al cargar el perfil</h1>
            <p className="text-gray-600 mt-2">No se pudieron obtener los datos del usuario</p>
            <button className="mt-4" onClick={() => window.location.reload()}>
                Intentar de nuevo
            </button>
        </div>
    );
}

// Formatea la fecha de última actualización del perfil
const updated_at = profile.updated_at.split("T")[0] + " " + profile.updated_at.split("T")[1].split(".")[0];
```

Ilustración 192 Perfil.jsx V

```
return (
    <div className="container mx-auto px-4 py-8">
        {/* Banner de portada con nombre y año de registro */}
        <div className="mb-8">
            <div className="relative h-48 w-full rounded-xl overflow-hidden mb-4">
                
                <div className="absolute inset-0 bg-gradient-to-t from-black/60 to-transparent"></div>
                <div className="absolute bottom-4 left-4 text-white">
                    <h1 className="text-3xl font-bold">{profile.Nombre} {profile.Apellidos}</h1>
                    <p className="text-sm opacity-90">
                        Miembro desde {new Date(profile.created_at).getFullYear()}
                    </p>
                </div>
            </div>
        </div>

        <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
            {/* Tarjeta de información personal */}
            <div className="md:col-span-1">
                <Card>
                    <CardHeader className="pb-2">
                        <CardTitle>Información Personal</CardTitle>
                        <CardDescription>Detalles de tu cuenta</CardDescription>
                    </CardHeader>
                    <CardContent>
                        <div className="flex flex-col items-center mb-6">
                            <Avatar className="h-24 w-24 mb-4">
                                <AvatarImage src={`https://api.dicebear.com/7.x/initials/svg?seed=${profile.Nombre} ${profile.Apellidos}`}>
                                <AvatarFallback>{profile.Nombre.charAt(0)}{profile.Apellidos.charAt(0)}</AvatarFallback>
                            </Avatar>
                            <h2 className="text-xl font-semibold">{profile.Nombre} {profile.Apellidos}</h2>
                            <Badge className="mt-2" variant={profile.Tipo === "admin" ? "destructive" : "default"}>
                                {profile.Tipo === "admin" ? "Administrador" : "Usuario"}
                            </Badge>
                        </div>
                    </CardContent>
                </Card>
            </div>
        </div>
    </div>
```

Ilustración 193 Perfil.jsx VI

```

<div className="space-y-4">
  {/* Email */}
  <div className="flex items-center gap-3">
    <Mail className="h-5 w-5 text-muted-foreground" />
    <div>
      <p className="text-sm font-medium">Email</p>
      <p className="text-sm text-muted-foreground">{profile.Email}</p>
    </div>
  </div>
  {/* Fecha de nacimiento y edad */}
  <div className="flex items-center gap-3">
    <CalendarDays className="h-5 w-5 text-muted-foreground" />
    <div>
      <p className="text-sm font-medium">Fecha de nacimiento</p>
      <p className="text-sm text-muted-foreground">{formatearFecha(profile.Fecha_nacimiento)} ({calcularEdad(profile.Fecha_nacimiento)})</p>
    </div>
  </div>
  {/* Tipo de cuenta */}
  <div className="flex items-center gap-3">
    <Shield className="h-5 w-5 text-muted-foreground" />
    <div>
      <p className="text-sm font-medium">Tipo de cuenta</p>
      <p className="text-sm text-muted-foreground capitalize">{profile.Tipo}</p>
    </div>
  </div>
  {/* Última modificación */}
  <div className="flex items-center gap-3">
    <Clock className="h-5 w-5 text-muted-foreground" />
    <div>
      <p className="text-sm font-medium">Última modificación del perfil</p>
      <p className="text-sm text-muted-foreground">{updated_at}</p>
    </div>
  </div>
</div>
</CardContent>
</Card>

```

Ilustración 194 Perfil jsx VII

```

<CardFooter>
  {/* Botón para abrir el modal de edición */}
  <Button variant="outline" className="w-full" onClick={handleEditProfile}>
    Editar perfil
  </Button>
</CardFooter>
</Card>
</div>

/* Sección de actividad reciente (reservas) */
<div className="md:col-span-2">
  <Tabs defaultValue="actividad" className="w-full">
    <TabsList className="grid grid-cols-1 mb-6">
      <TabsTrigger value="actividad">Actividad Reciente</TabsTrigger>
    </TabsList>

    <TabsContent value="actividad">
      <Card>
        <CardHeader>
          <CardTitle>Actividad Reciente</CardTitle>
          <CardDescription>Historial de tus últimas actividades</CardDescription>
        </CardHeader>
        <CardContent>
          <div className="space-y-6">
            {/* Lista de reservas del usuario */}
            {reservas.map((item) => {
              const publicacion = item.publicacion;
              const imagenes = publicacion?.imagen?.split(";") || [];

              return (
                <div key={item.id} className="flex items-start gap-4 pb-4 border-b last:border-b-0">
                  <div className="rounded-md overflow-hidden w-16 h-16 flex-shrink-0">
                    <img
                      src={imagenes[0] ? IMAGE_URL + imagenes[0] : "/placeholder.svg"}
                      alt={publicacion?.titulo || "Sin título"}
                      width={100}
                      height={100}
                      className="object-cover w-full h-full"

```

Ilustración 195 Perfil jsx VIII

```
</div>
<div className="flex-1">
  <h3 className="font-medium">Reserva: {publicacion?.titulo}</h3>
  <p className="text-sm text-muted-foreground">
    Total pagado: {item.total_pagar} € | Personas: {item.personas}
  </p>
  <div className="flex items-center gap-2 mt-1">
    <CalendarDays className="h-3 w-3 text-muted-foreground" />
    <span className="text-xs text-muted-foreground">
      Fecha: {new Date(item.fecha_reserva).toLocaleDateString()!}
    </span>
  </div>
</div>
);
</div>
</CardContent>
</Card>
</TabsContent>
</div>
</div>
/* Modal de edición de usuario */
{isModalOpen && (
  < UsuariosModal
    isOpen={isModalOpen}
    setIsopen={setIsModalopen}
    user={profile}
    isProfileEdit={true} // Indicamos que es una edición desde el perfil
  />
)
};
```

Ilustración 196 Perfil jsx IX

Mostrador.jsx

El mostrador es el encargado de mostrar los datos de la publicación además de su opción de compra y comprobaciones necesarias.

Este usa los componentes person-chooser.jsx y time-slider.jsx que será explicado en los siguientes apartados.

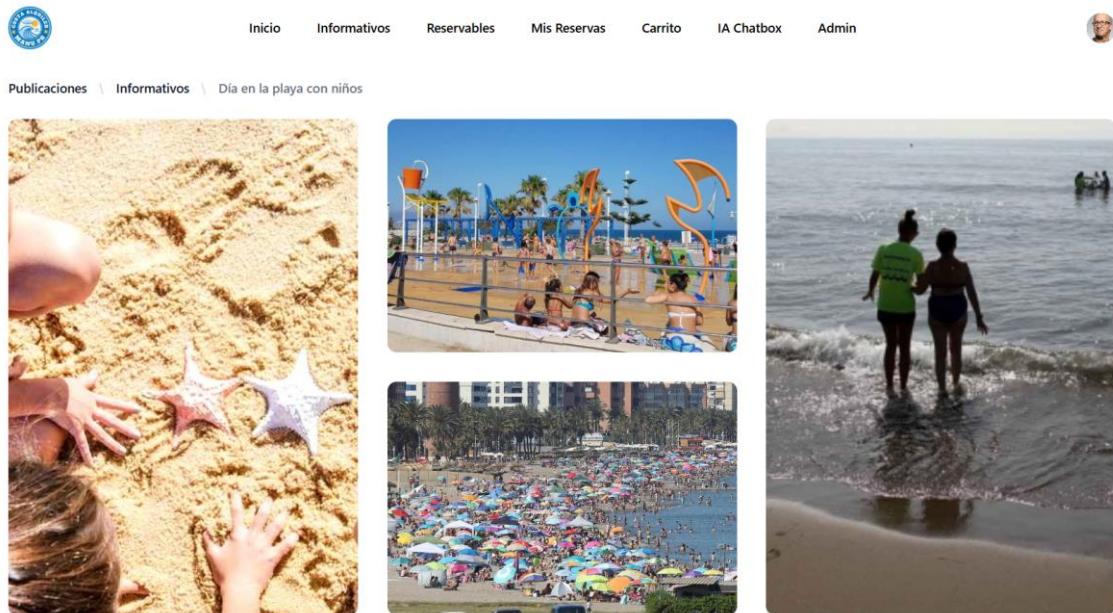


Ilustración 197 Mostrador Informativos I

Día en la playa con niños

Consejos para disfrutar un día en familia junto al mar

1 de julio de 2025

Este producto no está disponible para alquiler
Este artículo es de tipo informativo

Highlights

- ☀️ Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🌐 Portal de compra 100% seguro
- 🚫 Cancela cuando Quieras

Detalles

Disfruta de tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

Ilustración 198 Mostrador Informativos II

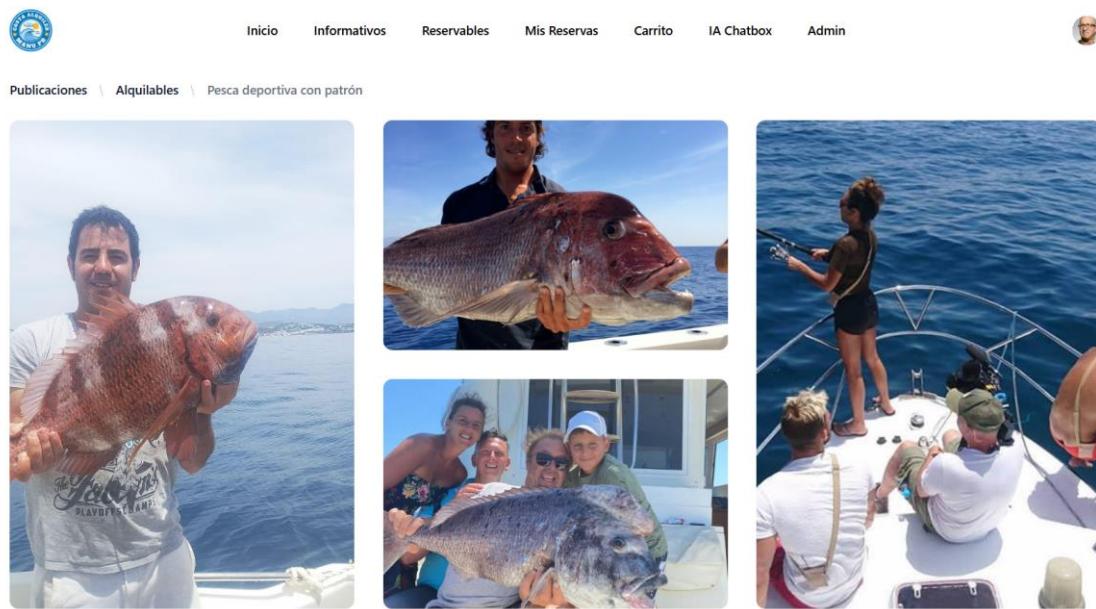


Ilustración 199 Mostrador Alquilables I

Pesca deportiva con patrón

Excursión de pesca para grupos

4 de julio de 2025

Highlights

- 🌟 Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🛡️ Portal de compra 100% seguro
- 🔴 Cancela cuando Quieras

Detalles

Disfruta de tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

\$90

Momento del Día

15:00 PM

08:00 15:00 22:00

Personas



[Añadir al Carrito](#)

Ilustración 200 Mostrador Alquilables II

Alquiler equipo de pesca

Caña y material por día completo

6 de julio de 2025

Highlights

- 🌟 Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🛡️ Portal de compra 100% seguro
- 🔴 Cancela cuando Quieras

Detalles

Disfruta de tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

\$15
Momento del Día

20:00 PM

No Disponible

08:00 15:00 22:00

Personas



Añadir al Carrito

Ilustración 201 Mostrador Alquilables III

Paseo en helicóptero

10 min de recorrido costero (premium)

3 de julio de 2025

Highlights

- 🌟 Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🛡️ Portal de compra 100% seguro
- 🔴 Cancela cuando Quieras

Detalles

Disfruta de tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

\$250
Momento del Día

15:00 PM

No Disponible

08:00 15:00 22:00

Personas



Añadir al Carrito

Ilustración 202 Mostrador Alquilables IV

Código – Mostrador

```
/*
 * @file Mostrador.jsx
 * @description Página de detalle de una publicación seleccionada. Permite ver información, imágenes, seleccionar hora y personas, y añadir al carrito.
 */

import { useState, useEffect } from "react"
import { useLocation, useNavigate } from "react-router-dom"
import { API_URL, IMAGE_URL } from "../utilities/apiRest"
import Timeslider from "../components/time-slider"
import SelectorPersonas from "../components/person-chooser"
import axios from "axios"

/**
 * Componente de detalle de publicación (Mostrador).
 * Muestra información, imágenes, permite seleccionar hora/personas y añadir al carrito.
 * @component
 * @param {object} props
 * @param {Array} props.cart - Carrito de compras global.
 * @param {Function} props.setCart - Setter para actualizar el carrito.
 * @returns {JSX.Element}
 */
export default function Mostrador({ cart, setCart }) {
  /**
   * Hook de React Router para obtener el estado de navegación.
   */
  const location = useLocation()
  /**
   * Hook de navegación para redireccionar.
   */
  const navigate = useNavigate()
  /**
   * Hora seleccionada para la reserva.
   * {[number, Function]}
   */
  const [hora, setHora] = useState(15)
  /**
   * Indica si la hora seleccionada está disponible.
   * {[boolean, Function]}
   */
  const [disponibilidadHora, setDisponibilidadHora] = useState(true)
```

Ilustración 203 Mostrador.jsx I

```
/**
 * Número de personas seleccionadas.
 * {[number, Function]}
 */
const [personas, setPersonas] = useState(1)
/**
 * Número máximo de personas disponibles para reservar.
 * {[number, Function]}
 */
const [personasDisponibles, setPersonasDisponibles] = useState(4)
/**
 * Estado de carga de la página.
 * {[boolean, Function]}
 */
const [loading, setLoading] = useState(true)
/**
 * Indica si el producto ya fue añadido al carrito.
 * {[boolean, Function]}
 */
const [añadido, setAñadido] = useState(false)
/**
 * Precio total calculado según personas.
 * {[number, Function]}
 */
const [precioTotal, setPrecioTotal] = useState()
/**
 * Indica si la hora seleccionada ya ha pasado.
 * {[boolean, Function]}
 */
const [horaPasada, setHoraPasada] = useState(false)

// Si no hay elemento en el estado de navegación, no renderiza nada
if (!location.state || !location.state.elemento) return null

// Obtiene el elemento seleccionado desde la navegación
```

Ilustración 204 Mostrador.jsx II

```

    /**
     * Efecto para mostrar en consola el número de personas seleccionadas.
     */
    useEffect(() => {
      console.log("Personas seleccionadas: ", personas)
    }, [personas])

    /**
     * Efecto para comprobar si el producto ya está en el carrito.
     */
    useEffect(() => {
      const existeEnCarrito = cart.some((item) => item.id === elemento.id)
      setAñadido(existeEnCarrito)
    }, [cart, elemento, hora])

    /**
     * Efecto para obtener la capacidad máxima disponible para la publicación.
     */
    useEffect(() => {
      const url = API_URL + "api/capacidadDisponible"
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}
      axios
        .post(url, { idPublicacion: elemento.id }, { headers })
        .then((response) => {
          setPersonasDisponibles(response.data.max_reservables)
        })
        .catch((error) => {
          console.error("Error al cargar la capacidad disponible:", error)
        })
        .finally(() => {
          setLoading(false)
        })
    }, [personasDisponibles])
  
```

Ilustración 205 Mostrador.jsx III

```

    /**
     * Efecto para actualizar el precio total al cambiar el número de personas.
     */
    useEffect(() => {
      setPrecioTotal(elemento.precio * personas)
    }, [personas])

    /**
     * Efecto para comprobar si la hora seleccionada ya ha pasado según la fecha/hora del servidor.
     */
    useEffect(() => {
      axios
        .get(API_URL + "api/horaFecha")
        .then((response) => {
          if (response.data.fecha == elemento.fecha_evento.split(" ")[0]) {
            const horaEntera = Number.parseInt(response.data.hora.split(":")[0], 10)
            setHoraPasada(!horaEntera < hora))
          } else setHoraPasada(false)
        })
        .catch((error) => {
          console.error("Error al comprobar hora pasada:", error)
        })
    }, [disponibilidadHora, hora])
  
```

Ilustración 206 Mostrador.jsx IV

```

    /**
     * Efecto para comprobar la disponibilidad de la hora seleccionada.
     */
    useEffect(() => {
      const url = API_URL + "api/disponibilidadReserva"
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}
      axios
        .post(url, { idPublicacion: elemento.id, horaReserva: hora + ":00" }, { headers })
        .then((response) => {
          setDisponibilidadHora(response.data.disponible)
        })
        .catch((error) => {
          console.error("Error al comprobar disponibilidad de hora:", error)
        })
    }, [hora, disponibilidadHora])

    /**
     * Efecto para redirigir si no hay elemento en el estado de navegación.
     */
    useEffect(() => {
      if (!location.state || !location.state.elemento) navigate("/notFound")
    }, [location.state, navigate])
  
```

Ilustración 207 Mostrador.jsx V

```

    /**
     * Maneja el evento de añadir el producto al carrito.
     * @param {React.FormEvent} e
     */
    const handleAddToCart = (e) => {
      e.preventDefault()
      if (disponibilidadHora) {
        const nuevoElemento = {
          ...elemento,
          horaReserva: hora,
          precio: precioTotal,
          personas: Number.parseInt(personas),
        }
        setCart((prev) => [...prev, nuevoElemento])
        setAñadido(true)
      }
    }

    /**
     * Formatea una fecha a formato legible en español.
     * @param {string} dateString
     * @returns {string}
     */
    function formatDate(dateString) {
      const date = new Date(dateString)
      const options = { year: "numeric", month: "long", day: "numeric" }
      return date.toLocaleDateString("es-ES", options)
    }

    // Array de imágenes del elemento
    const imagenes = elemento.imagen.split(";")
  
```

Ilustración 208 Mostrador.jsx VI

```

return (
  <div className="bg-white">
    <div className="pt-6">
      /* Breadcrumb de navegación */
      <nav aria-label="Breadcrumb">
        <ol role="list" className="mx-auto flex max-w-2xl items-center space-x-2 px-4 sm:px-6 lg:max-w-7xl lg:px-8">
          <li>
            <div className="flex items-center">
              <p className="mr-2 text-sm font-medium text-gray-900">
                | Publicaciones
              </p>
              <svg
                width="16"
                height="20"
                viewBox="0 0 16 20"
                fill="currentColor"
                aria-hidden="true"
                className="h-5 w-4 text-gray-300"
              >
                <path d="M5.697 4.34L8.98 16.532h1.327L7.025 4.341H5.697z" />
              </svg>
            </div>
          </li>
          <li>
            <div className="flex items-center">
              <a href={"/" + elemento.tipo + "s"} className="mr-2 text-sm font-medium text-gray-900 capitalize">
                | {elemento.tipo + "s"}
              </a>
              <svg
                width="16"
                height="20"
                viewBox="0 0 16 20"
                fill="currentColor"
                aria-hidden="true"
                className="h-5 w-4 text-gray-300"
              >
                <path d="M5.697 4.34L8.98 16.532h1.327L7.025 4.341H5.697z" />
              </svg>
            </div>
          </li>
        </ol>
      </nav>
    </div>
  </div>
)

```

Ilustración 209 Mostrador.jsx VII

```

<li className="text-sm">
  <p aria-current="page" className="font-medium text-gray-500 hover:text-gray-600">
    | {elemento.titulo}
  </p>
</li>
</ol>
</nav>

/* Galería de imágenes del producto */
<div className="mx-auto mt-6 max-w-2xl sm:px-6 lg:grid lg:max-w-7xl lg:grid-cols-3 lg:gap-x-8 lg:px-8">
  <img
    src={IMAGE_URL + imagenes[0] || "/placeholder.svg"}
    alt="Imagen principal"
    className="hidden size-full rounded-lg object-cover lg:block"
  />
  <div className="hidden lg:grid lg:grid-cols-1 lg:gap-y-8">
    <img
      src={IMAGE_URL + imagenes[1] || "/placeholder.svg"}
      alt="Imagen secundaria 1"
      className="aspect-3/2 w-full rounded-lg object-cover"
    />
    <img
      src={IMAGE_URL + imagenes[2] || "/placeholder.svg"}
      alt="Imagen secundaria 2"
      className="aspect-3/2 w-full rounded-lg object-cover"
    />
  </div>
  <img
    src={IMAGE_URL + imagenes[3] || "/placeholder.svg"}
    alt="Imagen secundaria 3"
    className="aspect-4/5 size-full object-cover sm:rounded-lg lg:aspect-auto"
  />
</div>

/* Información y formulario de reserva */
<div className="mx-auto max-w-2xl px-4 pt-10 pb-16 sm:px-6 lg:grid lg:max-w-7xl lg:grid-cols-3 lg:grid-rows-[auto_auto_ifr] lg:gap-x-8">
  <div className="lg:col-span-2 lg:border-r lg:border-gray-200 lg:pr-8">
    <h1 className="text-2xl font-bold tracking-tight text-gray-900 sm:text-3xl">{elemento.titulo}</h1>
  </div>

```

Ilustración 210 Mostador.jsx VII

```

/* Loader de carga */
loading ? (
  <div className="flex justify-center items-center">
    | <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
  </div>
) : (
  <div className="mt-4 lg:row-span-3 lg:mt-0">
    /* Si el producto es reservable muestra el formulario, si es informativo muestra mensaje */
    {elemento.tipo != "informativo" ? (
      <form className="mt-5" onSubmit={handleAddToCart}>
        <h2 className="sr-only">Product information</h2>
        <p className="text-3xl tracking-tight text-gray-900">${precioTotal}</p>
        <div>
          <h3 className="text-sm font-medium text-gray-900">Momento del Día</h3>
          /* Slider de selección de hora */
          <TimeSlider
            id={elemento.id}
            setHora={setHora}
            hora={hora}
            disponible={disponibilidadHora}
            horaPasada={horaPasada}
          />
        </div>
        <div className="mt-10">
          <div className="flex items-center justify-between">
            | <h3 className="text-sm font-medium text-gray-900">Personas</h3>
            </div>
            /* Selector de personas */
            <SelectorPersonas personasDisponibles={personasDisponibles} setPersonas={setPersonas} selected={1} />
          </div>
        <div style={{marginTop: 10}}>
          /* Mensaje de añadido al carrito */
          {añadido && <p className="mt-4 text-green-600 font-medium text-center">Añadido al carrito</p>}
        </div>
        /* Botón de añadir al carrito, solo si hay disponibilidad y la hora no ha pasado */
        {disponibilidadHora && personasDisponibles > 0 && !añadido && !horaPasada ? (
          <button
            type="submit"
            className="mt-10 flex w-full items-center justify-center rounded-md border border-transparent bg-indigo-600 px-8 py-3 text-white transition duration-150 ease-in-out hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-indigo-500 uppercase"
            onClick={handleAddToCart}
          >Añadir al Carrito</button>
        ) : (
          <button
            type="submit"
            disabled
            className="mt-10 flex w-full items-center justify-center rounded-md border border-transparent bg-indigo-300 px-8 py-3 text-white transition duration-150 ease-in-out hover:bg-indigo-400 focus:outline-none focus:ring-2 focus:ring-indigo-500 uppercase"
            onClick={handleAddToCart}
          >Añadir al Carrito</button>
        )}
      </form>
    ) : (
      // Si es informativo, muestra mensaje
      <div className="mt-5 p-4 bg-gray-50 rounded-lg border border-gray-200">
        <p className="text-gray-700 font-medium">Este producto no está disponible para alquiler</p>
        <p className="text-gray-500 text-sm mt-2">Este artículo es de tipo informativo</p>
      </div>
    )
  )
)

/* Descripción y detalles del producto */
<div className="py-10 lg:col-span-2 lg:col-start-1 lg:border-r border-gray-200 lg:pt-6 lg:pr-8 lg:pb-16">
  <div>
    <h3 className="sr-only">Description</h3>
    <div className="space-y-6">
      <p className="text-base text-gray-900">{elemento.descripcion}</p>
    </div>
    <div className="space-y-6 mt-4">
      <p className="text-base text-gray-500">{formatDate(elemento.fecha_evento)}</p>
    </div>
  </div>
  <div style={{marginTop: 10}}>
    /* Lista de highlights */
    <div className="mt-10">
      <h3 className="text-sm font-medium text-gray-900">Highlights</h3>
      <div className="mt-4">
        <ul role="list" className="list-disc space-y-2 pl-4 text-sm">

```

Ilustración 211 Mostrador.jsx VIII

```

>
      | Añadir al Carrito
    </button>
  ) : (
    <button
      type="submit"
      disabled
      className="mt-10 flex w-full items-center justify-center rounded-md border border-transparent bg-indigo-300 px-8 py-3 text-white transition duration-150 ease-in-out hover:bg-indigo-400 focus:outline-none focus:ring-2 focus:ring-indigo-500 uppercase"
      onClick={handleAddToCart}
    >Añadir al Carrito</button>
  )}
</form>
) : (
  // Si es informativo, muestra mensaje
  <div className="mt-5 p-4 bg-gray-50 rounded-lg border border-gray-200">
    <p className="text-gray-700 font-medium">Este producto no está disponible para alquiler</p>
    <p className="text-gray-500 text-sm mt-2">Este artículo es de tipo informativo</p>
  </div>
)
)

/* Descripción y detalles del producto */
<div className="py-10 lg:col-span-2 lg:col-start-1 lg:border-r border-gray-200 lg:pt-6 lg:pr-8 lg:pb-16">
  <div>
    <h3 className="sr-only">Description</h3>
    <div className="space-y-6">
      <p className="text-base text-gray-900">{elemento.descripcion}</p>
    </div>
    <div className="space-y-6 mt-4">
      <p className="text-base text-gray-500">{formatDate(elemento.fecha_evento)}</p>
    </div>
  </div>
  <div style={{marginTop: 10}}>
    /* Lista de highlights */
    <div className="mt-10">
      <h3 className="text-sm font-medium text-gray-900">Highlights</h3>
      <div className="mt-4">
        <ul role="list" className="list-disc space-y-2 pl-4 text-sm">
```

Ilustración 212 Mostrador.jsx IX

Ilustración 213 Mostrador jsx X

COMPONENTES

buscador.jsx

Componente creado y usado para hacer búsqueda de datos para las publicaciones



Ilustración 214 Buscador

Código – Buscador

```
/*
 * @file buscador.jsx
 * @description Componente de buscador con filtro de fecha para publicaciones. Permite buscar por texto y filtrar por rango de fechas.
 */
import { useState } from "react"
import { Search, Calendar, X } from "lucide-react"

/**
 * Componente Buscador.
 *
 * @param {Object} props
 * @param {function} props.onSearch - Callback para búsqueda por texto.
 * @param {function} props.onFilter - Callback para filtrar por fecha.
 * @param {function} props.onReset - Callback para limpiar filtros y búsqueda.
 * @returns {JSX.Element} El componente de buscador con filtros.
 */
export default function Buscador({ onSearch, onFilter, onReset }) {
  const [searchTerm, setSearchTerm] = useState("")
  const [filterDate, setFilterDate] = useState("")
  const [isFilterOpen, setIsFilterOpen] = useState(false)

  /**
   * Maneja el cambio en el input de búsqueda.
   * @param {React.ChangeEvent<HTMLInputElement>} e
   */
  const handleSearchChange = (e) => {
    const value = e.target.value
    setSearchTerm(value)
    onSearch(value)
  }

  /**
   * Maneja el cambio en el filtro de fecha.
   * @param {React.ChangeEvent<HTMLInputElement>} e
   */
}
```

Ilustración 215 Buscador jsx I

```
/**
 * Resetea los filtros y la búsqueda.
 */
const handleReset = () => {
  setSearchTerm("")
  setFilterDate("")
  onReset()
}

return (
  <div className="w-full mb-8 transition-all duration-300">
    <div className="flex items-center gap-3">
      {/* Buscador minimalista */}
      <div className="relative flex-grow">
        <div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none">
          <Search className="w-4 h-4 text-gray-400" />
        </div>
        <input
          type="text"
          className="w-full py-2 pl-10 pr-10 text-sm bg-white border-b border-gray-200 focus:border-gray-400 focus:outline-none transition"
          placeholder="Buscar publicaciones..."
          value={searchTerm}
          onChange={handleSearchChange}
        />
        {searchTerm && (
          <button
            className="absolute inset-y-0 right-0 flex items-center pr-3 text-gray-400 hover:text-gray-600 transition-colors"
            onClick={() => {
              setSearchTerm("")
              onSearch("")
            }}
          >
            <X className="w-4 h-4" />
          </button>
        )}
    </div>
  </div>
)
```

Ilustración 216 Buscador jsx II

```
/* Filtro de fecha minimalista */


<button
    onClick={() => setIsFilterOpen(!isFilterOpen)}
    className={`p-2 rounded-full transition-all duration-200 ${filterDate ? "bg-gray-100 text-gray-800" : "text-gray-400 hover:bg-gray-50"}`}
  >
    title="Filtrar por fecha"
  </button>

  {isFilterOpen && (
    <div className="absolute right-0 mt-2 w-48 bg-white shadow-lg rounded-md py-1 z-10 border border-gray-100">
      <div className="py-1">
        <button
          className="w-full text-left px-4 py-2 text-sm ${filterDate === "" ? "bg-gray-50 text-gray-800" : "text-gray-600 hover:bg-gray-50"}`}
        >
          onClick={() => {
            setFilterDate("")
            onFilter("")
            setIsFilterOpen(false)
          }}
        >
          Todas las fechas
        </button>
        <button
          className="w-full text-left px-4 py-2 text-sm ${filterDate === "today" ? "bg-gray-50 text-gray-800" : "text-gray-600 hover:bg-gray-50"}`}
        >
          onClick={() => {
            setFilterDate("today")
            onFilter("today")
            setIsFilterOpen(false)
          }}
        >
          Hoy
        </button>
    </div>
  )}


```

Ilustración 217 Buscador jsx III

```
<button
  className="w-full text-left px-4 py-2 text-sm ${filterDate === "week" ? "bg-gray-50 text-gray-800" : "text-gray-600 hover:bg-gray-50"}`}
  onClick={() => {
    setFilterDate("week")
    onFilter("week")
    setIsFilterOpen(false)
  }}
>
  Esta semana
</button>
<button
  className="w-full text-left px-4 py-2 text-sm ${filterDate === "month" ? "bg-gray-50 text-gray-800" : "text-gray-600 hover:bg-gray-50"}`}
  onClick={() => {
    setFilterDate("month")
    onFilter("month")
    setIsFilterOpen(false)
  }}
>
  Este mes
</button>
<button
  className="w-full text-left px-4 py-2 text-sm ${filterDate === "year" ? "bg-gray-50 text-gray-800" : "text-gray-600 hover:bg-gray-50"}`}
  onClick={() => {
    setFilterDate("year")
    onFilter("year")
    setIsFilterOpen(false)
  }}
>
  Este año
</button>
</div>
</div>
)}
```

Ilustración 218 Buscador jsx IV

```

    </div>

    /* Botón de reset minimalista, solo visible cuando hay filtros activos */
    {searchTerm || filterDate) && (
      <button className="text-sm text-gray-500 hover:text-gray-700 transition-colors" onClick={handleReset}>
        Limpiar
      </button>
    )
  </div>

  /* Indicador de filtros activos */
  {filterDate && (
    <div className="mt-3 flex items-center">
      <span className="text-xs text-gray-500 mr-2">Filtrado por:</span>
      <span className="inline-flex items-center px-2 py-1 rounded-full text-xs font-medium bg-gray-100 text-gray-800">
        {filterDate === "today"
          ? "Hoy"
          : filterDate === "week"
            ? "Esta semana"
            : filterDate === "month"
              ? "Este mes"
              : "Este año"}
        <button
          className="ml-1 text-gray-500 hover:text-gray-700"
          onClick={() => {
            setFilterdate("")
            onFilter("")}}
        >
          <x className="w-3 h-3" />
        </button>
      </span>
    </div>
  )
}
}

```

Ilustración 219 Buscador jsx V

Calendar.jsx

Es el calendario en sí el cual hace las peticiones al servidor y según los datos que retorna muestra x dia dicha reserva.

Este va de la mano con days-detail-modal.jsx que se encarga de mostrar las reservas mediante un modal además de la posibilidad de cancelar dicha reserva.

Julio De 2025							<	Hoy	>
Mon	Tue	Wed	Thu	Fri	Sat	Sun			
30	1 Tour en barco por la bahía	2 Moto de agua 30 min	3 Paseo en helicóptero Paseo en velero privado	4	5	6 Alquiler equipo de pesca			
7	8 Parasailing	9	10	11	12	13			
14	15	16	17	18	19	20			

Ilustración 220 Calendario

Código – Calendario

```

/*
 * @file Calendar.jsx
 * @description Componente de calendario que muestra las reservas del usuario y permite ver detalles diarios y cancelar reservas.
 * @module components/Calendar
 */

import { useState, useEffect } from "react"
import { ChevronLeft, ChevronRight } from "lucide-react"
import { Button } from "@/components/ui/button"
import axios from "axios"
import { API_URL } from "../utilities/apiRest"
import DayDetailsModal from "./day-details-modal"

/**
 * Componente Calendar.
 * Muestra un calendario mensual con las reservas del usuario, permite navegar entre meses,
 * ver detalles de un día y cancelar reservas.
 *
 * @function
 * @returns {JSX.Element} El calendario de reservas.
 */
export default function Calendar() {
    // Estado para el mes actual mostrado
    const [currentMonth, setCurrentMonth] = useState(new Date())
    // Estado de carga de datos
    const [loading, setLoading] = useState(true)
    // Lista de eventos/reservas
    const [events, setEvents] = useState([])
    // Día seleccionado para mostrar detalles
    const [selectedDay, setSelectedDay] = useState(null)
    // Estado para mostrar/ocultar el modal de detalles
    const [isModalOpen, setIsModalOpen] = useState(false)
}


```

Ilustración 221 Calendario jsx I

```

// Carga las reservas del usuario al montar el componente
useEffect(() => {
    const url = API_URL + "api/obtenerReservasUsuario"
    const token = localStorage.getItem("authToken")
    const headers = token ? { Authorization: `Bearer ${token}` } : {}

    axios
        .get(url, { headers })
        .then((response) => {
            // Colores para los eventos
            const colores = [
                "bg-red-200",
                "bg-green-200",
                "bg-yellow-200",
                "bg-purple-200",
                "bg-blue-200",
                "bg-pink-200",
            ]
            // Transforma las reservas en eventos para el calendario
            const eventosTransformados = response.data.map((reserva, index) => ({
                id: reserva.id.toString(),
                title: reserva.publicacion.titulo,
                date: new Date(reserva.fecha_reserva),
                color: colores[index % colores.length],
            }))
            setEvents(eventosTransformados)

            // Si hay eventos, muestra el mes del primer evento
            if (eventosTransformados.length > 0) {
                setCurrentMonth(eventosTransformados[0].date)
            }
        })
        .catch((error) => {
            console.error("Error al cargar los eventos:", error)
        })
        .finally(() => {
            setLoading(false)
        })
    }, [])


```

Ilustración 222 Calendario jsx II

```
// Días de la semana
const daysOfWeek = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

/**
 * Genera la estructura de semanas y días para el mes actual.
 * @param {Date} date
 * @returns {Array} Array de semanas, cada una con 7 días.
 */
const getMonthData = (date) => {
  const year = date.getFullYear()
  const month = date.getMonth()
  const firstDay = new Date(year, month, 1)
  const lastDay = new Date(year, month + 1, 0)
  let firstDayOfWeek = firstDay.getDay() - 1
  if (firstDayOfWeek === -1) firstDayOfWeek = 6

  const daysFromPrevMonth = firstDayOfWeek
  const totalDays = 42
  const calendarDays = []

  const prevMonth = new Date(year, month - 1)
  const prevMonthLastDay = new Date(year, month, 0).getDate()

  // Días del mes anterior
  for (let i = prevMonthLastDay - daysFromPrevMonth + 1; i <= prevMonthLastDay; i++) {
    calendarDays.push({
      date: new Date(prevMonth.getFullYear(), prevMonth.getMonth(), i),
      isCurrentMonth: false,
    })
  }

  // Días del mes actual
  for (let i = 1; i <= lastDay.getDate(); i++) {
    calendarDays.push({
      date: new Date(year, month, i),
      isCurrentMonth: true,
    })
  }
}
```

Ilustración 223 Calendario jsx III

```

// Días del mes siguiente
const nextMonth = new Date(year, month + 1)
const remainingDays = totalDays - calendarDays.length

for (let i = 1; i <= remainingDays; i++) {
  calendarDays.push({
    date: new Date(nextMonth.getFullYear(), nextMonth.getMonth(), i),
    isCurrentMonth: false,
  })
}

// Agrupa los días en semanas
const weeks = []
for (let i = 0; i < calendarDays.length; i += 7) {
  weeks.push(calendarDays.slice(i, i + 7))
}

return weeks
}

// Semanas del mes actual
const weeks = getMonthData(currentMonth)

/**
 * Formatea el mes y año para mostrar en el encabezado.
 * @param {Date} date
 * @returns {string}
 */
const formatMonth = (date) => {
  return date.toLocaleDateString("es-ES", { month: "long", year: "numeric" })
}

// Navega al mes anterior
const handlePrevMonth = () => {
  setCurrentMonth(new Date(currentMonth.getFullYear(), currentMonth.getMonth() - 1))
}

```

Ilustración 224 Calendario jsx IV

```

// Navega al mes siguiente
const handleNextMonth = () => {
  setCurrentMonth(new Date(currentMonth.getFullYear(), currentMonth.getMonth() + 1))
}

// Vuelve al mes actual
const handleToday = () => {
  setCurrentMonth(new Date())
}

/**
 * Cancela una reserva (elimina el evento del calendario).
 * @param {string} eventId
 */
const handleCancelReserva = (eventId) => {
  const updatedEvents = events.filter((event) => event.id !== eventId)
  setEvents(updatedEvents)

  if (selectedDay) {
    const updatedDayEvents = selectedDay.events.filter((event) => event.id !== eventId)
    setSelectedDay({ ...selectedDay, events: updatedDayEvents })
  }
}

/**
 * Obtiene los eventos para una fecha específica.
 * @param {Date} date
 * @returns {Array}
 */
const getEventsForDate = (date) => {
  return events.filter(
    (event) =>
      event.date.getDate() === date.getDate() &&
      event.date.getMonth() === date.getMonth() &&
      event.date.getFullYear() === date.getFullYear(),
  )
}

```

Ilustración 225 Calendario jsx V

```

    /**
     * Determina si una fecha es hoy.
     * @param {Date} date
     * @returns {boolean}
     */
    const isToday = (date) => {
        const today = new Date()
        return (
            date.getDate() === today.getDate() &&
            date.getMonth() === today.getMonth() &&
            date.getFullYear() === today.getFullYear()
        )
    }

    /**
     * Maneja el click en un día del calendario, abre el modal de detalles.
     * @param {Object} day
     * @param {Array} events
     */
    const handleDayClick = (day, events) => {
        setSelectedDay({
            date: day.date,
            events: events,
        })
        setIsModalOpen(true)
    }
}

```

Ilustración 226 Calendario.jsx VI

```

return (
    <>
    {loading ? (
        <div className="flex justify-center items-center">
            <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
        </div>
    ) : (
        <div className="w-full mt-5 mb-15 mx-auto bg-white rounded-lg shadow-sm border">
            <div className="flex justify-between items-center p-4 border-b">
                <h2 className="text-xl font-semibold capitalize">{formatMonth(currentMonth)}</h2>
                <div className="flex items-center gap-2">
                    <div className="flex border rounded-lg overflow-hidden">
                        <Button variant="ghost" size="sm" className="rounded-none border-r h-10" onClick={handlePrevMonth}>
                            <ChevronLeft className="h-4 w-4" />
                        </Button>
                        <Button variant="ghost" size="sm" className="rounded-none h-10" onClick={handleToday}>
                            Hoy
                        </Button>
                        <Button variant="ghost" size="sm" className="rounded-none border-l h-10" onClick={handleNextMonth}>
                            <ChevronRight className="h-4 w-4" />
                        </Button>
                    </div>
                </div>
            </div>
            <div className="grid grid-cols-7 border-b">
                {daysOfWeek.map((day) => (
                    <div key={day} className="py-2 text-center font-medium text-sm">
                        {day}
                    </div>
                ))}
            </div>
            <div className="grid grid-cols-7 grid-rows-6 h-[calc(100vh-12rem)] min-h-[600px] scroll-y-no-scrollbar overflow-y-auto">
                {weeks.flat().map((day, index) => {
                    const dayEvents = getEventsForDate(day.date)
                    <div key={index} className="flex flex-col justify-between items-start gap-2">
                        <div>
                            {dayEvents.map((event) => (
                                <div>
                                    {event}
                                </div>
                            ))}
                        </div>
                    </div>
                ))}
            </div>
        </div>
    )
)
}

```

Ilustración 227 Calendario.jsx VII

```

return (
  <div
    key={index}
    className={`border-b border-r p-1 relative
      ${!day.isCurrentMonth ? "text-gray-400" : ""}
      ${isToday(day.date) ? "bg-blue-100" : ""}
      cursor-pointer hover:bg-gray-50`}
    onClick={() => handleDayClick(day, dayEvents)}
  >
    <div className="flex flex-col h-full">
      <div className="flex items-start justify-between">
        <span className="p-1 font-medium text-sm">{day.getDate()}

```

Ilustración 228 Calendario jsx VIII

Days-details-modal.jsx

Es el modal que se despliega cuando clicamos en un dia para hacer el cancelamiento de la reserva en el apartado de mis reservas



Ilustración 229 Days-details-modal

Código – days-details-modal

```
/*
 * @file day-details-modal.jsx
 * @description Modal que muestra los detalles de los eventos/reservas de un día concreto y permite cancelar reservas.
 * @module components/day-details-modal
 */

import { useState } from "react"
import { Calendar, Clock, AlertTriangle } from "lucide-react"
import { Dialog,DialogContent,DialogTitle } from "@/components/ui/dialog"
import { ScrollArea } from "@/components/ui/scroll-area"
import { Badge } from "@/components/ui/badge"
import { Button } from "@/components/ui/button"
import axios from "axios"
import { API_URL } from "../utilities/apiRest"

/**
 * Modal de detalles de un día con eventos/reservas.
 *
 * @param {Object} props
 * @param {boolean} props.isOpen - Si el modal está abierto.
 * @param {function} props.onClose - Función para cerrar el modal.
 * @param {Object} props.day - Objeto con la fecha y los eventos de ese día.
 * @param {function} props.onCancel - Callback al cancelar una reserva.
 * @returns {JSX.Element|null} El modal de detalles del día.
 */
export default function DayDetailsModal({ isOpen, onClose, day, onCancel }) {
  const [confirmDialogOpen, setConfirmDialogOpen] = useState(false)
  const [selectedEventId, setSelectedEventId] = useState(null)
  const [isLoading, setIsLoading] = useState(false)

  if (!day) return null

  /**
   * Formatea la fecha a un string legible en español.
   * @param {Date} date
   * @returns {string}
   */
  const formatDate = (date) => {
    return date.toLocaleDateString("es-ES", {
      weekday: "long",
      year: "numeric",
      month: "long",
      day: "numeric",
    })
  }

  /**
   * Formatea la hora a un string legible en español.
   * @param {Date} date
   * @returns {string}
   */
  const formatTime = (date) => {
    return date.toTimeString("es-ES", {
      hour: "2-digit",
      minute: "2-digit",
    })
  }

  /**
   * Cancela la reserva llamando a la API y ejecuta el callback onCancel.
   * @param {string} eventId
   */
  const handleCancelarReserva = async (eventId) => {
    console.log(`Cancelando reserva con ID: ${eventId}`)
    try {
      setIsLoading(true)
      const url = API_URL + `/api/reservas/${eventId}`
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}
      await axios.delete(url, { headers })
      if (onCancel) onCancel(eventId)
      setConfirmDialogOpen(false)
    } catch (error) {
      console.error(`Error al cancelar la reserva: ${error}`)
    } finally {
  
```

Ilustración 230 days-details-modal I

```
const formatDate = (date) => {
  return date.toLocaleDateString("es-ES", {
    weekday: "long",
    year: "numeric",
    month: "long",
    day: "numeric",
  })
}

/**
 * Formatea la hora a un string legible en español.
 * @param {Date} date
 * @returns {string}
 */
const formatTime = (date) => {
  return date.toTimeString("es-ES", {
    hour: "2-digit",
    minute: "2-digit",
  })
}

/**
 * Cancela la reserva llamando a la API y ejecuta el callback onCancel.
 * @param {string} eventId
 */
const handleCancelarReserva = async (eventId) => {
  console.log(`Cancelando reserva con ID: ${eventId}`)
  try {
    setIsLoading(true)
    const url = API_URL + `/api/reservas/${eventId}`
    const token = localStorage.getItem("authToken")
    const headers = token ? { Authorization: `Bearer ${token}` } : {}
    await axios.delete(url, { headers })
    if (onCancel) onCancel(eventId)
    setConfirmDialogOpen(false)
  } catch (error) {
    console.error(`Error al cancelar la reserva: ${error}`)
  } finally {

```

Ilustración 231 days-details-modal II

```

        console.error(`Error al cancelar la reserva.`, error)
    } finally {
        setIsLoading(false)
    }
}

/**
 * Abre el diálogo de confirmación para cancelar una reserva.
 * @param {string} eventId
 */
const openConfirmDialog = (eventId) => {
    setSelectedEventId(eventId)
    setConfirmDialogOpen(true)
}

return (
    <>
    {/* Modal principal con detalles del día */}
    <Dialog open={isOpen} onOpenChange={onClose}>
        <DialogContent className="sm:max-w-[500px] p-0 overflow-hidden">
            <DialogHeader className="p-6 pb-2">
                <div className="flex items-center justify-between">
                    <DialogTitle className="text-xl font-semibold">{formatDate(day.date)}</DialogTitle>
                </div>
            </DialogHeader>

            <div className="px-6 py-2">
                <Badge variant="outline" className="bg-primary/10 text-primary border-primary/20">
                    {day.events.length} {day.events.length === 1 ? "evento" : "eventos"}
                </Badge>
            </div>

            <ScrollArea className="p-6 max-h-[60vh]">
                {day.events.length > 0 ? (
                    <div className="space-y-4">
                        {day.events.map((event) => (
                            <div key={event.id} className="bg-gray-50 rounded-lg p-4 border border-gray-100">
                                <h3 className="font-medium text-lg mb-2">{event.title}</h3>
                            </div>
                        ))
                    </div>
                ) : (
                    <div className="text-center py-8 text-gray-500">
                        <Calendar className="h-12 w-12 mx-auto mb-2 text-gray-300" />
                        <p>No hay eventos programados para este día</p>
                    </div>
                )}
            </ScrollArea>
        </DialogContent>
    </Dialog>
)

/* Diálogo de confirmación para cancelar reserva */
<Dialog open={confirmDialogOpen} onOpenChange={setConfirmDialogOpen}>
    <DialogContent className="sm:max-w-[425px]">
        <DialogHeader>
            <DialogTitle className="flex items-center gap-2">
                <AlertTriangle className="h-5 w-5 text-red-500" />
                Confirmar cancelación
            </DialogTitle>
        </DialogHeader>

```

Ilustración 232 days-details-modal III

```

        </DialogTitle>
    </DialogHeader>

```

Ilustración 233 days-details-modal IV

```

    </DialogHeader>
    <div className="py-4">
      <p className="text-gray-700">
        ¿Estás seguro que deseas cancelar esta reserva? Esta acción no se puede deshacer.
      </p>
    </div>
    <div className="flex justify-end gap-3">
      <Button variant="outline" onClick={() => setConfirmDialogOpen(false)}>
        Cancelar
      </Button>
      <Button variant="destructive" onClick={() => handleCancelarReserva(selectedEventId)} disabled={isLoading}>
        {isLoading ? "Cancelando..." : "Sí, cancelar reserva"}
      </Button>
    </div>
  </DialogContent>
</Dialog>
</>
)
}
}

```

Ilustración 234 days-details-modal V

Footer.jsx

Es el pie de página y es mostrado en todas las páginas gracias al enrutador



Ilustración 235 Footer

Código – Footer.jsx

```

// Importa componentes y hooks de React Router y los íconos de lucide-react
import { Link, useNavigate } from "react-router-dom"
import { Facebook, Instagram, LinkedIn, Twitter, Mail, Phone } from "lucide-react"

// Componente Funcional Footer
export default function Footer() {
  const navigate = useNavigate() // Hook para navegación programática (no se usa en este componente)

  return (
    // Contenedor principal del Footer con estilos de fondo y texto
    <footer className="bg-blue-700 text-white py-8">
      /* Contenedor centrado y con padding */
      <div className="mx-auto max-w-7xl px-2 sm:px-6 lg:px-8">
        /* Grid para organizar las 3 columnas del footer */
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-8">
          /* Columna 1: Enlaces rápidos */
          <div>
            <h3 className="text-lg font-semibold mb-4">Enlaces rápidos</h3>
            <ul className="space-y-2">
              /* Enlace a la página de inicio */
              <li>
                <Link to="/" className="text-gray-300 hover:text-white">
                  Inicio
                </Link>
              </li>
              /* Enlace a informativos */
              <li>
                <Link to="/informativos" className="text-gray-300 hover:text-white">
                  Informativos
                </Link>
              </li>
              /* Enlace a reservables */
              <li>
                <Link to="/disponibles" className="text-gray-300 hover:text-white">
                  Reservables
                </Link>
              </li>
            </ul>
          </div>
        </div>
      </div>
    </footer>
  )
}

```

Ilustración 236 Footer jsx I

```

/* Enlace a mis reservas */
- <Link to="/reservas" className="text-gray-300 hover:text-white">
    Mis Reservas
  </Link>

/* Enlace al carroto */
- <Link to="/carrito" className="text-gray-300 hover:text-white">
    | Carrito
  </Link>

/* Enlace al chatbox de IA */
- <Link to="/iacohere" className="text-gray-300 hover:text-white">
    | IA Chatbox
  </Link>

```

```

/* Columna 2: Información de contacto */


<h3 className="text-lg font-semibold mb-4">Contacto</h3>
  <ul className="space-y-2">
    /* Correo electrónico con icono */
    <li className="flex items-center gap-2">
      <Mail className="h-4 w-4 text-gray-300" />
      <a href="mailto:manuel5365@gmail.com" className="text-gray-300 hover:text-white">
        | manuel5365@gmail.com
      </a>
    </li>
    /* Teléfono con icono */
    <li className="flex items-center gap-2">
      <Phone className="h-4 w-4 text-gray-300" />
      <p className="text-gray-300">Teléfono: +34 641 130 893</p>
    </li>
  </ul>


```

Ilustración 237 Footer.jsx II

```

/* Columna 3: Redes sociales */


<h3 className="text-lg font-semibold mb-4">Síguenos</h3>
  /* Iconos de redes sociales con enlaces externos */
  <div className="flex space-x-4">
    /* Instagram */
    <a
      href="https://www.instagram.com"
      className="text-gray-300 hover:text-white"
      target="_blank"
      rel="noopener noreferrer"
    >
      <Instagram className="h-6 w-6" />
    </a>
    /* Facebook */
    <a
      href="https://www.facebook.com"
      className="text-gray-300 hover:text-white"
      target="_blank"
      rel="noopener noreferrer"
    >
      <Facebook className="h-6 w-6" />
    </a>
    /* Twitter */
    <a
      href="https://x.com"
      className="text-gray-300 hover:text-white"
      target="_blank"
      rel="noopener noreferrer"
    >
      <Twitter className="h-6 w-6" />
    </a>
    /* Linkedin */
    <a
      href="https://www.linkedin.com"
      className="text-gray-300 hover:text-white"
      target="_blank"
      rel="noopener noreferrer"
    >
      <Linkedin className="h-6 w-6" />
    </a>
  </div>


```

Ilustración 238 Footer.jsx III

```
    <linkedin className="h-6 w-6" />
  </a>
</div>
</div>
</div>

/* Pie de página con derechos de autor */
<div className="mt-8 text-center text-sm text-gray-400">
  <p>&copy; {new Date().getFullYear()} Marina Rent. Todos los derechos reservados.</p>
</div>
</div>
</footer>
)
```

Ilustración 239 Footer jxs IV

MarbellaMap.jxs

Es un componente descargado de internet que muestra el mapa de donde se sitúa nuestro negocio.



Ilustración 240 MarbellaMap

Código - MarbellaMap

```
/*
 * #file MarbellaMap.jsx
 * @description Muestra un mapa interactivo centrado en Marbella con un marcador y popup usando react-leaflet.
 * @module components/MarbellaMap
 */

import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";
import { LatitudeExpression } from "leaflet"; // Para definir la posición
import "leaflet/dist/leaflet.css"; // Estilo de Leaflet

/**
 * Componente MarbellaMap.
 * Muestra un mapa de Marbella con un marcador y popup.
 */
@function
@returns {JSX.Element} El mapa interactivo de Marbella.
*/
const MarbellaMap = () => {
    // Coordenadas de Marbella
    const position = [36.567, -4.882]; // Marbella

    return (
        <div className="my-16">
            <h2 className="text-sxl font-semibold text-center text-blue-900">
                ¿Dónde nos situamos?
            </h2>
            <div className="mt-8">
                <MapContainer center={position} zoom={18} style={{ height: "400px", width: "100%" }}>
                    <TileLayer
                        url="https://s.tile.openstreetmap.org/{z}/{x}/{y}.png"
                        attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
                    />
                    /* Marcador para Marbella */
                    <Marker position={position}>
                        <Popup>Estamos situados en Marbella</Popup>
                    </Marker>
                </MapContainer>
            </div>
        </div>
    );
};


```

Ilustración 241 MarbellaMap.jsx

Navbar.jsx

El navbar es el menú de navegación el cual está presente en todas las secciones de la página, este contiene los direccionamientos de las diferentes páginas



Ilustración 242 Navbar

Código – Navbar

```
/**
 * @file Navbar.jsx
 * @description Barra de navegación principal del sitio. Incluye enlaces a las páginas principales, menú de usuario y control de visibilidad para Admin.
 */

import { Disclosure, DisclosureButton, DisclosurePanel, Menu, MenuButton, MenuItem, MenuItems } from "@headlessui/react"
import LogoutButton from "../utilities/auth"
import { Bars3Icon, XMarkIcon } from "@heroicons/react/24/outline"
import { Link, useLocation, useNavigate } from "react-router-dom"

/**
 * Devuelve una cadena de clases CSS unidas por espacio, ignorando valores falsy.
 * @param {...string} classes
 * @returns {string}
 */
function classNames(...classes) {
  return classes.filter(Boolean).join(" ")
}

/**
 * Componente Navbar.
 * Muestra la barra de navegación superior con enlaces, menú de usuario y control de visibilidad para Admin.
 * @param {Object} props
 * @param {boolean} props.admin - Indica si el usuario es administrador.
 * @returns {JSX.Element} La barra de navegación.
 */

```

Ilustración 243 Navbar.jsx I

```
export default function Example({ admin }) {
  const location = useLocation()
  const navigate = useNavigate()

  // Incluir siempre todos los elementos de navegación, incluyendo Admin
  const navigation = [
    { name: "Inicio", href: "/home" },
    { name: "Informativos", href: "/informativos" },
    { name: "Reservables", href: "/alquilables" },
    { name: "Mis Reservas", href: "/reservas" },
    { name: "Carrito", href: "/carrito" },
    { name: "IA Chatbox", href: "/iacochere" },
    { name: "Admin", href: "/admin" }, // Siempre incluido pero se ocultará con CSS
  ].map((item) => ({
    ...item,
    current: location.pathname === item.href,
  }))

  return (
    <Disclosure as="nav" className="bg-white">
      {({ close }) => (
        <>
          <div className="mx-auto max-w-7xl px-2 sm:px-6 lg:px-8">
            <div className="relative flex h-16 items-center justify-between">
              <div className="absolute inset-y-0 left-0 flex items-center rounded-md p-2 text-gray-400 hover:text-gray-500">
                <span className="absolute inset-0.5" />
                <span className="sr-only">Open main menu:</span>
                <Bars3Icon aria-hidden="true" className="block size-6 group-data-open:hidden" />
                <XMarkIcon aria-hidden="true" className="hidden size-6 group-data-open:block" />
              </div>
              <div className="flex flex-1 items-center justify-center sm:items-stretch sm:justify-start">
                <div className="flex shrink-0 items-center">
                  
                </div>
                <div className="hidden sm:flex sm:items-center sm:justify-center flex-1">
                  <div className="flex space-x-4">
```

Ilustración 244 Navbar.jsx II

```

{navigation.map((item) => (
  // Ocultar el elemento Admin si admin es false
  <Link
    key={item.name}
    to={item.href}
    className={classNames(
      item.current ? "bg-blue-900 text-white" : "text-black hover:bg-blue-900 hover:text-white",
      "rounded-md px-3 py-2 text-sm font-medium transition-opacity duration-300",
      // Ocultar Admin si no es admin
      item.name === "Admin" && !admin ? "opacity-0 invisible absolute" : "opacity-100 visible",
    )}
    aria-hidden={item.name === "Admin" && !admin ? "true" : "false"}
    tabIndex={item.name === "Admin" && !admin ? -1 : 0}
  >
  {item.name}
  </Link>
))
</div>
</div>
</div>

<div className="absolute inset-y-0 right-0 flex items-center pr-2 sm:static sm:inset-auto sm:ml-6 sm:pr-0">
  <Menu as="div" className="relative ml-3">
    <div>
      <MenuButton className="relative flex rounded-full bg-gray-800 text-sm focus:ring-2 focus:ring-white focus:ring-offset-2 focus-visible:outline-0">
        <span className="absolute inset-1.5" />
        <span className="sr-only">Open user menu</span>
        
      </MenuButton>
    </div>
    <MenuItems className="absolute right-0 z-10 mt-2 w-48 origin-top-right rounded-md bg-white py-1 ring-1 shadow-lg ring-black">
      <MenuItem>
        {({ active }) => (
          <button
            onClick={() => navigate("/perfil")}
            className={classNames(
              active ? "bg-blue-200" : "",
              "w-full text-left px-4 py-2 text-sm text-black",
            )}
          >
            Mi Perfil
          </button>
        )}
      </MenuItem>
      <MenuItem>
        <LogoutButton />
      </MenuItem>
    </MenuItems>
  </Menu>
</div>
</div>
</div>

<DisclosurePanel className="sm:hidden">
  <div className="space-y-1 px-2 pt-2 pb-3">
    {navigation.map((item) => (
      // Changed to use navigate instead of href for mobile menu
      <DisclosureButton
        key={item.name}
        as="button" // Changed from "a" to "button"
        onClick={() => {
          navigate(item.href)
          close() // Close the mobile menu after navigation
        }}
        aria-current={item.current ? "page" : undefined}
        className={classNames(
          item.current ? "bg-blue-900 text-white" : "text-black hover:bg-blue-900 hover:text-white",
          "block w-full text-left rounded-md px-3 py-2 text-base font-medium transition-opacity duration-300",
          // Ocultar Admin si no es admin
          item.name === "Admin" && !admin ? "hidden" : "block",
        )}
      >
        {item.name}
      </DisclosureButton>
    ))
  </div>
</div>

```

Ilustración 245 Navbar.jsx III

```

        </div>
      </div>
    </div>
  </div>
</div>

<div className="space-y-1 px-2 pt-2 pb-3">
  {navigation.map((item) => (
    // Changed to use navigate instead of href for mobile menu
    <DisclosureButton
      key={item.name}
      as="button" // Changed from "a" to "button"
      onClick={() => {
        navigate(item.href)
        close() // Close the mobile menu after navigation
      }}
      aria-current={item.current ? "page" : undefined}
      className={classNames(
        item.current ? "bg-blue-900 text-white" : "text-black hover:bg-blue-900 hover:text-white",
        "block w-full text-left rounded-md px-3 py-2 text-base font-medium transition-opacity duration-300",
        // Ocultar Admin si no es admin
        item.name === "Admin" && !admin ? "hidden" : "block",
      )}
    >
      {item.name}
    </DisclosureButton>
  ))
</div>

```

Ilustración 246 Navbar.jsx IV

```

        aria-hidden={item.name === "Admin" && !admin ? "true" : "false"}
        tabIndex={item.name === "Admin" && !admin ? -1 : 0}
      >
    | {item.name}
    </DisclosureButton>
  ))}
</div>
</DisclosurePanel>
</>
)
</Disclosure>
}
}

```

Ilustración 247 Navbar.jsx V

Person-chooser.jsx

Componente usado para seleccionar el número de personas para la hora de la reserva tanto en el apartado de reservar como en el de admin.

Código – Person-chooser

```

/** 
 * #file person-chooser.jsx
 * #description Selector visual para elegir el número de personas, deshabilitando opciones según disponibilidad.
 * #module components/person-chooser
 */

import { useState } from 'react';

/**
 * SelectorPersonas
 * Componente para seleccionar el número de personas, mostrando opciones deshabilitadas si exceden la disponibilidad.
 */
const SelectorPersonas = ({ personasDisponibles, setPersonas, selected }) => {
  const [selectedPerson, setSelectedPerson] = useState(selected);

  /**
   * Maneja el cambio de selección de persona.
   * @param {React.ChangeEvent<HTMLInputElement>} e
   */
  const handleSelect = (e) => {
    setSelectedPerson(e.target.value);
    setPersonas(e.target.value);
  };
}

```

Ilustración 248 Person-chooser.jsx I

```

return (
  <fieldset aria-label="Selecciona número de personas" className="mt-4">
    <div className="grid grid-cols-4 gap-4 sm:grid-cols-8 lg:grid-cols-4">
      {[1, 2, 3, 4].map((num) => (
        const isEnabled = num > personasDisponibles;
        const isSelected = selectedPerson === String(num);

        const baseClasses =
          "group relative flex items-center justify-center rounded-md border px-4 py-3 text-sm font-medium uppercase sm:flex-1 sm:py-6 transition";
        const enabledClasses =
          "cursor-pointer bg-white text-gray-900 shadow-xs hover:bg-gray-50";
        const disabledClasses =
          "cursor-not-allowed bg-gray-50 text-gray-200";
        const borderClasses = isSelected
          ? "border-4 border-indigo-500"
          : "border-gray-300";

        return (
          <label
            key={num}
            className={`${[isEnabled ? enabledClasses : disabledClasses] ${borderClasses}}`}>
            <input
              type="radio"
              name="size-choice"
              value={num}
              className="sr-only"
              onChange={handleSelect}
              checked={isSelected}
              disabled={isDisabled}
            />
            <span>{num}</span>
          {isDisabled ? (
            <span>
              aria-hidden="true"
              className="pointer-events-none absolute -inset-px rounded-md border-2 border-gray-200"
            </span>
          ) : null}
        
      ))
    </div>
  </fieldset>
)

```

Ilustración 249 Person-chooser.jsx II

```

    >           |
    <svg
      |   className="absolute inset-0 size-full stroke-2 text-gray-200"
      |   viewBox="0 0 100 100"
      |   preserveAspectRatio="none"
      |   stroke="currentColor"
      |   >
      |     <line
      |       |   x1="0"
      |       |   y1="100"
      |       |   x2="100"
      |       |   y2="0"
      |       |   vectorEffect="non-scaling-stroke"
      |     />
      |   </svg>
      | </span>
    ) : (
      |   <span
      |     |   className="pointer-events-none absolute -inset-px rounded-md"
      |     |   aria-hidden="true"
      |   />
      | )
    );
  );
}

</div>
</fieldset>
);
};

export default selectorPersonas;

```

Ilustración 250 Person-chooser.jsx III

Time-slider.jsx

Es el encargado de poder seleccionar la hora en las reservas y en el modal de modificar reserva.



Ilustración 251 Time-slider

Código – Time-Slider

```
/*
 * @file time-slider.jsx
 * @description Componente selector de hora con slider visual, iconos según el momento del día y control de disponibilidad.
 * @module components/time-slider
 */

import { useState, useEffect } from "react"
import { Slider } from "@components/ui/slider"
import { Clock, Sun, Sunrise, Sunset, Moon } from "lucide-react"

/**
 * Componente TimeSlider.
 * Permite seleccionar una hora del día con un slider, mostrando iconos y estilos según el momento (amanecer, día, atardecer, noche).
 *
 * @param {Object} props
 * @param {string|number} props.id - Identificador del slider.
 * @param {number} props.hora - Hora inicial seleccionada.
 * @param {function} props.setHora - Función para actualizar la hora seleccionada.
 * @param {boolean} props.disponible - Indica si la hora está disponible.
 * @param {boolean} props.horaPasada - Indica si la hora ya ha pasado.
 * @returns {JSX.Element} El slider de selección de hora.
 */
export default function TimeSlider({ id, hora, setHora, disponible, horaPasada }) {
  const [hour, setHour] = useState(hora)
  const [minute, setMinute] = useState(0)
  const [timeOfDay, setTimeOfDay] = useState("day")

  /**
   * Maneja el cambio del slider y actualiza la hora y minutos.
   * @param {Array<number>} value
   */
  const handleSliderChange = (value) => {
    const totalHours = value[0]
    const hours = Math.floor(totalHours)
    const mins = Math.round((totalHours - hours) * 60)
    setHour(hours)
    setMinute(mins)
  }
}
```

Ilustración 252 Time-slider.jsx I

```
/*
 * Formatea la hora y minutos a string HH:mm.
 * @returns {string}
 */
const formatTime = () => {
  const formattedHour = hour.toString().padStart(2, "0")
  const formattedMinute = minute.toString().padStart(2, "0")
  return `${formattedHour}:${formattedMinute}`
}

/**
 * Devuelve el ícono correspondiente al momento del día.
 * @returns {JSX.Element}
 */
const getTimeIcon = () => {
  switch (timeOfDay) {
    case "dawn":
      return <Sunrise className="h-8 w-8 text-amber-500" />
    case "day":
      return <Sun className="h-8 w-8 text-yellow-500" />
    case "dusk":
      return <Sunset className="h-8 w-8 text-orange-500" />
    case "night":
      return <Moon className="h-8 w-8 text-white" />
    default:
      return <Clock className="h-8 w-8 text-gray-700" />
  }
}
```

Ilustración 253 Time-slider.jsx II

```

    /**
     * Devuelve los estilos de fondo según el momento del día.
     * @returns {string}
     */
    const getTimeStyles = () => {
      switch (timeOfDay) {
        case "dawn":
          return "bg-gradient-to-br from-amber-100 to-rose-100 text-amber-900"
        case "day":
          return "bg-gradient-to-br from-sky-100 to-blue-100 text-sky-900"
        case "dusk":
          return "bg-gradient-to-br from-orange-100 to-rose-100 text-orange-900"
        case "night":
          return "bg-gradient-to-br from-indigo-900 to-purple-900 text-white"
        default:
          return "bg-gradient-to-br from-gray-100 to-gray-200 text-gray-900"
      }
    }

    /**
     * Muestra mensajes de disponibilidad según el estado.
     * @returns {JSX.Element|undefined}
     */
    const comprobarDisponibilidad = () => {
      if (disponible === false)
        return <div className="text-red-500 font-semibold">No Disponible</div>
      else if (horaPasada === true)
        return <div className="text-red-500 font-semibold">Hora pasada</div>
    }
  }

```

Ilustración 254 Time-slider.jsx III

```

  return (
    <div className="w-full max-w-md mt-6 mx-auto p-6 rounded-lg shadow-lg transition-colors duration-500 ${getTimeStyles()}">
      <div className="flex flex-col items-center space-y-6">
        {/* Línea con ícono, hora y AM/PM */}
        <div className="flex items-center gap-3">
          {getTimeIcon()}
          <div className="text-3xl font-bold tracking-tight">{formatTime()}</div>
          <div className="text-sm font-medium opacity-80">{hour < 12 ? "AM" : "PM"}</div>
        </div>

        {/* Mostrar disponibilidad */}
        {comprobarDisponibilidad()}

        {/* Slider */}
        <div className="w-full px-2">
          <Slider value={[hour]} max={22} step={1} onValueChange={handleSliderChange} onPointerUp={() => setHora(hour)} className="my-4" />
          <div className="flex justify-between text-xs font-medium opacity-70">
            <span>08:00</span>
            <span>15:00</span>
            <span>22:00</span>
          </div>
        </div>
      </div>
    </div>
  )
}

```

Ilustración 255 Time-slider.jsx IV

Usuarios.jsx

Es el encargado de mostrar los usuarios en el apartado de admin usuarios

Panel de Administración

Gestiona usuarios, reservas y anuncios desde este panel

USUARIO	EMAIL	FECHA NACIMIENTO	ROL	RESTABLECER CONTRASEÑA	ACCIONES
MG Manuel Gonzalez	manuel5365.mgl@gmail.com	2004-01-03	admin	Enviar Restablecimiento	Editar Borrar
MC Manuel Cuenta222	manuelgonzalezperez@iesflorenciopintado.es	2024-11-29	usuario	Enviar Restablecimiento	Editar Borrar
LK Lennie Kessler	pcmanuela3@gmail.com	2011-11-04	usuario	Enviar Restablecimiento	Editar Borrar

Ilustración 256 Usuarios

Código – Usuarios

```
/**
 * @file usuarios.jsx
 * @description Vista de administración para listar, buscar, crear, editar y eliminar usuarios.
 * Incluye paginación, búsqueda, restablecimiento de contraseña y modales para edición/creación.
 */

import { useEffect, useState } from "react"
import { Search, ChevronLeft, ChevronRight, AlertTriangle } from "lucide-react"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import axios from "axios"
import { Dialog,DialogContent,DialogTitle } from "@/components/ui/dialog"
import { API_URL } from "../../utilities/apiRest"
import { Avatar,AvatarFallback } from "@/components/ui/avatar"
import UsuariosModal from "./modal-usuarios"

export default function Usuarios() {
    /**
     * Lista de usuarios cargados.
     * {[Array, Function]}
     */
    const [usuarios, setUsuarios] = useState([])
    /**
     * Estado de carga.
     * {[boolean, Function]}
     */
    const [loading, setLoading] = useState(true)
    /**
     * Estado para mostrar el modal de edición/creación.
     * {[boolean, Function]}
     */
    const [isModalOpen, setIsModalOpen] = useState(false)
    /**
     * Usuario seleccionado para editar.
     * {[Object|null, Function]}
     */
    const [selectedUser, setSelectedUser] = useState(null)
```

Ilustración 257 Usuarios jsx I

```
/**  
 * Estado para mostrar el diálogo de confirmación de borrado.  
 * {[boolean, Function]}  
 */  
const [confirmDialogOpen, setConfirmDialogOpen] = useState(false)  
/**  
 * Usuario seleccionado para eliminar.  
 * {[Object|null, Function]}  
 */  
const [userToDelete, setUserToDelete] = useState(null)  
/**  
 * Estado de carga para el borrado.  
 * {[boolean, Function]}  
 */  
const [isDeleting, setIsDeleting] = useState(false)  
/**  
 * Estado de búsqueda.  
 * {(string|Function)[]}  
 */  
const [searchQuery, setSearchQuery] = useState("")  
/**  
 * Mensaje de feedback tras acciones.  
 * {(string|Function)[]}  
 */  
const [feedbackMsg, setFeedbackMsg] = useState("")  
  
// Pagination states  
/**  
 * Página actual.  
 * {[number, Function]}  
 */  
const [currentPage, setCurrentPage] = useState(1)  
/**  
 * Total de páginas.  
 * {[number, Function]}  
 */  
const [totalPages, setTotalPages] = useState(1)
```

Ilustración 258 Usuarios.jsx II

```
useEffect(() => {
  fetchUsuarios(currentPage)
  setFeedbackMsg("")
}, [isModalOpen, confirmDialogOpen, currentPage])

/**
 * Obtiene los usuarios paginados desde la API.
 * @param {number} page
 */
const fetchUsuarios = (page = 1) => {
  setLoading(true)
  const token = localStorage.getItem("authToken")
  const headers = token ? { Authorization: `Bearer ${token}` } : {}

  axios
    .post(API_URL + "api/usuariosPaginados", { pagina: page }, { headers })
    .then((response) => {
      if (response.data.success) {
        setUsuarios(response.data.data)
        setCurrentPage(response.data.page)
        setTotalPages(response.data.totalPages)
      } else {
        console.error("Error en la respuesta:", response.data)
      }
    })
    .catch((error) => {
      console.error("Error al cargar los usuarios:", error)
    })
    .finally(() => {
      setLoading(false)
    })
}

}
```

Ilustración 259 Usuarios jsx III

```

    /**
     * Abre el modal para editar o crear usuario.
     * @param {Object|null} user
     */
    const handleEditClick = (user) => {
        setSelectedUser(user)
        setIsModalOpen(true)
    }

    /**
     * Envía un correo de restablecimiento de contraseña.
     * @param {Object} user
     */
    const handleEnviarRestablecimientoClick = (user) => {
        const token = localStorage.getItem("authToken")
        const headers = token ? { Authorization: `Bearer ${token}` } : {}
        axios
            .post(API_URL + "api/enviar-restablecimiento", { email: user.Email }, { headers })
            .then(() => {
                setFeedbackMsg("Correo de restablecimiento enviado correctamente.")
                setTimeout(() => setFeedbackMsg("", 2000)
            })
            .catch((error) => {
                setFeedbackMsg("Error al enviar el correo de restablecimiento.")
                setTimeout(() => setFeedbackMsg("", 2000)
                console.error("Error al enviar restablecimiento:", error)
            })
    }
}

```

Ilustración 260 Usuarios jsx IV

```

    /**
     * Elimina el usuario seleccionado.
     */
    const handleDelete = () => {
        if (!userToDelete) return

        setIsDeleting(true)
        const token = localStorage.getItem("authToken")
        const headers = token ? { Authorization: `Bearer ${token}` } : {}

        axios
            .delete(`${API_URL}api/usuarios/${userToDelete.id}`, { headers })
            .then(() => {
                fetchUsuarios(currentPage)
                setConfirmDialogOpen(false)
                setUserToDelete(null)
                setFeedbackMsg("Usuario eliminado correctamente.")
                setTimeout(() => setFeedbackMsg("", 2000)
            })
            .catch((error) => {
                setFeedbackMsg("Error al eliminar usuario.")
                setTimeout(() => setFeedbackMsg("", 2000)
                console.error("Error al eliminar usuario:", error)
            })
            .finally(() => {
                setIsDeleting(false)
            })
    }

    /**
     * Cambia la página actual de la paginación.
     * @param {number} newPassword
     */
    const handlePageChange = (newPage) => {
        if (newPage >= 1 && newPassword <= totalPages) {
            setCurrentPage(newPage)
        }
    }
}

```

Ilustración 261 Usuarios jsx V

```
// Filtra usuarios según la búsqueda
const filteredUsuarios = usuarios.filter(
  (user) =>
    user.Nombre.toLowerCase().includes(searchQuery.toLowerCase()) ||
    user.Apellidos.toLowerCase().includes(searchQuery.toLowerCase()) ||
    user.Email.toLowerCase().includes(searchQuery.toLowerCase()),
)
}

return (
  <>
  {loading ? (
    <div className="flex justify-center items-center h-64">
      <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
    </div>
  ) : (
    <>
      <div className="flex justify-between items-center mb-6">
        <div className="relative w-full max-w-md">
          <Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400" size={20} />
          <input
            type="search"
            placeholder="Buscar usuarios..."
            className="pl-10 pr-4 py-2 w-full"
            value={searchQuery}
            onChange={(e) => setSearchQuery(e.target.value)}
          />
        </div>
        <Button size="sm" onClick={() => handleEditClick(null)}>
          Añadir Usuario
        </Button>
      </div>
    </div>
    /* Mensaje de feedback */
    {feedbackMsg && (
      <div className="mb-4 px-4 py-2 rounded bg-green-100 text-green-800 border border-green-200 text-center font-medium">
        {feedbackMsg}
      </div>
    )}
  )}
)
```

Ilustración 262 Usuarios jsx VI

```
<div className="bg-white rounded-lg border overflow-sm-x-scroll overflow-y-hidden">
  <table className="min-w-full divide-y divide-gray-200">
    <thead className="bg-gray-50">
      <tr>
        <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          | Usuario
        </th>
        <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          | Email
        </th>
        <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          | Fecha Nacimiento
        </th>
        <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          | Rol
        </th>
        <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          | Restablecer Contraseña
        </th>
        <th className="px-6 py-3 text-right text-xs font-medium text-gray-500 uppercase tracking-wider">
          | Acciones
        </th>
      </tr>
    </thead>
    <tbody className="bg-white divide-y divide-gray-200">
      {filteredUsuarios.map((user, i) => (
        <tr key={i}>
          <td className="px-6 py-4 whitespace nowrap">
            <div className="flex items-center">
              <div className="flex-shrink-0 h-10 w-10">
                <Avatar>
                  <AvatarFallback>
                    {user.Nombre.split(" ")}
                    .map((n) => n[0])
                    .join("")
                  {user.Apellidos.split(" ")}
                  .map((n) => n[0])
                  .join("")
                </AvatarFallback>
              </Avatar>
            </div>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
</div>
```

Ilustración 263 Usuarios jsx VII

```

        </div>
        <div className="ml-4">
            <div className="text-sm font-medium text-gray-900">
                {user.Nombre} {user.Apellidos}
            </div>
        </div>
    </td>
    <td className="px-6 py-4 whitespace-nowrap">
        <div className="text-sm text-gray-500">{user.Email}</div>
    </td>
    <td className="px-6 py-4 whitespace-nowrap">
        <div className="text-sm text-gray-500">{user.Fecha_nacimiento}</div>
    </td>
    <td className="px-6 py-4 whitespace-nowrap">
        <span>
            className={`px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${user.Tipo === "admin" ? "bg-green-100 text-white" : "bg-red-100 text-white"} ${user.Tipo}</span>
    </td>
    <td className="px-6 py-4 whitespace-nowrap">
        <button variant="outline" size="sm" onClick={() => handleEnviarRestablecimientoClick(user)}>
            Enviar Restablecimiento
        </button>
    </td>
    <td className="px-6 py-4 whitespace-nowrap text-right text-sm font-medium flex gap-1 justify-end">
        <button variant="outline" size="sm" onClick={() => handleEditClick(user)}>
            Editar
        </button>
        <button variant="destructive" size="sm" onClick={() => {
            setUserToDelete(user)
            setConfirmDialogOpen(true)
        }}>
            Eliminar
        </button>
    </td>
</tr>
)
</tbody>
</table>
</div>

```

Ilustración 264 Usuarios jsx VIII

```

        <td>
            <button type="button" onClick={() => handleDeleteUser(user)}> Eliminar </button>
        </td>
    </tr>
)
</tbody>
</table>
</div>

/* Pagination Controls */
!searchQuery && (
    <div className="flex justify-center items-center mt-8 gap-2">
        <button
            variant="outline"
            size="sm"
            onClick={() => handlePageChange(currentPage - 1)}
            disabled={currentPage === 1}
        >
            <ChevronLeft className="h-4 w-4" />
        </button>
        <span className="text-sm">
            Página {currentPage} de {totalPages}
        </span>
        <button
            variant="outline"
            size="sm"
            onClick={() => handlePageChange(currentPage + 1)}
            disabled={currentPage === totalPages}
        >
            <ChevronRight className="h-4 w-4" />
        </button>
    </div>
)
</>
)
</>
}

{isModalOpen && <UsuariosModal isOpen={isModalOpen} setIsOpen={setIsModalOpen} user={selectedUser} />}

```

Ilustración 265 IX

```

<Dialog open={confirmDialogOpen} onChange={setConfirmDialogOpen}>
  <DialogContent className="sm:max-w-[425px]">
    <DialogTitle>
      <AlertTriangle className="h-5 w-5 text-red-500" />
      Confirmar eliminación
    </DialogTitle>
    <div className="py-4">
      <p className="text-gray-700">
        ¿Estás seguro que deseas eliminar este usuario? Esta acción no se puede deshacer.
      </p>
    </div>
    <div className="flex justify-end gap-3">
      <Button variant="outline" onClick={() => setConfirmDialogOpen(false)}>
        Cancelar
      </Button>
      <Button variant="destructive" onClick={handleDelete} disabled={isDeleting}>
        {isDeleting ? "Eliminando..." : "Sí, eliminar"}
      </Button>
    </div>
  </DialogContent>
</Dialog>
</>
}

```

Ilustración 266 Usuarios.jsx X

Modal-usuarios.jsx

Es el modal que se despliega desde usuarios a la hora de añadir o editar un usuario.

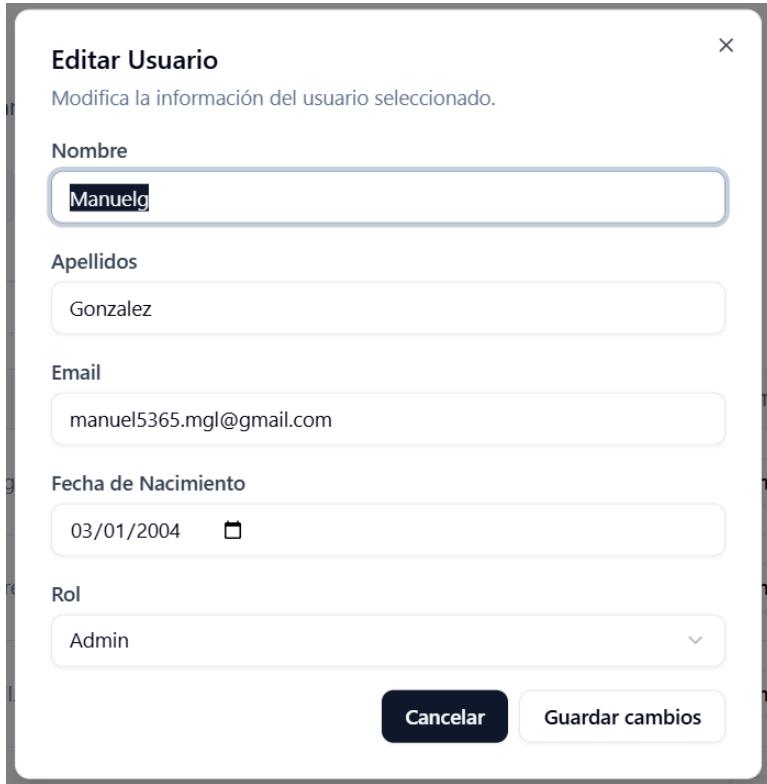


Ilustración 267 Modal-usuarios

Código – Modal-usuarios

```

import React, { useState, useEffect } from 'react';
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { Dialog,DialogContent, DialogHeader, DialogFooter,DialogTitle, DialogDescription } from "@/components/ui/dialog";
import axios from "axios";
import { API_URL } from '../../../../../utilities/apirest';

/**
 * Modal para crear o editar usuarios desde el panel de administración.
 * Permite modificar datos personales y el rol del usuario.
 * @component
 * @param {Object} props
 * @param {boolean} props.isOpen - Si el modal está abierto.
 * @param {Function} props.setIsOpen - Función para abrir/cerrar el modal.
 * @param {Object|null} props.user - Usuario a editar (si existe) o null para crear.
 * @returns {JSX.Element}
 */
export default function UsuariosModal({ isOpen, setIsOpen, user }) {
    /**
     * Estado de carga para el guardado.
     * {[boolean, Function]}
     */
    const [isLoading, setIsLoading] = useState(false)
    /**
     * Estado del formulario del usuario.
     * {[Object, Function]}
     */
    const [formData, setFormData] = useState({
        Nombre: '',
        Apellidos: '',
        Email: '',
        Fecha_nacimiento: '',
        Tipo: ''
    });
}

```

Ilustración 268 Modal-Usuarios.jsx I

```

    /**
     * Efecto para cargar los datos del usuario al abrir el modal.
     */
    useEffect(() => {
        if (user) {
            setFormData({
                id_usuario: user.id,
                Nombre: user.Nombre,
                Apellidos: user.Apellidos,
                Email: user.Email,
                Fecha_nacimiento: user.Fecha_nacimiento,
                Tipo: user.Tipo
            });
        }
    }, [user]);

    /**
     * Maneja el cambio de los campos del formulario.
     * @param {React.ChangeEvent<HTMLInputElement>} e
     */
    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData((prevData) => ({ ...prevData, [name]: value }));
    };
}

```

Ilustración 269 Modal-Usuarios.jsx II

```

    /**
     * Guarda los cambios del usuario (actualiza o crea).
     */
    const handleSave = () => {
        setIsLoading(true)
        const token = localStorage.getItem("authToken");
        const headers = token ? { Authorization: `Bearer ${token}` } : {};
        if (user != null) {
            axios.post(`${API_URL}api/usuarios/actualizar`, formData, { headers })
                .then((response) => {
                    console.log('Usuario actualizado', response.data);
                    setIsOpen(false); // Cerrar el modal
                })
                .catch((error) => {
                    console.error('Error al actualizar usuario', error);
                }).finally(() => {
                    setIsLoading(false);
                });
        } else {
            axios.post(`${API_URL}api/usuarios`, formData, { headers })
                .then((response) => {
                    console.log('Usuario guardado', response.data);
                    setIsOpen(false); // Cerrar el modal
                })
                .catch((error) => {
                    console.error('Error al actualizar usuario', error);
                }).finally(() => {
                    setIsLoading(false);
                });
        }
    };
}

```

Ilustración 270 Modal-Usuarios.jsx III

```

return (
    <Dialog open={isopen} onOpenChange={(open) => setIsOpen(open)}>
        <DialogContent>
            <DialogTitle>Editar Usuario</DialogTitle>
            <DialogDescription>Modifica la información del usuario seleccionado.</DialogDescription>
        </DialogContent>
        <div className="space-y-4">
            {/* Campo Nombre */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Nombre</label>
            <Input
                label="Nombre"
                name="Nombre"
                value={formData.Nombre}
                onChange={handleChange}
            />
            {/* Campo Apellidos */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Apellidos</label>
            <Input
                label="Apellidos"
                name="Apellidos"
                value={formData.Apellidos}
                onChange={handleChange}
            />
            {/* Campo Email */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Email</label>
            <Input
                label="Email"
                name="Email"
                value={formData.Email}
                onChange={handleChange}
            />
            {/* Campo Fecha de Nacimiento */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Fecha de Nacimiento</label>
        </div>
    </Dialog>
)

```

Ilustración 271 Modal-Usuarios.jsx IV

```

<input
  label="Fecha de Nacimiento"
  name="Fecha_nacimiento"
  type="date"
  value={formData.Fecha_nacimiento}
  onChange={handlechange}
/>
{/* Campo Contraseña solo al crear usuario */}
{user === null && (
  <>
    <label className="text-sm font-medium text-gray-700 mb-1 block">Contraseña</label>
    <Input
      label="Contraseña"
      name="password"
      type="password"
      onChange={handlechange}
    />
  </>
)}
 {/* selector de rol */}
<div>
  <label className="text-sm font-medium text-gray-700 mb-1 block">Rol</label>
  <Select
    value={formData.Tipo}
    onValueChange={(value) =>
      setFormData((prevData) => ({ ...prevData, Tipo: value }))
    }
  >
    <SelectTrigger className="w-full">
      <SelectValue placeholder="Selecciona un rol" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="admin">Admin</SelectItem>
      <SelectItem value="usuario">Usuario</SelectItem>
    </SelectContent>
  </select>
</div>
</div>

```

Ilustración 272 Modal-Usuarios.jsx V

```

<DialogFooter>
  {isLoading ? (
    <Button variant="outline">Guardando...</Button>
  ) : (
    <>
      <Button onClick={() => setIsOpen(false)}>Cancelar</Button>
      <Button variant="outline" onClick={handleSave}>Guardar cambios</Button>
    </>
  )}
</DialogFooter>
</DialogContent >
</Dialog >
};

```

Ilustración 273 Modal-Usuarios.jsx VI

Publicacion.jsx

Es la parte de editar publicaciones que viene en la sección de admin.

Esta sección usa modal-publicaciones.jsx que vamos a explicar en el siguiente apartado.

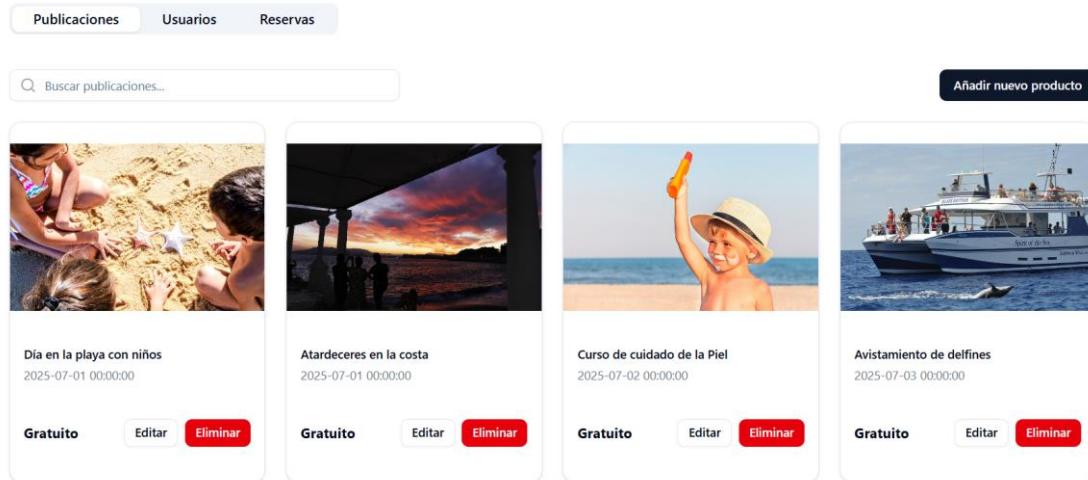


Ilustración 274 Publicacion

Código – Publicacion

```
/*
 * @file publicaciones.jsx
 * @description Vista de administración para listar, buscar, crear, editar y eliminar publicaciones.
 * Incluye paginación, búsqueda y modales para edición/creación.
 */

import { useEffect, useState } from "react"
import { Search, AlertTriangle, ChevronLeft, ChevronRight } from "lucide-react"
import { Input } from "@/components/ui/input"
import { Button } from "@/components/ui/button"
import { Card,CardContent,CardFooter } from "@/components/ui/card"
import { Dialog,DialogContent,DialogTitle } from "@/components/ui/dialog"
import axios from "axios"
import { API_URL, IMAGE_URL } from "../../utilities/apiRest"
import ProductModal from "./modal-publicaciones"

/**
 * Componente principal para la gestión de publicaciones.
 * Permite buscar, pagination, crear, editar y eliminar publicaciones.
 * @component
 * @returns {JSX.Element}
 */
export default function Publicaciones() {
  /**
   * Estado para la búsqueda de publicaciones.
   * {string|Function}[]
   */
  const [searchQuery, setSearchQuery] = useState("")
  /**
   * Lista de publicaciones cargadas.
   * {[Array, Function]}
   */
  const [publicaciones, setPublicaciones] = useState([])
  /**
   * Estado de carga.
   * {[boolean, Function]}
   */
}
```

Ilustración 275 Publicacion jsx /

```

    /**
     * Estado de carga.
     * {[boolean, Function]}
     */
    const [loading, setLoading] = useState(true)
    /**
     * Estado para mostrar el modal de edición.
     * {[boolean, Function]}
     */
    const [isModalOpen, setIsModalOpen] = useState(false)
    /**
     * Publicación seleccionada para editar.
     * {[Object, Function]}
     */
    const [publication, setPublication] = useState()
    /**
     * Estado para mostrar el modal de añadir.
     * {[boolean, Function]}
     */
    const [addModalOpen, setAddModalOpen] = useState(false)

    // Estados de paginación
    /**
     * Página actual.
     * {[number, Function]}
     */
    const [currentPage, setCurrentPage] = useState(1)
    /**
     * Total de páginas.
     * {[number, Function]}
     */
    const [totalPages, setTotalPages] = useState(1)

    // Estados para eliminar con confirmación
    /**
     * Estado para mostrar el diálogo de confirmación de borrado.
     * {[boolean, Function]}
     */
    const [confirmDialogOpen, setConfirmDialogOpen] = useState(false)

```

Ilustración 276 Publicacion jsx II

```

    /**
     * ID de la publicación seleccionada para eliminar.
     * {[number|null, Function]}
     */
    const [selectedPublicationId, setSelectedPublicationId] = useState(null)
    /**
     * Estado de carga para el borrado.
     * {[boolean, Function]}
     */
    const [isDeleting, setIsDeleting] = useState(false)

    /**
     * Obtiene las publicaciones paginadas desde la API.
     * @param {number} page
     */
    const fetchPublicaciones = (page = 1) => {
        setLoading(true)
        const token = localStorage.getItem("authToken")
        const headers = token ? { Authorization: `Bearer ${token}` } : {}

        axios
            .post(`${API_URL}api/publicacionesPaginadas`, { pagina: page }, { headers })
            .then((response) => {
                if (response.data.success) {
                    console.log("Publicaciones cargadas:", response.data.data)
                    setPublicaciones(response.data.data)
                    setCurrentPage(response.data.page)
                    setTotalPages(response.data.totalPages)
                } else {
                    console.error("Error en la respuesta:", response.data)
                }
            })
            .catch((error) => {
                console.error("Error al cargar las publicaciones:", error)
            })
            .finally(() => setLoading(false))
    }

```

Ilustración 277 Publicacion jsx III

```

    /**
     * Efecto para cargar publicaciones al abrir/cerrar modales o eliminar.
     */
    useEffect(() => {
      fetchPublicaciones(currentPage)
    }, [isModalOpen, addModalOpen, isDeleting])

    /**
     * Callback al actualizar una publicación.
     * @param {Object} updatedPublication
     */
    const handleUpdate = (updatedPublication) => {
      setPublication(updatedPublication)
    }

    /**
     * Callback al añadir un nuevo producto.
     * @param {Object} newProduct
     */
    const handleProductAdded = (newProduct) => {
      console.log("Producto añadido:", newProduct)
      fetchPublicaciones(1) // Volver a la primera página después de añadir
    }
  
```

Ilustración 278 Publicacion.jsx IV

```

    /**
     * Elimina una publicación seleccionada.
     */
    const handleDelete = async () => {
      if (!selectedPublicationId) return
      setIsDeleting(true)

      try {
        const token = localStorage.getItem("authToken")
        const headers = token ? { Authorization: `Bearer ${token}` } : {}
        await axios.delete(`${API_URL}api/publicaciones/${selectedPublicationId}`, { headers })

        // Refetch data instead of manually filtering
        fetchPublicaciones(currentPage)
        setConfirmDialogOpen(false)
        setSelectedPublicationId(null)
      } catch (error) {
        console.error("Error al eliminar la publicación:", error)
        alert("Ocurrió un error al intentar eliminar la publicación.")
      } finally {
        setIsDeleting(false)
      }
    }

    /**
     * Cambia la página actual de la paginación.
     * @param {number} newPassword
     */
    const handlePageChange = (newPage) => {
      if (newPage >= 1 && newPage <= totalPages) {
        setCurrentPage(newPage)
        fetchPublicaciones(newPage)
      }
    }
  
```

Ilustración 279 Publicacion.jsx V

```

    /**
     * Filtra las publicaciones según la búsqueda.
     */
    const filteredPublicaciones = publicaciones.filter(item) =>
      item.titulo.toLowerCase().includes(searchQuery.toLowerCase()),
    )

    return (
      <>
        {/* Loader mientras se cargan los datos */}
        {loading ? (
          <div className="flex justify-center items-center h-64">
            <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin" />
          </div>
        ) : (
          <>
            {/* Barra de búsqueda y botón para añadir */}
            <div className="flex justify-between items-center mb-6">
              <div className="relative w-full max-w-md">
                <Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400" size={20} />
                <Input
                  type="search"
                  placeholder="Buscar publicaciones..."
                  className="pl-10 pr-4 py-2 w-full"
                  value={searchQuery}
                  onChange={(e) => setSearchQuery(e.target.value)}
                />
              </div>
              <Button onClick={() => setAddModalOpen(true)}>Añadir nuevo producto</Button>
            </div>
          </>
        )}
      </>
    )
  )
}

```

Ilustración 280 Publicacion jsx VI

```

/* Grid de publicaciones */
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
  {filteredPublicaciones.map((item) => (
    <Card key={item.id} className="overflow-hidden">
      <div className="relative h-48 w-full">
        <img
          src={IMAGE_URL + item.imagen.split(";")[0] || "/placeholder.svg"}
          alt="Imagen de alquiler"
          className="object-cover w-full h-full"
        />
      </div>
      <CardContent className="p-4">
        <h3 className="font-medium text-sm line-clamp-2 mb-1">{item.titulo}</h3>
        <p className="text-sm text-gray-500">{item.fecha_evento}</p>
      </CardContent>
      <CardFooter className="p-4 pt-0 flex justify-between items-center">
        <span className="font-bold">{item.precio !== 0 ? "$" + item.precio : "Gratis"}</span>
        <div className="flex gap-2">
          <Button
            variant="outline"
            size="sm"
            onClick={() => {
              setPublication(item)
              setIsModalOpen(true)
            }}
          >
            Editar
          </Button>
          <Button
            variant="destructive"
            size="sm"
            onClick={() => {
              setSelectedPublicationId(item.id)
              setConfirmDialogOpen(true)
            }}
          >
            Eliminar
          </Button>
        </div>
      </CardFooter>
    </Card>
  ))
</div>

```

Ilustración 281 Publicacion jsx VII

```

    |   </Card>
    | })
  </div>

  /* Controles de paginación */
  {!searchQuery && (
    <div className="flex justify-center items-center mt-8 gap-2">
      <Button
        variant="outline"
        size="sm"
        onClick={() => handlePageChange(currentPage - 1)}
        disabled={currentPage === 1}
      >
        <ChevronLeft className="h-4 w-4" />
      </Button>
      <span className="text-sm">
        Página {currentPage} de {totalPages}
      </span>
      <Button
        variant="outline"
        size="sm"
        onClick={() => handlePageChange(currentPage + 1)}
        disabled={currentPage === totalPages}
      >
        <ChevronRight className="h-4 w-4" />
      </Button>
    </div>
  )}

  /* Modal para editar publicaciones existentes */
<ProductModal
  isOpen={isModalOpen}
  onClose={() => setIsModalOpen(false)}
  product={publication}
  onUpdate={handleUpdate}
/>

```

Ilustración 282 Publicacion jsx VIII

```

  /* Modal para agregar un nuevo producto */
<ProductModal
  isOpen={addModalOpen}
  onClose={() => setAddModalOpen(false)}
  onUpdate={handleProductAdded}
  isAddMode={true}
/>

  /* Diálogo de confirmación para eliminar */
<Dialog open={confirmDialogOpen} onChange={setConfirmDialogOpen}>
  <DialogContent className="sm:max-w-[425px]">
    <DialogTitle>
      <AlertTriangle className="h-5 w-5 text-red-500" />
      Confirmar eliminación
    </DialogTitle>
    <DialogHeader>
      <div className="py-4">
        <p className="text-gray-700">
          ¿Estás seguro que deseas eliminar esta publicación? Esta acción no se puede deshacer.
        </p>
      </div>
      <div className="flex justify-end gap-3">
        <Button variant="outline" onClick={() => setConfirmDialogOpen(false)}>
          Cancelar
        </Button>
        <Button variant="destructive" onClick={handleDelete} disabled={isDeleting}>
          {isDeleting ? "Eliminando..." : "Sí, eliminar"}
        </Button>
      </div>
    </DialogHeader>
  </DialogContent>
</Dialog>
  )}
</>
}

```

Ilustración 283 Publicacion jsx IX

Modal-publicaciones.jsx

Es el modal que se despliega a la hora de añadir o editar una publicación.

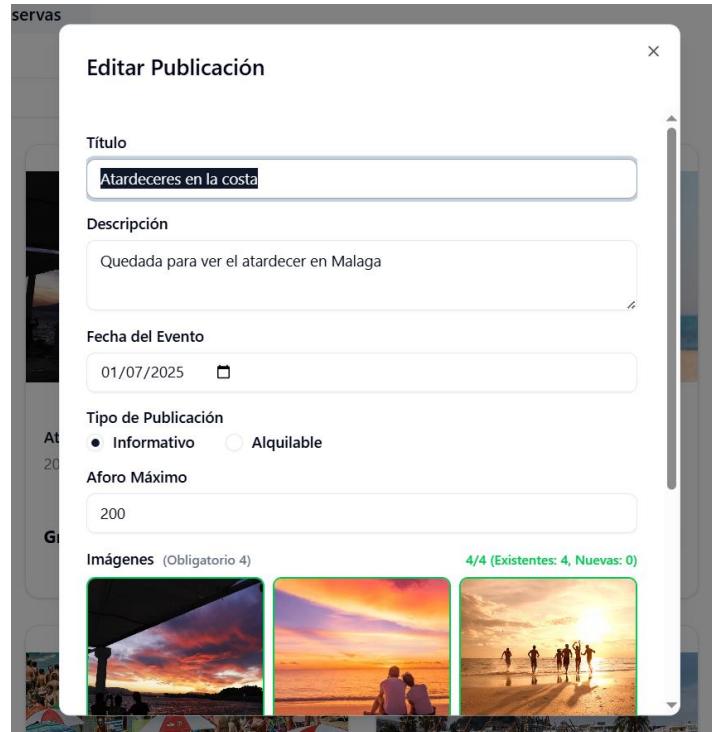


Ilustración 284 Modal-Publicaciones

Código – Modal-publicaciones

```
/*
 * @file modal-publicaciones.jsx
 * @description Modal para crear o editar publicaciones (informativas o alquilables) en el panel de administración.
 * Permite gestionar imágenes, validar campos y actualizar la información.
 */

import { useState, useEffect } from "react"
import { ImageIcon, Save, X, Plus } from "lucide-react"
import { Dialog, DialogContent, DialogHeader,DialogTitle } from "@/components/ui/dialog"
import { ScrollArea } from "@/components/ui/scroll-area"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Textarea } from "@/components/ui/textarea"
import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group"
import axios from "axios"
import { API_URL, IMAGE_URL } from "../../utilities/apirest"

/**
 * Modal para crear o editar una publicación.
 * @param {Object} props
 * @param {boolean} props.isOpen - Si el modal está abierto.
 * @param {Function} props.onClose - Función para cerrar el modal.
 * @param {Object} [props.product] - Publicación a editar (si existe).
 * @param {Function} props.onUpdate - Callback al actualizar.
 * @param {boolean} [props.isAddMode=false] - Si es modo añadir.
 * @returns {JSX.Element}
 */
export default function ProductModal({ isOpen, onClose, product, onUpdate, isAddMode = false }) {
  const [title, setTitle] = useState(product?.title || "")
  const [description, setDescription] = useState(product?.description || "")
  const [date, setDate] = useState(product?.date || new Date())
  const [capacity, setCapacity] = useState(product?.capacity || 0)
  const [isInformativo, setIsInformativo] = useState(product?.isInformativo || true)
  const [images, setImages] = useState(product?.images || [])
  const [error, setError] = useState(null)
```

Ilustración 285 Modal-publicaciones.jsx /

```

export default function ProductModal({ isOpen, onClose, product, onUpdate, isAddMode = false }) {
  /**
   * Estado del formulario de la publicación.
   * {[object, Function]}
   */
  const [formData, setFormData] = useState({
    título: "",
    descripción: "",
    fecha_evento: "",
    tipo: "informativo",
    precio: 0,
    aforo_maximo: 1,
  })
  /**
   * Estado de carga para el botón de guardar.
   * {[boolean, Function]}
   */
  const [isLoading, setIsLoading] = useState(false)
  /**
   * Estado de errores de validación.
   * {[Object, Function]}
   */
  const [errors, setErrors] = useState({})
  /**
   * Estado para previsualización de imágenes (existentes y nuevas).
   * {[Array, Function]}
   */
  const [imagePreviews, setImagePreviews] = useState([])
  /**
   * Estado para las nuevas imágenes seleccionadas.
   * {[Array, Function]}
   */
  const [newImages, setNewImages] = useState([])
  /**
   * Número de imágenes existentes.
   * {[number, Function]}
   */
}

```

Ilustración 286 Modal-publicaciones jsx II

```

const REQUIRED_IMAGES = 4
/**
 * Estado para controlar el cierre seguro del modal.
 * {[boolean, Function]}
 */
const [isClosing, setIsClosing] = useState(false)

/**
 * Efecto para inicializar el formulario y las imágenes al abrir el modal.
 */
useEffect(() => {
  if (product && !isAddMode) {
    // Formatear la fecha para el input de tipo date
    let fechaEvento = ""
    if (product.fecha_evento) {
      const date = new Date(product.fecha_evento)
      const year = date.getFullYear()
      const month = String(date.getMonth() + 1).padStart(2, "0")
      const day = String(date.getDate()).padStart(2, "0")
      fechaEvento = `${year}-${month}-${day}`
    }

    setFormData({
      ...product,
      fecha_evento: fechaEvento,
    })

    // Preparar las previsualizaciones de imágenes
    if (product.imagen) {
      const existingImages = product.imagen.split(";").filter(Boolean)
      setExistingImageCount(existingImages.length)

      const imageUrl = existingImages.map((img) => ({
        url: IMAGE_URL + img,
        isExisting: true,
        filename: img,
      }))
    }
  }
})

```

Ilustración 287 Modal-publicaciones jsx III

```

    } else {
      setImagePreviews([])
      setExistingImageCount(0)
    }
  } else {
    // Reset form for add mode
    setFormData({
      titulo: "",
      descripcion: "",
      fecha_evento: "",
      tipo: "informativo",
      precio: 0,
      aforo_maximo: 1,
    })
    setImagePreviews([])
    setExistingImageCount(0)
  }

  setNewImages([])
  setErrors({})
  setIsClosing(false)
}, [product, isOpen, isAddMode])

/**
 * Valida los campos del formulario y las imágenes.
 * @returns {boolean}
 */
const validateForm = () => {
  const newErrors = {}

  if (!formData.titulo?.trim()) {
    newErrors.titulo = "El título es obligatorio"
  }
  if (!formData.descripcion?.trim()) {
    newErrors.descripcion = "La descripción es obligatoria"
  }
}

```

Ilustración 288 Modal-publicaciones.jsx IV

```

    if (formData.tipo === "alquilable" && (!formData.precio || formData.precio <= 0)) {
      newErrors.precio = "El precio debe ser mayor que 0 para eventos alquilables"
    }
    if (!formData.aforo_maximo || formData.aforo_maximo <= 0) {
      newErrors.aforo_maximo = "El aforo máximo debe ser mayor que 0"
    }
    if (imagePreviews.length !== REQUIRED_IMAGES) {
      newErrors.images = `Debe tener exactamente ${REQUIRED_IMAGES} imágenes (actualmente tiene ${imagePreviews.length})`
    }
  }

  setErrors(newErrors)
  return Object.keys(newErrors).length === 0
}

/**
 * Maneja el cambio de los campos del formulario.
 * @param {React.ChangeEvent<HTMLInputElement|HTMLTextAreaElement>} e
 */
const handleInputChange = (e) => {
  const { name, value, type } = e.target

  if (type === "number") {
    setFormData({
      ...formData,
      [name]: Number.parseFloat(value) || 0,
    })
  } else {
    setFormData({
      ...formData,
      [name]: value,
    })
  }
}

```

Ilustración 289 Modal-publicaciones.jsx V

```

const handleImageChange = (e) => {
  if (e.target.files && e.target.files.length > 0) {
    const totalImages = imagePreviews.length + e.target.files.length

    if (totalImages > REQUIRED_IMAGES) {
      setErrors({
        ...errors,
        images: `Solo se permiten exactamente ${REQUIRED_IMAGES} imágenes (${imagePreviews.length} actuales)`,
      })
      return
    }

    const filesArray = Array.from(e.target.files)
    const newFilesArray = [...newImages, ...filesArray]
    setNewImages(newFilesArray)

    const newImageObjects = filesArray.map((file) => ({
      url: URL.createObjectURL(file),
      isExisting: false,
      file: file,
    }))

    setImagePreviews([...imagePreviews, ...newImageObjects])

    if (imagePreviews.length + newImageObjects.length === REQUIRED_IMAGES) {
      setErrors({
        ...errors,
        images: "",
      })
    } else {
      setErrors({
        ...errors,
        images: `Debe tener exactamente ${REQUIRED_IMAGES} imágenes (actualmente tiene ${imagePreviews.length + newImageObjects.length})`,
      })
    }
  }
}

```

Ilustración 290 Modal-publicaciones.jsx VI

```

/**
 * Elimina una imagen del array de previsualización.
 * @param {number} index
 */
const removeImage = (index) => {
  const updatedPreviews = [...imagePreviews]
  const removedImage = updatedPreviews[index]
  updatedPreviews.splice(index, 1)
  setImagePreviews(updatedPreviews)

  if (!removedImage.isExisting) {
    const fileToRemove = removedImage.file
    const newImagesUpdated = newImages.filter((file) => file !== fileToRemove)
    setNewImages(newImagesUpdated)
  }

  if (updatedPreviews.length !== REQUIRED_IMAGES) {
    setErrors({
      ...errors,
      images: `Debe tener exactamente ${REQUIRED_IMAGES} imágenes (actualmente tiene ${updatedPreviews.length})`,
    })
  } else {
    setErrors({
      ...errors,
      images: "",
    })
  }
}

/**
 * Cierra el modal de forma segura.
 */
const handleClose = () => {
  setIsClosing(true)
  onClose()
}

```

Ilustración 291 Modal-publicaciones.jsx VII

```

    /**
     * Envía el formulario para crear o actualizar la publicación.
     * @param {React.FormEvent<HTMLFormElement>} e
     */
    const handleSubmit = async (e) => {
      e.preventDefault()

      if (!validateForm()) return

      if (imagePreviews.length !== REQUIRED_IMAGES) {
        setErrors({
          ...errors,
          images: `Debe tener exactamente ${REQUIRED_IMAGES} imágenes (actualmente tiene ${imagePreviews.length})`,
        })
        return
      }

      try {
        setIsLoading(true)
        const formDataToSend = new FormData()
        Object.entries(formData).forEach(([key, value]) => {
          if (key !== "imagen" && value !== undefined) {
            const capitalizedKey = key.charAt(0).toUpperCase() + key.slice(1)
            formDataToSend.append(capitalizedKey, value)
          }
        })

        if (!isEditMode && product?.id) {
          formDataToSend.append("id_publicacion", product.id)
        }

        const existingImages = imagePreviews.filter((img) => img.isExisting)
        const newImageObjects = imagePreviews.filter((img) => !img.isExisting)

        newImageObjects.forEach((imgObj) => {
          formDataToSend.append("imagenes[]", imgObj.file)
        })

        existingImages.forEach((img) => {
      
```

Ilustración 292 Modal-publicaciones jsx VIII

```

      return (
        <Dialog open={isOpen} onOpenChange={isClosing ? onClose : handleClose}>
          <DialogContent className="sm:max-w-[600px] p-0 overflow-hidden flex flex-col max-h-[90vh]">
            <DialogHeader className="p-6 pb-2">
              <div className="flex items-center justify-between">
                <DialogTitle className="text-xl font-semibold">
                  {isEditMode ? "Añadir Producto" : "Editar Publicación"}
                </DialogTitle>
              </div>
            </DialogHeader>

            <form onSubmit={handleSubmit} className="flex flex-col flex-1 overflow-y-scroll">
              <ScrollArea className="flex-1 p-6">
                <div className="space-y-4 pb-4">
                  {errors.general && <div className="bg-red-50 p-3 rounded-md text-red-600 text-sm">{errors.general}</div>}

                  {/* Campo título */}
                  <div className="space-y-2">
                    <Label htmlFor="titulo">Título:</Label>
                    <Input
                      id="titulo"
                      name="titulo"
                      value={formData.titulo || ""}
                      onChange={handleInputChange}
                      className={errors.titulo ? "border-red-500" : ""}
                    />
                    {errors.titulo && <p className="text-red-500 text-xs">{errors.titulo}</p>}
                  </div>
                </div>
              </ScrollArea>
            </form>
          </DialogContent>
        </Dialog>
      )
    
```

Ilustración 293 Modal-publicaciones jsx IX

```

    {/* Campo descripción */}
    <div className="space-y-2">
      <Label htmlFor="descripcion">Descripción</Label>
      <Textarea
        id="descripcion"
        name="descripcion"
        value={formData.descripcion || ""}
        onChange={handleInputChange}
        rows={4}
        className={errors.descripcion ? "border-red-500" : ""}>
      />
      {errors.descripcion && <p className="text-red-500 text-xs">{errors.descripcion}</p>}
    </div>

    {/* campo fecha del evento */}
    <div className="space-y-2">
      <Label htmlFor="fecha_evento">Fecha del Evento</Label>
      <Input
        id="fecha_evento"
        name="fecha_evento"
        type="date"
        value={formData.fecha_evento || ""}
        onChange={handleInputChange}
        className={errors.fecha_evento ? "border-red-500" : ""}>
      />
      {errors.fecha_evento && <p className="text-red-500 text-xs">{errors.fecha_evento}</p>}
    </div>

    {/* selector de tipo de publicación */}
    <div className="space-y-2">
      <Label>Tipo de Publicación</Label>
      <RadioGroup
        value={formData.tipo || "informativo"}
        onValueChange={handleTipoChange}
        className="flex space-x-4">
        >
    
```

Ilustración 294 Modal-publicaciones.jsx X

```

    {/* Campo precio solo si es alquilable */}
    {formData.tipo === "alquilable" && (
      <div className="space-y-2">
        <Label htmlFor="precio">Precio</Label>
        <Input
          id="precio"
          name="precio"
          type="number"
          min="0"
          step="0.01"
          value={formData.precio || 0}
          onChange={handleInputChange}
          className={errors.precio ? "border-red-500" : ""}>
        />
        {errors.precio && <p className="text-red-500 text-xs">{errors.precio}</p>}
      </div>
    )}

    {/* Campo aforo máximo */}
    <div className="space-y-2">
      <Label htmlFor="aforo_maximo">Aforo Máximo</Label>
      <Input
        id="aforo_maximo"
        name="aforo_maximo"
        type="number"
        min="1"
        value={formData.aforo_maximo || ""}
        onChange={handleInputChange}
        className={errors.aforo_maximo ? "border-red-500" : ""}>
      />
      {errors.aforo_maximo && <p className="text-red-500 text-xs">{errors.aforo_maximo}</p>}
    </div>
  
```

Ilustración 295 Modal-publicaciones.jsx XI

```

/* Gestión de imágenes */


<div className="flex items-center justify-between">
    <label>
      | Imágenes <span className="text-xs text-gray-500">(Obligatorio {REQUIRED_IMAGES})</span>
    </label>
    <span>
      className={"text-xs ${imagePreviews.length === REQUIRED_IMAGES ? "text-green-500 font-semibold" : "text-red-500"} ${imagePreviews.length > REQUIRED_IMAGES ? "text-gray-500 font-semibold" : ""}}
      | ${imagePreviews.length}/{REQUIRED_IMAGES} (Existentes: {existingImagesCount}, Nuevas: {newImagesCount})
    </span>
  </div>
  {errors.images && <p className="text-red-500 text-xs mt-1">{errors.images}</p>}
<div className="grid grid-cols-3 gap-2 mt-2">
  {imagePreviews.map((image, index) => (
    <div key={index} className="relative group">
      <div>
        | className="aspect-square bg-gray-100 rounded-md overflow-hidden ${image.isExisting ? "border-2 border-gray-100" : "border-2 border-red-500"} ${image.url ? "bg-cover" : "bg-black bg-opacity-50 rounded-b-lg"} text-white"
        | <img src={(image.url || "/placeholder.svg") alt="Imagen ${index + 1}"} className="w-full h-full object-cover" />
        <div className="absolute bottom-0 left-0 right-0 bg-black bg-opacity-50 rounded-b-lg text-white">
          {image.isExisting ? "existente" : "Nueva"}
        </div>
        <button type="button" onClick={() => removeImage(index)} className="absolute top-1 right-1 bg-red-500 text-white rounded-full p-1 opacity-0 group-hover:opacity-100">
          <x className="h-4 w-4" />
        </button>
      </div>
    </div>
  )))
</div>


```

Ilustración 296 Modal-publicaciones.jsx XII

```

<label className="aspect-square bg-gray-200 rounded-md border-2 border-dashed border-gray-200 flex flex-col items-center justify-end gap-2 p-4 w-8 w-8 mb-2" />
  <ImageIcon className="h-8 w-8 text-gray-400 mb-2" />
  <span className="text-xs text-gray-500">Añadir imagen</span>
  <input type="file" accept="image/*" className="hidden" onChange={handleImageChange} multiple />
</label>
)
</div>
</div>
</scrollArea>

/* Botones de acción */
<div className="flex justify-end gap-3 p-4 border-t mt-auto bg-white">
  <Button variant="outline" type="button" onClick={handleClose}>
    Cancelar
  </Button>
  <Button type="submit" disabled={isLoading || imagePreviews.length !== REQUIRED_IMAGES} className="bg-green-600 hover:bg-green-700">
    {isLoading ? (
      <span className="flex items-center gap-2">
        <span className="h-4 w-4 border-2 border-white border-t-transparent rounded-full animate-spin" />
        {(isAddMode ? "Creando..." : "Guardando...")}
      </span>
    ) : (
      <span className="flex items-center gap-2">
        {(isAddMode ? <plus className="h-4 w-4" /> : <Save className="h-4 w-4" />)}
        {(isAddMode ? "Crear Producto" : "Guardar Cambios")}
      </span>
    )}
  </Button>
</div>
</form>
</DialogContent>
</Dialog>
}
```

Ilustración 297 Modal-publicaciones.jsx XIII

Reservas.jsx

Este componente es usado para mostrar las reservas en la sección de reservas del administrador, este incluye el componente modal-reservas que es el que se muestra a la hora de editar la reserva.

ID	USUARIO	ANUNCIO	FECHA	ESTADO	ACCIONES
61	Cristian Redondo Martínez	Tour histórico en barco	2025-07-07		<button>Editar</button> <button>Cancelar</button>
63	Cristian Redondo Martínez	Fiesta en catamarán	2025-07-02		<button>Editar</button> <button>Cancelar</button>
64	Cristian Redondo Martínez	Pesca deportiva con patrón	2025-07-04		<button>Editar</button> <button>Cancelar</button>
65	Manuelg Gonzalez	Alquiler equipo de pesca	2025-07-06		<button>Editar</button> <button>Cancelar</button>

Ilustración 298 Reservas

Código – reservas

```
/*
 * @file reservas.jsx
 * @description Vista de administración para listar, buscar, editar y cancelar reservas.
 * Incluye paginación, búsqueda y modales para edición/cancelación.
 */

import { useEffect, useState } from "react"
import { Search, ChevronLeft, ChevronRight } from "lucide-react"
import { Input } from "@/components/ui/input"
import { Button } from "@/components/ui/button"
import axios from "axios"
import { API_URL } from "../../utilities/apiRest"
import ReservasModal from "./modal-reservas"
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog"

/**
 * Componente principal para la gestión de reservas.
 * Permite buscar, pagination, editar y cancelar reservas.
 * @component
 * @returns {JSX.Element}
 */
export default function Reservas() {
  /**
   * Lista de reservas cargadas.
   * {[Array, Function]}
   */
  const [reservas, setReservas] = useState([])
  /**
   * Estado de carga.
   * {[boolean, Function]}
   */
  const [loading, setLoading] = useState(true)
}
```

Ilustración 299 Reservas.jsx /

```

    /**
     * Estado de carga.
     * {[boolean, Function]}
     */
    const [loading, setLoading] = useState(true)
    /**
     * Estado para mostrar el modal de edición.
     * {[boolean, Function]}
     */
    const [isModalOpen, setIsModalOpen] = useState(false)
    /**
     * Reserva seleccionada para editar.
     * {[Object|null, Function]}
     */
    const [selectedReserva, setSelectedReserva] = useState(null)
    /**
     * Estado para mostrar el diálogo de confirmación de cancelación.
     * {[boolean, Function]}
     */
    const [isAlertDialogOpen, setIsAlertDialogOpen] = useState(false)
    /**
     * ID de la reserva seleccionada para cancelar.
     * {[number|null, Function]}
     */
    const [reservaToDelete, setReservaToDelete] = useState(null)
    /**
     * Estado de búsqueda.
     * {((string|Function)[])}
     */
    const [searchQuery, setSearchQuery] = useState("")

    // Estados de paginación
    /**
     * Página actual.
     * {[number, Function]}
     */
    const [currentPage, setCurrentPage] = useState(1)
    /**

```

Ilustración 300 Reservas.jsx II

```

    /**
     * Total de páginas.
     * {[number, Function]}
     */
    const [totalPages, setTotalPages] = useState(1)

    /**
     * Efecto para cargar reservas al cambiar de página o cerrar el modal.
     */
    useEffect(() => {
      if (!isModalOpen)
        fetchReservas(currentPage)
    }, [currentPage, isModalOpen])

    /**
     * Obtiene las reservas paginadas desde la API.
     * @param {number} page
     */
    const fetchReservas = (page = 1) => {
      setLoading(true)
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}

      axios
        .post(API_URL + "api/reservasPaginadas", { pagina: page }, { headers })
        .then(response) => {
          if (response.data.success) {
            setReservas(response.data.data)
            setCurrentPage(response.data.page)
            setTotalPages(response.data.totalPages)
          } else {
            console.error("Error en la respuesta:", response.data)
          }
        }
        .catch(error) => {
          console.error("Error al cargar las reservas:", error)
        }
        .finally(() => {
          setLoading(false)
        })
    }

```

Ilustración 301 Reservas.jsx III

```

    /**
     * Maneja el click en el botón de editar reserva.
     * @param {Object} reservation
     */
    const handleEditClick = (reservation) => {
      // Obtener los datos completos de la reserva si es necesario
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}

      axios
        .get(`${API_URL}api/reservas/${reservation.id}`, { headers })
        .then((response) => {
          setSelectedReserva(response.data)
          setIsModalOpen(true)
        })
        .catch((error) => {
          console.error("Error al obtener detalles de la reserva:", error)
          // Fallback a los datos que ya tenemos
          setSelectedReserva(reservation)
          setIsModalOpen(true)
        })
    }

    /**
     * Cierra el modal de edición y refresca la lista si es necesario.
     * @param {boolean} refresh
     */
    const handleModalClose = (refresh = false) => {
      setIsModalOpen(false)
      setSelectedReserva(null)
      // Refresca la lista si se indica
      if (refresh) {
        fetchReservas(currentPage)
      }
    }
  
```

Ilustración 302 Reservas.jsx IV

```

    /**
     * Abre el diálogo de confirmación para cancelar una reserva.
     * @param {number} reservaId
     */
    const handleCancelReservation = (reservaId) => {
      setReservaToDelete(reservaId)
      setIsAlertDialogOpen(true)
    }

    /**
     * Confirma la cancelación de la reserva seleccionada.
     */
    const confirmCancelReservation = () => {
      if (!reservaToDelete) return

      setLoading(true)
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}

      axios
        .delete(`${API_URL}api/reservas/${reservaToDelete}`, { headers })
        .then((response) => {
          console.log("Reserva cancelada", response.data)
          setIsModalOpen(false)
          fetchReservas(currentPage)
        })
        .catch((error) => {
          console.error("Error al cancelar la reserva:", error)
        })
        .finally(() => {
          setLoading(false)
          setIsAlertDialogOpen(false)
          setReservaToDelete(null)
        })
    }
  
```

Ilustración 303 Reservas.jsx V

```

    /**
     * Cambia la página actual de la paginación.
     * @param {number} currentPage
     */
    const handlePageChange = (newPage) => {
      if (newPage >= 1 && newPage <= totalPages) {
        setCurrentPage(newPage)
      }
    }

    /**
     * Filtra las reservas según la búsqueda.
     */
    const filteredReservas = reservas.filter(
      (reservation) =>
        reservation.nombre_usuario.toLowerCase().includes(searchQuery.toLowerCase()) ||
        reservation.titulo_publicacion.toLowerCase().includes(searchQuery.toLowerCase()),
    )

    return (
      <>
        /* Loader mientras se cargan los datos */
        {loading ? (
          <div className="flex justify-center items-center h-64">
            <div className="w-12 h-12 border-4 border-blue-500 border-dashed rounded-full animate-spin"></div>
          </div>
        ) : (
          <>
            /* Barra de búsqueda */
            <div className="flex justify-between items-center mb-6">
              <div className="relative w-full max-w-md">
                <Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400" size={20} />
                <Input
                  type="search"
                  placeholder="Buscar reservas..."
                  className="pl-10 pr-4 py-2 w-full"
                  value={searchQuery}
                  onChange={(e) => setSearchQuery(e.target.value)}
                />
              </div>
            </div>
          </>
        ) : (
          <>
            /* Tabla de reservas */
            <div className="bg-white rounded-lg border overflow-sm-x-scroll overflow-y-hidden">
              <table className="min-w-full divide-y divide-gray-200">
                <thead className="bg-gray-50">
                  <tr>
                    <th
                      scope="col"
                      className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
                    >
                      ID
                    </th>
                    <th
                      scope="col"
                      className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
                    >
                      Usuario
                    </th>
                    <th
                      scope="col"
                      className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
                    >
                      Anuncio
                    </th>
                    <th
                      scope="col"
                      className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
                    >
                      Fecha
                    </th>
                    <th
                      scope="col"
                      className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
                    >
                      Estado
                    </th>
                    <th
                      scope="col"
                    >
                    </th>
                  </tr>
                </thead>
                <tbody>
                  {reservas.map((reserva) => (
                    <tr key={reserva.id}>
                      <td>{reserva.id}</td>
                      <td>{reserva.usuario}</td>
                      <td>{reserva.anuncio}</td>
                      <td>{reserva.fecha}</td>
                      <td>{reserva.estado}</td>
                      <td>{reserva.id}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </>
        ) : (
          <>
            /* Detalle de reserva */
            <div className="flex flex-col gap-4 p-4 bg-white rounded-lg border">
              <div>
                <h3>Reserva <span>#{reserva.id}</span></h3>
                <p>Usuario: <strong>{reserva.usuario}</strong></p>
                <p>Anuncio: <strong>{reserva.anuncio}</strong></p>
                <p>Fecha: <strong>{reserva.fecha}</strong></p>
                <p>Estado: <strong>{reserva.estado}</strong></p>
              </div>
              <div>
                <h4>Detalles</h4>
                <table>
                  <thead>
                    <tr>
                      <th>Nombre del Anuncio</th>
                      <th>Descripción del Anuncio</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
                      <td>{reserva.anuncio}</td>
                      <td>{reserva.descripcion}</td>
                    </tr>
                  </tbody>
                </table>
              </div>
            </div>
          </>
        )
      )
    )
  )
}

export default Reservas

```

Ilustración 304 Reservas.jsx VI

```

/* Tabla de reservas */
<div className="bg-white rounded-lg border overflow-sm-x-scroll overflow-y-hidden">
  <table className="min-w-full divide-y divide-gray-200">
    <thead className="bg-gray-50">
      <tr>
        <th
          scope="col"
          className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
        >
          ID
        </th>
        <th
          scope="col"
          className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
        >
          Usuario
        </th>
        <th
          scope="col"
          className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
        >
          Anuncio
        </th>
        <th
          scope="col"
          className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
        >
          Fecha
        </th>
        <th
          scope="col"
          className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider"
        >
          Estado
        </th>
        <th
          scope="col"
        >
        </th>
      </tr>
    </thead>
    <tbody>
      {reservas.map((reserva) => (
        <tr key={reserva.id}>
          <td>{reserva.id}</td>
          <td>{reserva.usuario}</td>
          <td>{reserva.anuncio}</td>
          <td>{reserva.fecha}</td>
          <td>{reserva.estado}</td>
          <td>{reserva.id}</td>
        </tr>
      ))}
    </tbody>
  </table>
</div>

```

Ilustración 305 Reservas.jsx VII

```

        |   Acciones
        |   </th>
        |</tr>
        |</thead>
        |<tbody className="bg-white divide-y divide-gray-200">
        |<filteredReservas.map((reservation, i) => (
        |  <tr key={i}>
        |    <td className="px-6 py-4 whitespace nowrap">
        |      <div className="text-sm font-medium text-gray-900">{reservation.id}</div>
        |    </td>
        |    <td className="px-6 py-4 whitespace nowrap">
        |      <div className="text-sm text-gray-900">{reservation.nombre_usuario}</div>
        |    </td>
        |    <td className="px-6 py-4 whitespace nowrap">
        |      <div className="text-sm text-gray-500 max-w-xs truncate">{reservation.titulo_publicacion}</div>
        |    </td>
        |    <td className="px-6 py-4 whitespace nowrap">
        |      <div className="text-sm text-gray-500">{reservation.fecha_reserva.split(" ")[0]}</div>
        |    </td>
        |    <td className="px-6 py-4 whitespace nowrap">
        |      <span
        |        className="px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${(
        |          reservation.estado === "pendiente"
        |            ? "bg-green-100 text-green-800"
        |            : reservation.estado === "pasada"
        |              ? "bg-yellow-100 text-yellow-800"
        |              : "bg-red-100 text-red-800"
        |        )}
        |      >
        |        {reservation.estado}
        |      </span>
        |    </td>
        |    <td className="px-6 py-4 whitespace nowrap text-right text-sm font-medium">
        |      <div className="flex justify-end gap-2">
        |        <button
        |          variant="outline"
        |          size="sm"
        |          onClick={() => handleEditClick(reservation)}
        |          disabled={reservation.estado === "pasada"}
        |          className={reservation.estado === "pasada" ? "opacity-50 cursor-not-allowed" : ""}>

```

Ilustración 306 Reservas jsx VIII

```

        |          disabled={reservation.estado === "pasada"}
        |          className={reservation.estado === "pasada" ? "opacity-50 cursor-not-allowed" : ""}>
        |        >
        |        Editar
        |      </button>
        |      <button variant="destructive" size="sm" onClick={() => handleCancelReservation(reservation.id)}>
        |        Cancelar
        |      </button>
        |    </div>
        |  </td>
        |</tr>
        |))
      </tbody>
    </table>
</div>

/* Controles de paginación */
{!searchQuery && (
  <div className="flex justify-center items-center mt-8 gap-2">
    <button
      variant="outline"
      size="sm"
      onClick={() => handlePageChange(currentPage - 1)}
      disabled={currentPage === 1}
    >
      <ChevronLeft className="h-4 w-4" />
    </button>
    <span className="text-sm">
      Página {currentPage} de {totalPages}
    </span>
    <button
      variant="outline"
      size="sm"
      onClick={() => handlePageChange(currentPage + 1)}
      disabled={currentPage === totalPages}
    >
      <ChevronRight className="h-4 w-4" />
    </button>
  </div>
)
}
```

Ilustración 307 Reservas jsx IX

```

    /* Modal para editar reservas */
    <ReservasModal
      isOpen={isModalOpen}
      setIsOpen={(open) => {
        if (!open) handleModalClose()
        else setIsModalOpen(open)
      }}
      reserva={selectedReserva}
      onCancelReservation={handleCancelReservation}
    />

    /* Diálogo de confirmación para cancelar */
    <AlertDialog open={isAlertDialogOpen} onChange={setIsAlertDialogOpen}>
      <AlertDialogContent>
        <AlertDialogHeader>
          <AlertDialogTitle>¿Estás seguro?</AlertDialogTitle>
          <AlertDialogDescription>
            Esta acción cancelará la reserva y no se puede deshacer. Se notificará al usuario sobre la cancelación.
          </AlertDialogDescription>
        </AlertDialogHeader>
        <AlertDialogFooter>
          <AlertDialogCancel>Cancelar</AlertDialogCancel>
          <AlertDialogAction onClick={confirmCancelReservation} className="bg-red-600 hover:bg-red-700">
            Confirmar cancelación
          </AlertDialogAction>
        </AlertDialogFooter>
      </AlertDialogContent>
    </AlertDialog>
  )
}

}

```

Ilustración 308 Reservas.jsx X

Modal-reservas.jsx

Es el modal encargado de mostrar los datos de la reserva a la hora de editarla desde el panel de administrador.

Este modal incluye los componentes de person-chooser.jsx y time-slider.jsx

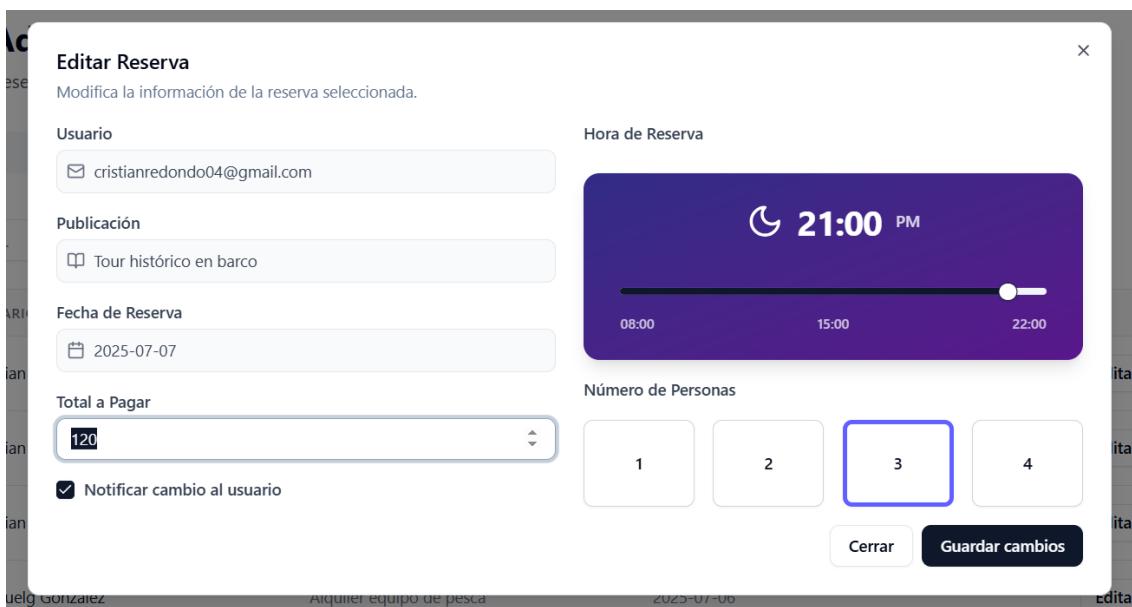


Ilustración 309 Modal-reservas

Código – Modal-reservas.jsx

```
/**
 * @file modal-reservas.jsx
 * @description Modal para crear o editar reservas en el panel de administración.
 * Permite modificar datos de la reserva, comprobar disponibilidad, aforo y notificar cambios al usuario.
 */
import { useState, useEffect } from "react"
import { Input } from "@/components/ui/input"
import { Button } from "@/components/ui/button"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select"
import { Checkbox } from "@/components/ui/checkbox"
import {
  Dialog,
 DialogContent,
  DialogHeader,
  DialogFooter,
 DialogTitle,
  DialogDescription,
} from "@/components/ui/dialog"
import axios from "axios"
import { API_URL } from "../../utilities/apirest"
import TimeSlider from "../../time-slider"
import SelectorPersonas from "../../person-chooser"
import { Label } from "@/components/ui/label"
import { AlertCircle, Mail, BookOpen, Calendar, AlertTriangle } from "lucide-react"

/**
 * Modal para crear o editar una reserva.
 * @component
 * @param {Object} props
 * @param {boolean} props.isOpen - Si el modal está abierto.
 * @param {Function} props.setisOpen - Función para abrir/cerrar el modal.
 * @param {Object|number} props.reserva - Reserva a editar o ID de la reserva.
 * @param {Function} props.onCancelReservation - Callback para cancelar la reserva.
 * @returns {JSX.Element}
 */
export default function ReservasModal({ isOpen, setIsOpen, reserva, onCancelReservation }) {
  /**
   * Estado de carga para el guardado.
   * {[boolean, Function]}
   */
}
```

Ilustración 310 Modal-reservas.jsx I

```
const [isLoading, setIsLoading] = useState(false)
/**
 * Estado del formulario de la reserva.
 * {[Object, Function]}
 */
const [formData, setFormData] = useState({
  usuario_id: "",
  publicacion_id: "",
  fecha_reserva: "",
  total_pagar: "",
  personas: ""
})
/**
 * Hora inicial de la reserva.
 * {[number, Function]}
 */
const [horaInicio, setHoraInicio] = useState(0)
/**
 * Hora seleccionada.
 * {[number, Function]}
 */
const [hora, setHora] = useState(15)
/**
 * Disponibilidad de la hora seleccionada.
 * {[boolean, Function]}
 */
const [disponibilidadHora, setDisponibilidadHora] = useState(true)
/**
 * Indica si la hora ya ha pasado.
 * {[boolean, Function]}
 */
const [horaPasada, setHoraPasada] = useState(false)
/**
 * Personas disponibles para reservar.
 * {[number, Function]}
 */
const [personasDisponibles, setPersonasDisponibles] = useState(4)
/**
 * Si se debe notificar al usuario del cambio.
 * {[boolean, Function]}
 */
```

Ilustración 311 Modal-reservas.jsx II

```

const [notificarCambio, setNotificarCambio] = useState(true)
/**
 * Email del usuario de la reserva.
 * ((string|Function)[])
 */
const [emailUsuario, setEmailUsuario] = useState("")
/**
 * Título de la publicación reservada.
 * ((string|Function)[])
 */
const [publicacionTitulo, setPublicacionTitulo] = useState("")
/**
 * Mensaje informativo.
 * ((string|Function)[])
 */
const [mensaje, setMensaje] = useState("")
/**
 * Estado para mostrar el diálogo de confirmación de guardado.
 * {[boolean, Function]}
 */
const [confirmSaveDialogOpen, setConfirmSaveDialogOpen] = useState(false)

/**
 * Efecto para cargar los datos de la reserva al abrir el modal.
 */
useEffect(() => {
  if (!reserva || !isopen) return

  const fetchData = async () => {
    try {
      let currentReserva = reserva
      // Si sólo tenemos un ID y no todos los datos de reserva
      if (typeof reserva === "number") {
        const res = await axios.get(`${API_URL}api/reservas/${reserva}`)
        currentReserva = res.data
      }
    }
  }
})

```

Ilustración 312 Modal-reservas.jsx III

```

// Fecha y hora
const formattedDate = currentReserva.fecha_reserva?.split(" ")[0] || ""
const formattedTime = Number.parseInt(currentReserva.fecha_reserva?.split(" ")[1]? .split(":")[0]) || 15
setHora(formattedTime)
setHoraInicio(formattedTime)

// Actualiza form
setFormData({
  id: currentReserva.id,
  usuario_id: currentReserva.usuario_id?.toString() || "",
  publicacion_id: currentReserva.publicacion_id?.toString() || "",
  fecha_reserva: formattedDate,
  total_pagar: currentReserva.total_pagar?.toString() || "",
  personas: currentReserva.personas?.toString() || "",
})

// Establecer el número de personas seleccionadas
if (currentReserva.personas) {
  handlePersonasChange(currentReserva.personas)
}

// Email
if (currentReserva.email_usuario) {
  setEmailUsuario(currentReserva.email_usuario)
} else if (currentReserva.usuario_id) {
  fetchUserEmail(currentReserva.usuario_id)
}

// Publicación
if (currentReserva.titulo_publicacion) {
  setPublicacionTitulo(currentReserva.titulo_publicacion)
} else if (currentReserva.publicacion_id) {
  fetchPublicationTitle(currentReserva.publicacion_id)
}
checkCapacity(currentReserva.publicacion_id)
} catch (error) {
  console.error("Error al cargar datos de la reserva", error)
}
}

```

Ilustración 313 Modal-reservas.jsx IV

```

    /**
     * Efecto para comprobar si la hora seleccionada está disponible y si ya ha pasado.
     */
    useEffect(() => {
      if (!formData.fecha_reserva || !formData.publicacion_id) return
      const checkHora = async () => {
        try {
          // Verificar si la hora ya ha pasado
          const horaRes = await axios.get(API_URL + "api/horaFecha")
          if (horaRes.data.fecha === formData.fecha_reserva) {
            const horaActual = Number.parseInt(horaRes.data.hora.split(":")[0], 10)
            setHoraPasada(horaActual >= hora)
          } else {
            setHoraPasada(false)
          }
          if (hora != horaInicio) checkTimeAvailability(formData.publicacion_id, hora)
          else setDisponibilidadHora(true)
        } catch (error) {
          console.error("Error al comprobar hora y disponibilidad:", error)
        }
      }
      checkHora()
    }, [hora, formData.fecha_reserva, formData.publicacion_id])

    /**
     * Comprueba la disponibilidad de la hora seleccionada para la publicación.
     * @param {string|number} publicacionId
     * @param {number} hora
     */
    const checkTimeAvailability = (publicacionId, hora) => {
      if (!publicacionId) return
      const url = API_URL + "api/disponibilidadReserva"
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}

```

Ilustración 314 Modal-reservas.jsx V

```

      axios
        .post(url, { idPublicacion: publicacionId, horaReserva: hora + ":00" }, { headers })
        .then((response) => {
          setDisponibilidadHora(response.data.disponible)
        })
        .catch((error) => {
          console.error("Error al comprobar disponibilidad:", error)
        })
    }

    /**
     * Consulta la capacidad máxima disponible para la publicación.
     * @param {string|number} publicacionId
     */
    const checkCapacity = (publicacionId) => {
      if (!publicacionId) return

      const url = API_URL + "api/capacidadDisponible"
      const token = localStorage.getItem("authToken")
      const headers = token ? { Authorization: `Bearer ${token}` } : {}

      axios
        .post(url, { idPublicacion: publicacionId }, { headers })
        .then((response) => {
          setPersonasDisponibles(response.data.max_reservables)
        })
        .catch((error) => {
          console.error("Error al comprobar capacidad:", error)
        })
    }

```

Ilustración 315 Modal-reservas.jsx VI

```
/**  
 * Obtiene el email del usuario por su ID.  
 * @param {string|number} userId  
 */  
const fetchUserEmail = (userId) => {  
  const token = localStorage.getItem("authToken")  
  const headers = token ? { Authorization: `Bearer ${token}` } : {}  
  
  axios  
    .get(`${API_URL}api/usuarios/${userId}`, { headers })  
    .then((response) => {  
      if (response.data && response.data.Email) {  
        setEmailUsuario(response.data.Email)  
      }  
    })  
    .catch((error) => {  
      console.error("Error al obtener email de usuario:", error)  
    })  
}  
  
/**  
 * Obtiene el título de la publicación por su ID.  
 * @param {string|number} publicationId  
 */  
const fetchPublicationTitle = (publicationId) => {  
  axios  
    .get(`${API_URL}api/publicaciones/${publicationId}`)  
    .then((response) => {  
      if (response.data && response.data.titulo) {  
        setPublicacionTitulo(response.data.titulo)  
      }  
    })  
    .catch((error) => {  
      console.error("Error al obtener título de publicación:", error)  
    })  
}
```

Ilustración 316 Modal-reservas.jsx VII

```
/**  
 * Maneja el cambio de los campos del formulario.  
 * @param {React.ChangeEvent<HTMLInputElement>} e  
 */  
const handleChange = (e) => {  
  const { name, value } = e.target  
  setFormData((prevData) => ({ ...prevData, [name]: value }))  
}  
  
/**  
 * Cambia el número de personas seleccionadas.  
 * @param {number|string} value  
 */  
const handlePersonasChange = (value) => {  
  setFormData((prevData) => ({ ...prevData, personas: value }))  
}  
  
/**  
 * Abre el diálogo de confirmación para guardar cambios.  
 */  
const handleSave = () => {  
  setConfirmSaveDialogOpen(true)  
}  
  
/**  
 * Intercambia fechas de reserva si la hora no está disponible.  
 * @param {string|number} publicationId  
 * @param {string} fecha  
 */
```

Ilustración 317 Modal-reservas.jsx VIII

```

const intercambiarFechas = (publicationId, fecha) => {
  const token = localStorage.getItem("authToken")
  const headers = token ? { Authorization: `Bearer ${token}` } : {}
  axios.post(`${API_URL}api/intercambiarFechas`,
    { id: publicationId, nueva_fecha_reserva: fecha, correo: notificarCambio },
    { headers },
  )
  .then((response) => {
    // Manejo de respuesta si es necesario
  })
  .catch((error) => {
    console.error("Error al intercambiar fechas:", error)
  })
}

/**
 * Envía un email al usuario notificando el cambio de hora.
 * @param {string} email
 * @param {string} nuevaHora
 */
function enviarEmailCambioHora(email, nuevaHora) {
  if (!email || !notificarCambio) return

  const token = localStorage.getItem("authToken")
  const headers = token ? { Authorization: `Bearer ${token}` } : {}

  // Formatear la hora para el mensaje
  const horaFormatada = nuevaHora.split(" ")[1].split(":")[0]

  // Crear mensaje para el usuario
  const mensaje = `Se ha modificado la hora de tu reserva para ${publicacionTitulo}. La nueva hora es: ${horaFormatada}:00`

  axios
    .post(`${API_URL}api/mailTo`, { email: email, mensaje: mensaje }, { headers })
    .then((response) => {
      // Email enviado correctamente
    })
    .catch((error) => {
      // Manejo de error
    })
}

```

Ilustración 318 Modal-reservas.jsx IX

```

/**
 * Confirma y guarda los cambios de la reserva.
 */
const confirmSave = () => {
  setIsLoading(true)
  const token = localStorage.getItem("authToken")
  const headers = token ? { Authorization: `Bearer ${token}` } : {}
  const horaFormatada = hora.toString().padStart(2, "0")
  const dataToSubmit = {
    ...formData,
    id: formData.id,
    usuario_id: Number.parseInt(formData.usuario_id),
    publicacion_id: Number.parseInt(formData.publicacion_id),
    total_pagar: Number.parseFloat(formData.total_pagar),
    personas: Number.parseInt(formData.personas),
    fecha_reserva: `${formData.fecha_reserva} ${horaFormatada}:00:00`,
    notificar: notificarCambio,
  }

  // Verificar si la hora ha cambiado
  const horaHaCambiado = hora !== horaInicio

  // Si no hay disponibilidad, intercambiar fechas
  if (!disponibilidadHora) {
    intercambiarFechas(formData.id, dataToSubmit.fecha_reserva)
  }
  // Si la hora ha cambiado y el usuario quiere ser notificado, enviar email
  else if (horaHaCambiado && notificarCambio) {
    enviarEmailCambioHora(emailUsuario, dataToSubmit.fecha_reserva)
  }

  setDisponibilidadHora(true)
  const url = `${API_URL}api/actualizarReservas`
  axios
    .post(url, dataToSubmit, { headers })
    .then((response) => {
      setIsOpen(false)
    })
    .catch((error) => {
      // Manejo de error
    })
}

```

Ilustración 319 Modal-reservas.jsx X

```

return (
  <Dialog open={isOpen} onOpenChange={(open) => setIsOpen(open)}>
    <DialogContent className="w-[1000px] !max-w-none">
      <DialogTitle>{reserva ? "Editar Reserva" : "Nueva Reserva"}</DialogTitle>
      <DialogDescription>
        {reserva ? "Modifica la información de la reserva seleccionada." : "Crea una nueva reserva."}
      </DialogDescription>
    </DialogContent>
  </Dialog>
  <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
    <div className="space-y-4">
      <div>
        <label className="text-sm font-medium text-gray-700 mb-1 block">Usuario</label>
        <div className="flex items-center p-2 bg-gray-50 rounded-md border border-gray-200">
          <Mail className="h-4 w-4 text-gray-500 mr-2" />
          <span className="text-sm text-gray-700">{emailUsuario}</span>
        </div>
      </div>
      <div>
        <label className="text-sm font-medium text-gray-700 mb-1 block">Publicación</label>
        {publicacionTitulo ? (
          <div className="flex items-center p-2 bg-gray-50 rounded-md border border-gray-200">
            <BookOpen className="h-4 w-4 text-gray-500 mr-2" />
            <span className="text-sm text-gray-700">{publicacionTitulo}</span>
          </div>
        ) : (
          <Input
            name="publicacion_id"
            type="number"
            value={formData.publicacion_id}
            onChange={handleChange}
            placeholder="ID de publicación"
          />
        )}
      </div>
    </div>
  </div>
)

```

Ilustración 320 Modal-reservas.jsx XI

```

<label className="text-sm font-medium text-gray-700 mb-1 block">Fecha de Reserva</label>
<div className="flex items-center p-2 bg-gray-50 rounded-md border border-gray-200">
  <Calendar className="h-4 w-4 text-gray-500 mr-2" />
  <span className="text-sm text-gray-700">{formData.fecha_reserva}</span>
</div>

<div>
  <label className="text-sm font-medium text-gray-700 mb-1 block">Total a Pagar</label>
  <input name="total_pagar" type="number" value={formData.total_pagar} onChange={handleChange} />
</div>

<div className="flex items-center space-x-2 mt-4">
  <Checkbox id="notificar" checked={notificarCambio} onChange={setNotificarCambio} />
  <Label htmlFor="notificar" className="text-sm font-medium text-gray-700">
    Notificar cambio al usuario
  </Label>
</div>
</div>

<div className="space-y-4">
  <div>
    <label className="text-sm font-medium text-gray-700 mb-1 block">Hora de Reserva</label>
    {reserva && (
      <TimeSlider
        id={formData.publicacion_id}
        hora={hora}
        setHora={setHora}
        disponible={disponibilidadHora}
        horaPasada={horaPasada}
      />
    )}
  </div>
  {!disponibilidadHora && (
    <div className="flex items-center gap-2 mt-2 p-3 bg-yellow-50 border border-yellow-200 rounded-md text-yellow-700">
      <AlertCircle className="h-4 w-4" />
      <div>
        <p>Atención</p>
        <p>Esta hora ya está reservada la hora será intercambiada.</p>
      </div>
    </div>
  )}
</div>

```

Ilustración 321 Modal-reservas.jsx XII

```

{horaPasada && (
  <div className="flex items-center mt-2 text-red-500 text-sm">
    <AlertCircle className="h-4 w-4 mr-1" />
    <span>Esta hora ya ha pasado</span>
  </div>
)}

</div>

<div>
  <label className="text-sm font-medium text-gray-700 mb-1 block">Número de Personas</label>
  {reserva ? (
    personasDisponibles > 0 ? (
      <SelectorPersonas
        personasDisponibles={personasDisponibles}
        setPersonas={handlePersonasChange}
        selected={formData.personas?.toString()}>
    ) : (
      <div className="flex items-center p-3 bg-red-50 rounded-md border border-red-200 text-red-700">
        <AlertCircle className="h-4 w-4 mr-2" />
        <span>Aforo completo. No es posible cambiar el número de personas.</span>
      </div>
    )
  ) : (
    <Select
      value={formData.personas}
      onChange={(value) => setFormData((prevData) => ({ ...prevData, personas: value }))}>
      <SelectTrigger className="w-full">
        <SelectValue placeholder="Selecciona número de personas" />
      </SelectTrigger>
      <SelectContent>
        {[1, 2, 3, 4].map((n) => (
          <SelectItem key={n} value={n.toString()}>
            {n} persona{n > 1 ? "s" : ""}
          </SelectItem>
        )))
      </SelectContent>
    </Select>
  )
}

```

Ilustración 322 Modal-reservas.jsx XIII

```

</div>
{mensaje && (
  <div className="mt-2 p-2 bg-blue-50 border border-blue-200 rounded-md text-blue-700 text-sm">
    <span>{mensaje}</span>
  </div>
)
}

</div>
</div>

<DialogFooter className="flex flex-col sm:flex-row sm:justify-end gap-2">
  <div className="flex gap-2">
    {isLoading ? (
      <Button disabled>Guardando...</Button>
    ) : (
      <>
        <Button variant="outline" onClick={() => setIsOpen(false)}>
          Cerrar
        </Button>
        <Button onClick={handleSave} disabled={reserva && horaPasada}>
          Guardar cambios
        </Button>
      </>
    )
  </div>
</DialogFooter>
</DialogContent>
/* Dialogo de confirmación para guardar */
<Dialog open={confirmSaveDialogOpen} onClose={setConfirmSaveDialogOpen}>
  <DialogContent className="sm:max-w-[425px]">
    <DialogHeader>
      <DialogTitle className="flex items-center gap-2">
        <AlertTriangle className="h-5 w-5 text-yellow-500" />
        Confirmar cambios
      </DialogTitle>
    </DialogHeader>
    <div className="py-4">
      <p className="text-gray-700">
        ¿Estás seguro que deseas guardar los cambios en esta reserva?
        {!disponibilidadHora && (
          <span className="block mt-2 font-medium text-yellow-600">

```

Ilustración 323 Modal-reservas.jsx XIV

```

        <span className="block mt-2 font-medium text-yellow-600">
            Recuerda que la hora seleccionada ya está reservada y será intercambiada.
        </span>
    )}
    </p>
</div>
<div className="flex justify-end gap-3">
    <Button variant="outline" onClick={() => setConfirmSaveDialogOpen(false)}>
        Cancelar
    </Button>
    <Button onClick={confirmSave} disabled={isLoading}>
        {isLoading ? "Guardando..." : "Sí, guardar cambios"}
    </Button>
</div>
</DialogContent>
</Dialog>
</Dialog>
}
}

```

Ilustración 324 Modal-reservas.jsx XV

Modal-usuarios-perfil.jsx

Es un modal empleado para editar los usuarios desde el apartado de perfil



Ilustración 325 Moda-usuarios-perfil

Código – Modal-usuarios-perfil

```

import React, { useState, useEffect } from 'react';
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { Dialog,DialogContent, DialogHeader, DialogFooter,DialogTitle, DialogDescription } from "@/components/ui/dialog";
import axios from "axios";
import { API_URL } from '../../../../../utilities/apirest';

/**
 * Modal para crear o editar usuarios desde el panel de administración.
 * Permite modificar datos personales y el rol del usuario.
 * @component
 * @param {Object} props
 * @param {boolean} props.isOpen - Si el modal está abierto.
 * @param {Function} props.setIsOpen - Función para abrir/cerrar el modal.
 * @param {Object|null} props.user - Usuario a editar (si existe) o null para crear.
 * @returns {JSX.Element}
 */
export default function UsuariosModal({ isOpen, setIsOpen, user }) {
    /**
     * Estado de carga para el guardado.
     * {[boolean, Function]}
     */
    const [isLoading, setIsLoading] = useState(false)
    /**
     * Estado del formulario del usuario.
     * {[Object, Function]}
     */
    const [formData, setFormData] = useState({
        Nombre: '',
        Apellidos: '',
        Email: '',
        Fecha_nacimiento: '',
        Tipo: ''
    });
}

```

Ilustración 326 Modal-usuarios-perfil.jsx I

```

    /**
     * Efecto para cargar los datos del usuario al abrir el modal.
     */
    useEffect(() => {
        if (user) {
            setFormData({
                id_usuario: user.id,
                Nombre: user.Nombre,
                Apellidos: user.Apellidos,
                Email: user.Email,
                Fecha_nacimiento: user.Fecha_nacimiento,
                Tipo: user.Tipo
            });
        }
    }, [user]);

    /**
     * Maneja el cambio de los campos del formulario.
     * @param {React.ChangeEvent<HTMLInputElement>} e
     */
    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData((prevData) => ({ ...prevData, [name]: value }));
    };
}

```

Ilustración 327 Modal-usuarios-perfil.jsx II

```

    /**
     * Guarda los cambios del usuario (actualiza o crea).
     */
    const handleSave = () => {
        setIsLoading(true)
        const token = localStorage.getItem("authToken");
        const headers = token ? { Authorization: `Bearer ${token}` } : {};
        if (user != null) {
            axios.post(`${API_URL}api/usuarios/actualizar`, formData, { headers })
                .then((response) => {
                    console.log('Usuario actualizado', response.data);
                    setIsOpen(false); // Cerrar el modal
                })
                .catch((error) => {
                    console.error('Error al actualizar usuario', error);
                }).finally(() => {
                    setIsLoading(false);
                });
        } else {
            axios.post(`${API_URL}api/usuarios`, formData, { headers })
                .then((response) => {
                    console.log('Usuario guardado', response.data);
                    setIsOpen(false); // Cerrar el modal
                })
                .catch((error) => {
                    console.error('Error al actualizar usuario', error);
                }).finally(() => {
                    setIsLoading(false);
                });
        }
    };
}

```

Ilustración 328 Modal-usuarios-perfil.jsx III

```

return (
    <Dialog open={isOpen} onOpenChange={(open) => setIsOpen(open)}>
        <DialogContent>
            <DialogTitle>Editar Usuario</DialogTitle>
            <DialogDescription>Modifica la información del usuario seleccionado.</DialogDescription>
        </DialogTitle>
        <div className="space-y-4">
            {/* Campo Nombre */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Nombre</label>
            <Input
                label="Nombre"
                name="Nombre"
                value={formData.Nombre}
                onChange={handleChange}
            />
            {/* Campo Apellidos */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Apellidos</label>
            <Input
                label="Apellidos"
                name="Apellidos"
                value={formData.Apellidos}
                onChange={handleChange}
            />
            {/* Campo Email */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Email</label>
            <Input
                label="Email"
                name="Email"
                value={formData.Email}
                onChange={handleChange}
            />
            {/* Campo Fecha de Nacimiento */}
            <label className="text-sm font-medium text-gray-700 mb-1 block">Fecha de Nacimiento</label>
        </div>
    </DialogContent>

```

Ilustración 329 Modal-usuarios-perfil.jsx IV

```

<input
    label="Fecha de Nacimiento"
    name="Fecha_nacimiento"
    type="date"
    value={formData.Fecha_nacimiento}
    onChange={handlechange}
/>
{/* Campo Contraseña solo al crear usuario */}
{user === null && (
  <>
    <label className="text-sm font-medium text-gray-700 mb-1 block">Contraseña</label>
    <Input
      label="Contraseña"
      name="password"
      type="password"
      onChange={handlechange}
    />
  </>
)}
 {/* selector de rol */}
<div>
  <label className="text-sm font-medium text-gray-700 mb-1 block">Rol</label>
  <Select
    value={formData.Tipo}
    onValueChange={(value) =>
      setFormData((prevData) => ({ ...prevData, Tipo: value }))
    }
  >
    <SelectTrigger className="w-full">
      <SelectValue placeholder="Selecciona un rol" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="admin">Admin</SelectItem>
      <SelectItem value="usuario">Usuario</SelectItem>
    </SelectContent>
  </select>
</div>
</div>

```

Ilustración 330 Modal-usuarios-perfil.jsx V

```

<DialogFooter>
  {isLoading ? (
    <Button variant="outline">Guardando...</Button>
  ) : (
    <>
      <Button onClick={() => setIsOpen(false)}>Cancelar</Button>
      <Button variant="outline" onClick={handleSave}>Guardar cambios</Button>
    </>
  )}
</DialogFooter>
</DialogContent >
</Dialog >
};

```

Ilustración 331 Modal-usuarios-perfil.jsx VI

7. Despliegue

Servidor

En primer lugar, tendremos que tener comprado un hosting el cual disponga de gestión de base de datos y archivos.

El hosting que he usado en mi caso es <https://atlas.profesionalhosting.com> proporcionado por el centro de estudio.

Para desplegarlo, lo primero que haremos será crear un nuevo sitio

1 items total

Nombre del dominio	Estado	Uso de disco	Tráfico
manu.cicloflorenciopintado.es	Activo	420.9 MB	0 MB/mes

Ilustración 332 Despliegue I

Al cual le asignaremos el nombre de nuestro proyecto final

Nombre del subdominio *

marinarent . manu.cicloflorenciopintado.es

Introduzca * para crear un subdominio wildcard.

Configuración de hosting

Raíz del documento *

/ marinarent.manu.cicloflorenci

Ruta al directorio principal del sitio web.

* Campos obligatorios

ACEPTAR

Cancelar

Ilustración 333 Despliegue II

Una vez creado nuestro sitio, lo que haremos será crear la base de datos, para ello haremos clic en base de datos

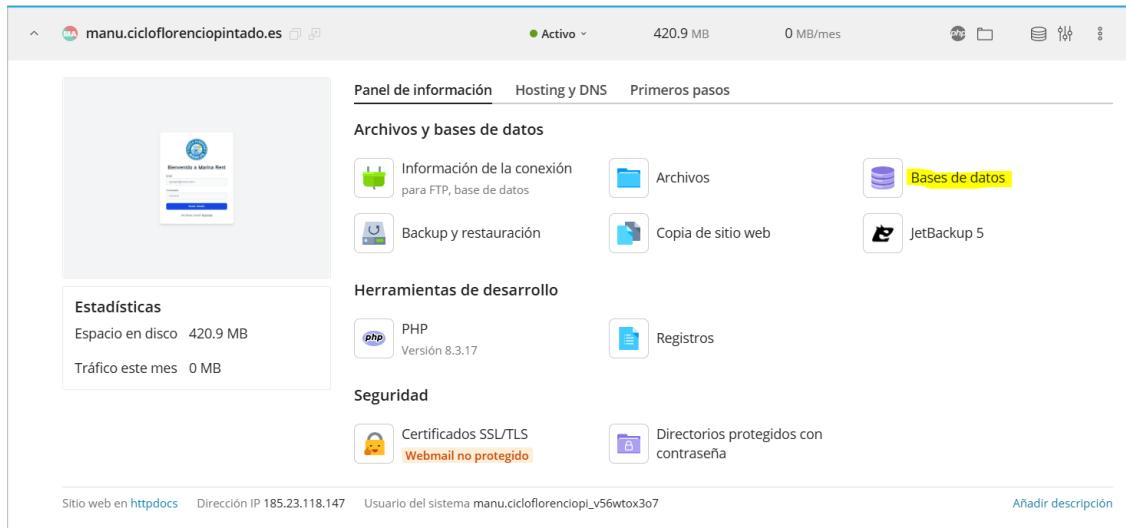


Ilustración 334 Despliegue III

Daremos clic en añadir base de datos

Bases de datos >

Bases de datos para [manu.cicloflorenciopintado.es](#)

✓ Información: [atlas.profesionalhosting.com - s53.profesionalhosting.com](#)

ATENCION:

Las copias de seguridad creadas por el administrador no computarán en su espacio contratado.

** Proteja su acceso a PLESK con un TWO FACTOR haciendo click [>>>AQUI<<<](#)

Aquí puede crear una base de datos nueva o administrar bases de datos existentes.

[+ Añadir base de datos](#)

Ilustración 335 Despliegue IV

Y rellenaremos los datos de la nueva base de datos seleccionando en todo momento para el hosting que vamos a crear la base de datos.

** Proteja su acceso a PLESK con un TWO FACTOR haciendo click [>>>AQUI<<<](#)

General

Nombre de la base de datos *

Servidor de bases de datos

Sitio relacionado

Ilustración 336 Despliegue V

A continuación daremos clic en el phpMyAdmin para gestionar la base de datos



Ilustración 337 Despliegue VI

Y daremos clic en importar base de datos

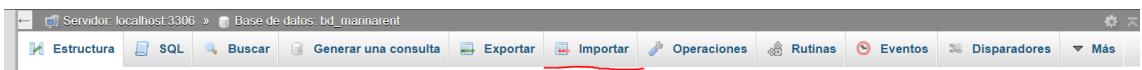


Ilustración 338 Despliegue VII

Seleccionaremos la exportación de la base de datos subida al repositorio y daremos en importar, importando así todos los datos de esta a su vez.

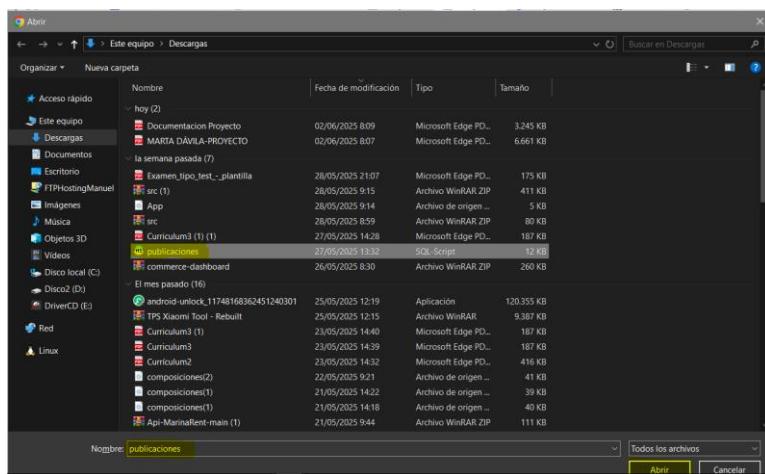


Ilustración 339 Despliegue VIII

Y aquí tenemos la prueba de que se ha importado correctamente

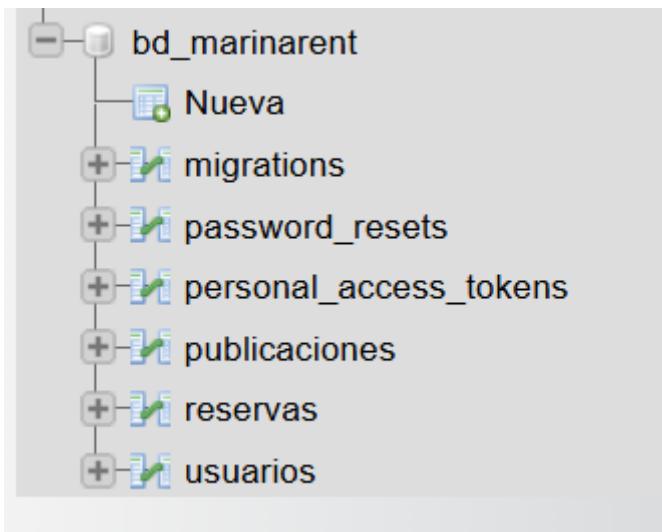


Ilustración 340 Despliegue IX

Para desplegar el servidor lo que haremos será dar clic en archivos

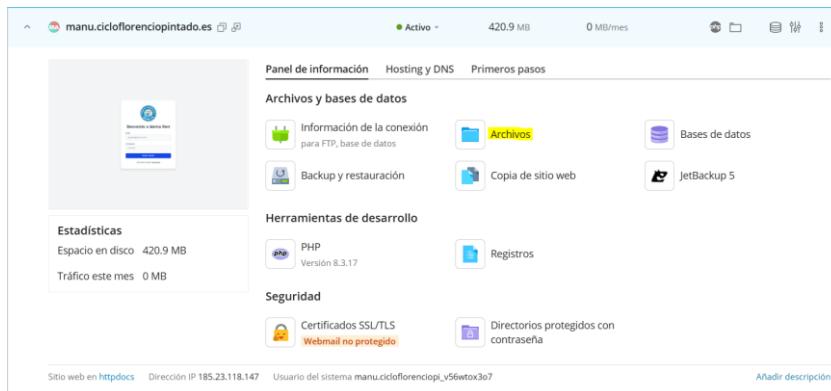


Ilustración 341 Despliegue X

Y una vez dentro lo que haremos será dar clic en httpdocs donde tendremos que crear la siguiente carpeta llamada laravel.

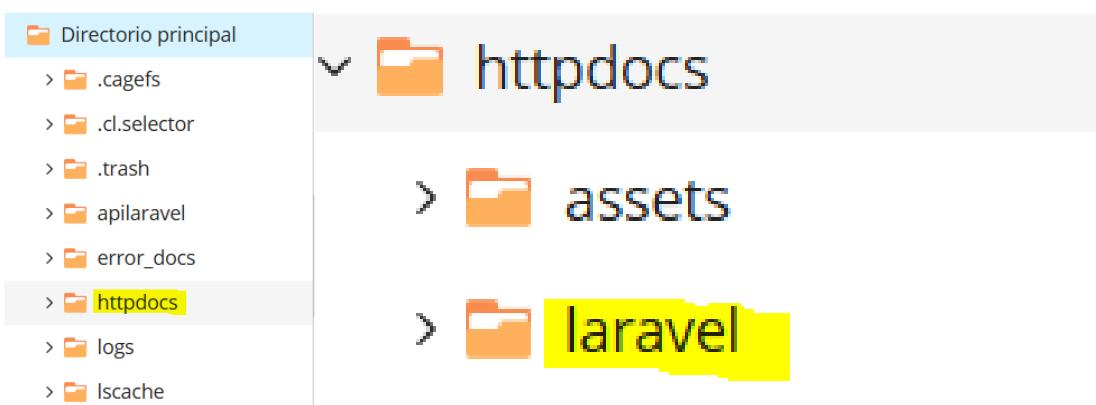


Ilustración 342 Despliegue XI

Dentro de laravel daremos clic en el + y a continuación en cargar directorio donde seleccionaremos las carpetas que queremos subir al hosting.

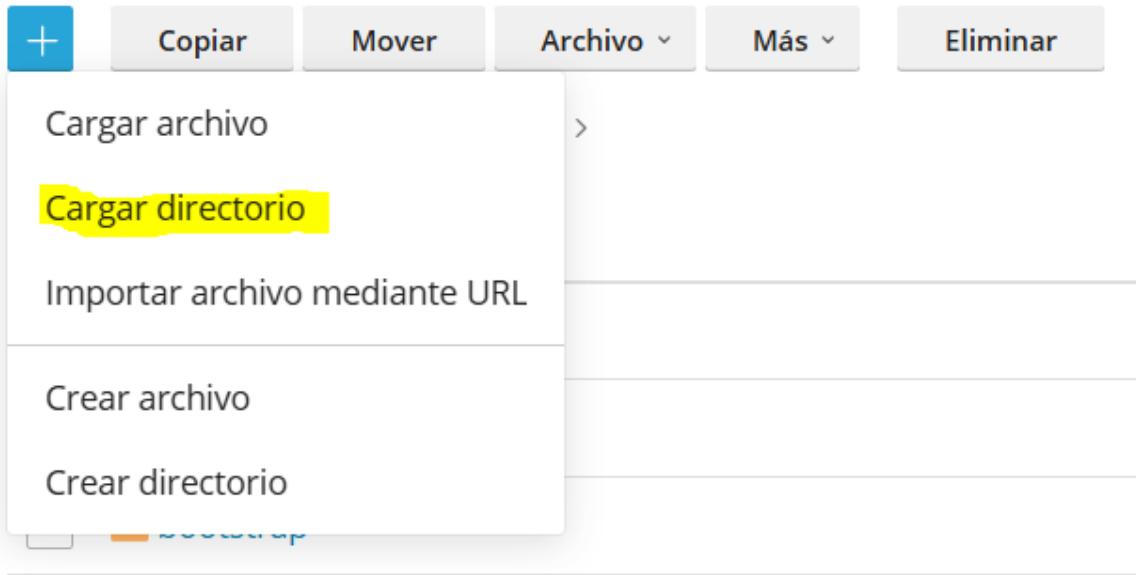


Ilustración 343 Despliegue XII

Para el funcionamiento de la API restFull en el hosting lo que necesitamos subir es todo menos test y los archivos de git como .gitignore y así. ñor de archivos para manu.cicloflorenciopintado.es

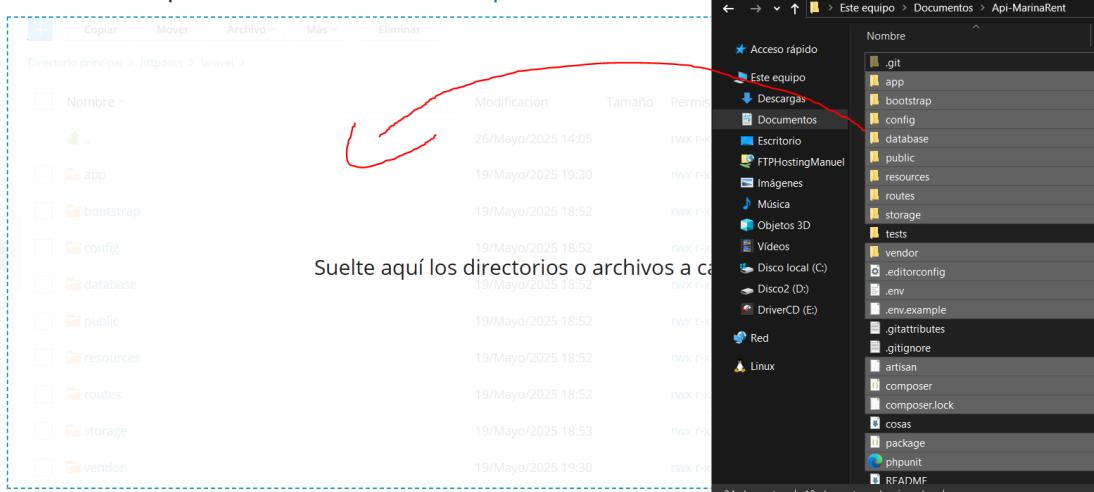


Ilustración 344 Despliegue XIII

Y nos tendrá que quedar tal que así

Nombre	Modificación	Tamaño	Permisos	Usuario	Grupo
..	26/Mayo/2025 14:05		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
app	19/Mayo/2025 19:30		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
bootstrap	19/Mayo/2025 18:52		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
config	19/Mayo/2025 18:52		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
database	19/Mayo/2025 18:52		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
public	19/Mayo/2025 18:52		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
resources	19/Mayo/2025 18:52		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
routes	19/Mayo/2025 18:52		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
storage	19/Mayo/2025 18:53		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln
vendor	19/Mayo/2025 19:30		rwx r-x r-x	manu.cicloflorenciopi_v56wtx3o7	psacln

Ilustración 345 Despliegue XIV

Una vez hecho esto iremos a la url de nuestro sitio y pondremos /laravel/public y nos tendrá que devolver la vista de laravel.

URL de prueba: <https://manu.cicloflorenciopintado.es/laravel/public>

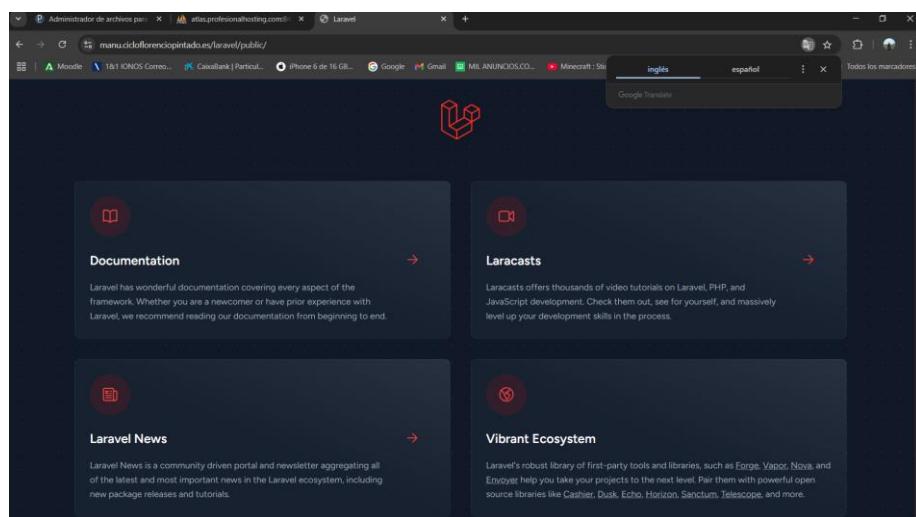


Ilustración 346 Despliegue XV

Cliente

Una vez tengamos nuestro hosting con el servidor en marcha podremos comenzar con la implementación de nuestro cliente

He de decir que todas ventanas y subpáginas de mi web son 100% responsive

Para desplegar el cliente lo que tendremos que hacer será una vez descargado el cliente haremos un “npm run build” para compilarlo.

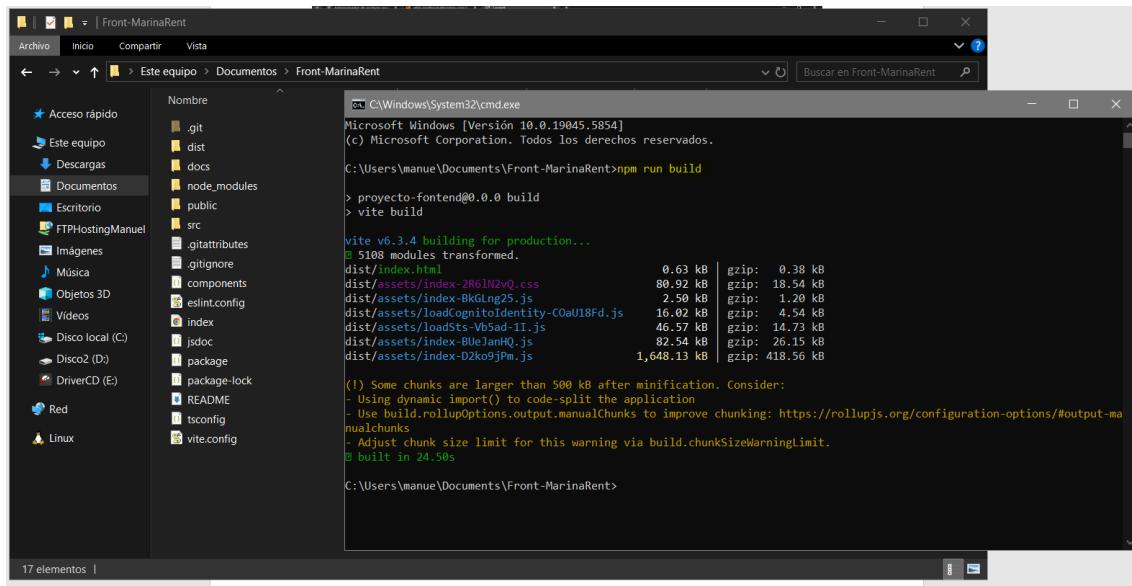


Ilustración 347 Despliegue XVI

Este comando nos creará una carpeta llamada dist la cual contiene los archivos necesarios para desplegarlo en el server.

En httpdocs soltaremos todos los archivos que contiene dist de la manera indicada previamente en el servidor.

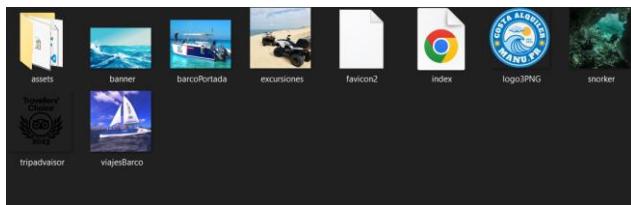


Ilustración 348 Despliegue XVII

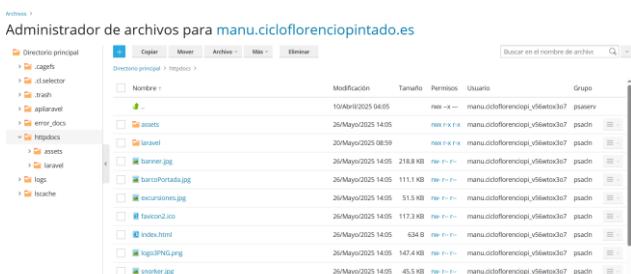


Ilustración 349 Despliegue XVIII

8. Seguridad

La seguridad en la app es uno de los puntos más importantes a la hora de crear una página web.

Por ello mi proyecto dispone de:

- Tokens de inicio de sesión.
- Tokens de recuperación de contraseña.
- EndPoints protegidos por sanctum.
- Contraseñas de la base de datos cifradas.
- Registro con contraseña de 6 caracteres, una mayúscula, una minúscula, al menos un número.
- Autenticación por email y contraseña.

También, utilizaremos control de versiones (Git) .Git es un sistema de control de versiones para nuestros proyectos de software de cualquier tipo, ya sean aplicaciones de escritorio, aplicaciones móviles o aplicaciones web.

Por tanto, teniendo nuestro proyecto subido al repositorio tendremos copia de seguridad de cada uno de los cambios realizados en nuestra aplicación.

<https://github.com/ManuelGonzalez709>

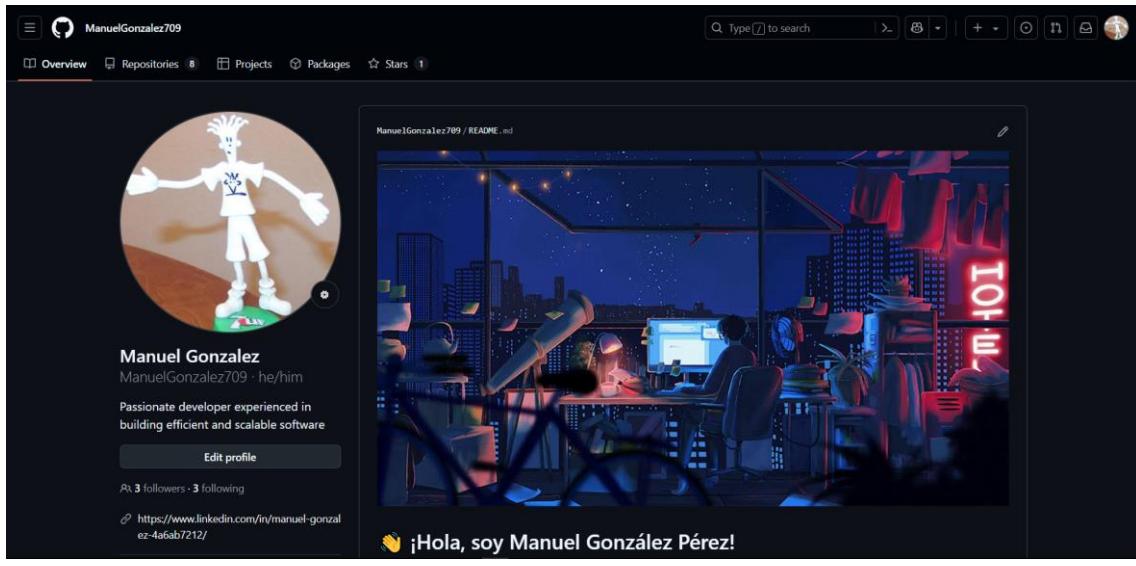


Ilustración 350 Perfil de Github

9. Pruebas

En el apartado de inicio de sesión podremos comprobar que si intentamos iniciar sesión con credenciales incorrectas no nos deja hacerlo

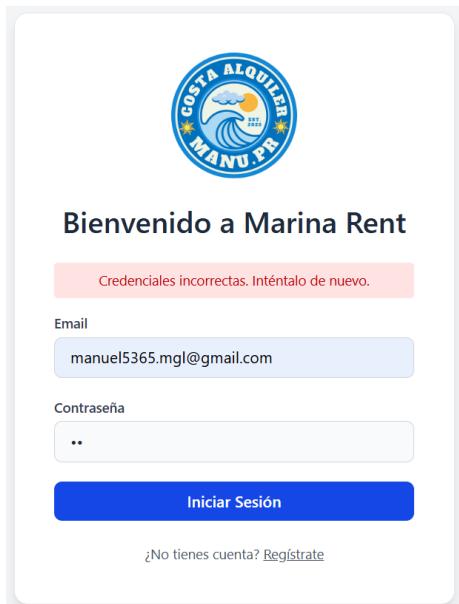


Ilustración 351 Pruebas I

Si intentamos enviar un restablecimiento de correo a un email que no existe no nos dejara hacerlo

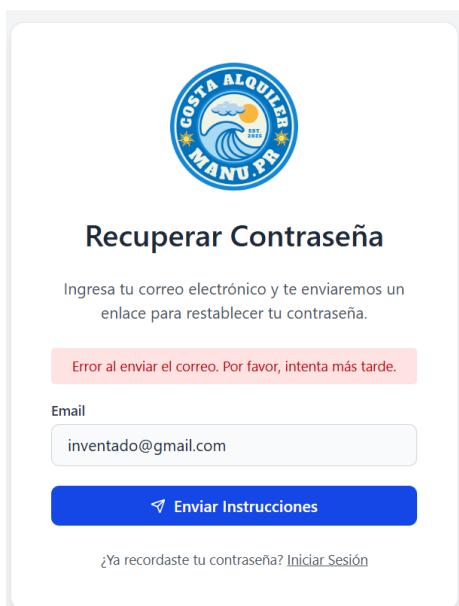


Ilustración 352 Pruebas II

En el register nos avisará de todos los campos que faltan:

Regístrate en Marina Rent

Nombre	<input type="text" value="tu nombre"/>
Apellidos	<input type="text" value="Tus apellidos"/> ! Completa este campo
Email	<input type="text" value="ejemplo@correo.com"/>
Fecha de Nacimiento	<input type="text" value="dd/mm/aaaa"/>
Contraseña	<input type="password" value="*****"/>
Confirmar Contraseña	<input type="password" value="*****"/>
Regístrate	

¿Ya tienes cuenta? [Iniciar Sesión](#)

Regístrate en Marina Rent

Nombre	<input type="text" value="hola"/>
Apellidos	<input type="text" value="Tus apellidos"/>
Email	<input type="text" value="ejemplo@correo.com"/> ! Completa este campo
Fecha de Nacimiento	<input type="text" value="dd/mm/aaaa"/>
Contraseña	<input type="password" value="*****"/>
Confirmar Contraseña	<input type="password" value="*****"/>
Regístrate	

¿Ya tienes cuenta? [Iniciar Sesión](#)

Regístrate en Marina Rent

Nombre	<input type="text" value="hola"/>
Apellidos	<input type="text" value="12"/>
Email	<input type="text" value="ejemplo@correo.com"/>
Fecha de Nacimiento	<input type="text" value="dd/mm/aaaa"/> ! Completa este campo
Contraseña	<input type="password" value="*****"/>
Confirmar Contraseña	<input type="password" value="*****"/>
Regístrate	

¿Ya tienes cuenta? [Iniciar Sesión](#)

Ilustración 353 Pruebas III

Ilustración 354 Pruebas IV

Ilustración 355 Pruebas V

Regístrate en Marina Rent

Nombre	<input type="text" value="hola"/>
Apellidos	<input type="text" value="12"/>
Email	<input type="text" value="12@gmail.com"/>
Fecha de Nacimiento	<input type="text" value="dd/mm/aaaa"/> ! Completa este campo
Contraseña	<input type="password" value="*****"/>
Confirmar Contraseña	<input type="password" value="*****"/>
Regístrate	

¿Ya tienes cuenta? [Iniciar Sesión](#)

Regístrate en Marina Rent

Nombre	<input type="text" value="hola"/>
Apellidos	<input type="text" value="12"/>
Email	<input type="text" value="12@gmail.com"/>
Fecha de Nacimiento	<input type="text" value="07/06/2025"/>
Contraseña	<input type="password" value="*****"/>
Confirmar Contraseña	<input type="password" value="*****"/> ! Completa este campo
Regístrate	

¿Ya tienes cuenta? [Iniciar Sesión](#)

Contraseña

Seguridad: Media
✓ Mínimo 6 caracteres
✓ Al menos una mayúscula
✓ Al menos una minúscula
✓ Al menos un número
✓ Al menos un carácter especial

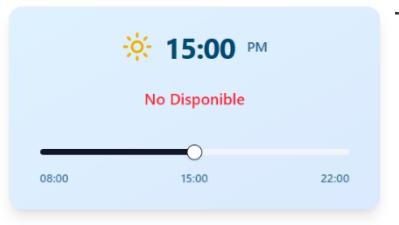
Confirmar Contraseña

Regístrate

¿Ya tienes cuenta? [Iniciar Sesión](#)

Ilustración 356 Pruebas VI Ilustración 357 Pruebas VII Ilustración 358 Pruebas VIII

No se podrá realizar ninguna reserva en caso de que la hora ya esté cogida, sea tardía o no haya posibilidad de coger dicha reserva para tal número de personas.



Personas



Añadir al Carrito

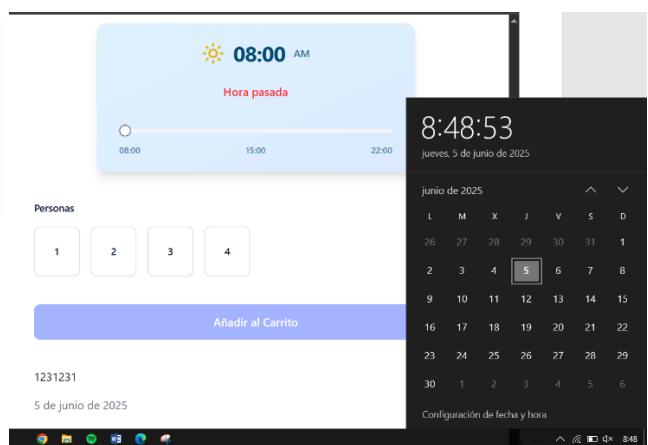
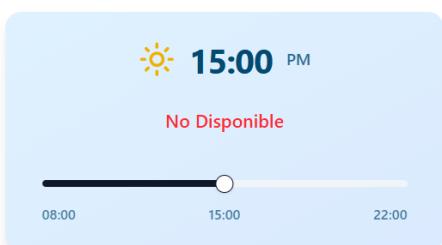


Ilustración 360 Pruebas X

\$300

Momento del Día



Personas



Añadir al Carrito

Ilustración 361 Pruebas XI

Al realizar una cancelación pide confirmación

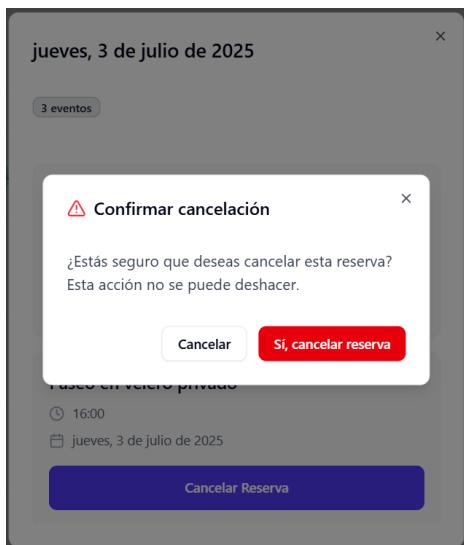


Ilustración 362 Pruebas XII

Al editar una publicación se obliga a tener 4 imágenes sí o sí para poder guardar los cambios de lo contrario no se aplicarán los cambios

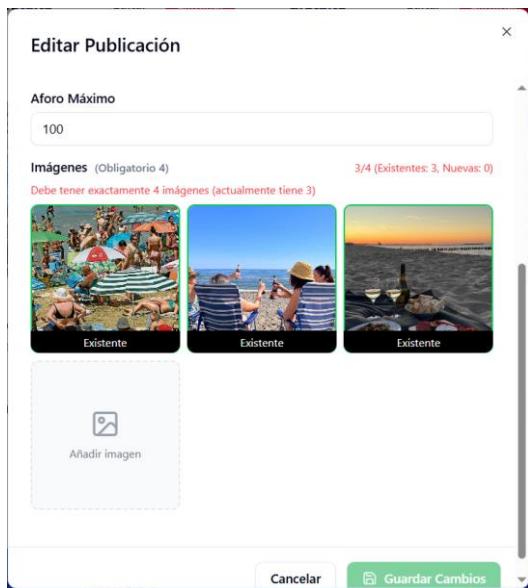


Ilustración 363 Pruebas XIII

Al cambiar la hora de la reserva, en caso de que alguien ya tenga reservado a esa hora lo que pasará es que mostrará que no está disponible y un mensaje de atención esta hora será intercambiada.

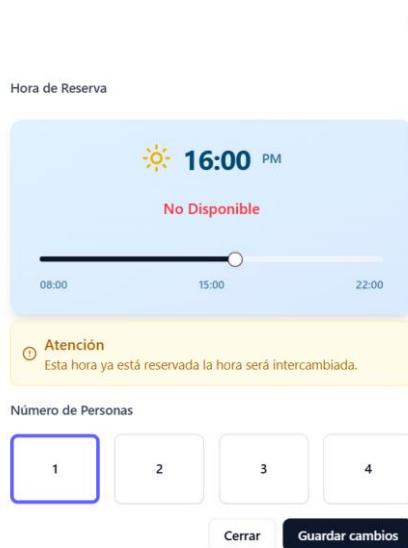


Ilustración 364 Pruebas XIX

10. Recursos

En este apartado vamos a ver los recursos que hemos necesitado para la implementación de nuestro proyecto.

Recursos Hardware

- Navegador como Chrome, Firefox, Edge, Safari o similar que soporte JavaScript moderno (ES6+).

- 8 GB de ram mínimos.

- Procesador de 16 núcleos.

Recursos Software

- Sistema Operativo Windows 10.

- Visual Studio Code (Gratis).

- GitHub.

- Hosting proporcionado por el centro.

Recursos Personales

- Responsable del proyecto (Manuel González Pérez).

Presupuesto

Recursos	Tipo	Horas	€/Horas	Total
Desarrollador	Personal	180€	10€	1800€
Hosting	Software	Gratis	0€	0€
			TOTAL	1800€

Ilustración 365 Presupuesto

11. Conclusiones

Consecución de los Objetivos.

El objetivo principal de nuestro proyecto, se ha llevado a cabo correctamente ya que era poder llevar el control total de los entrenamientos.

En la siguiente tabla, muestro que se han cumplido todos los objetivos, y como se llevan a cabo.

Objetivos	
Control de Horas	Se ha podido controlar las horas de reserva y la horas anticuadas
Control de Personas	Se ha podido controlar el aforo máximo y mínimo en cada momento.
Mapa de Marbella dinámico	Se ha podido implementar un mapa dinámico
Diseño minimalista	Se ha podido rediseñar la página para obtener una apariencia minimalista
Seguridad de la APP	Se ha podido implementar muchísima seguridad en la APP

Ilustración 366 Tabla de Objetivos

Problemas encontrados

Problemas con el email para el envío de correos, para ello cambié esta línea.

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME="vdevwul@gmail.com"
MAIL_PASSWORD="tofm oqrw kqku nzse"
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="marinaRent@gmail.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Ilustración 367 Problema encontrado

Mejoras

Por ultimo en mejoras incluiría que la IA pueda interactuar con la página, pero para ello hay que pagar una mensualidad de la API de 20€.

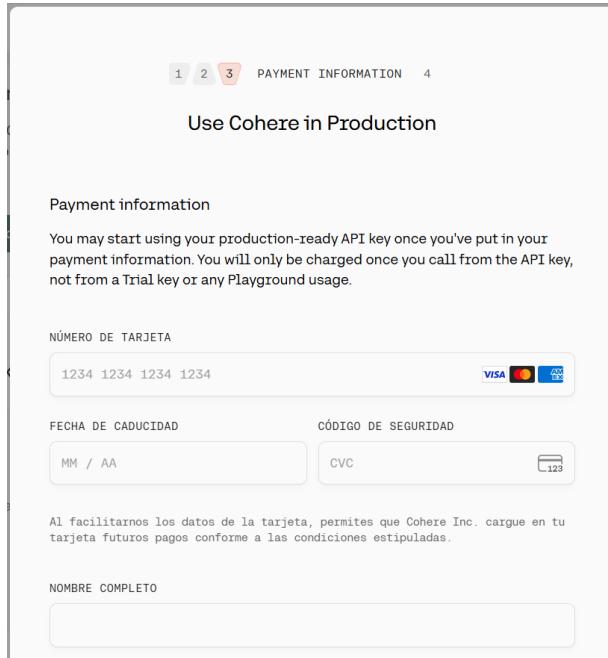


Ilustración 368 Pago mensual de cohore

12. Bibliografía

<https://www.udemy.com/course/react-de-principiante-a-experto-creando-mas-de-10-aplicaciones/learn/lecture/42828920#overview> (Curso de Udemy)

<https://www.youtube.com/watch?v=wGxDfSWC4Ww> (Tutorial 1 de react)

https://www.youtube.com/watch?v=yIrl_1CasXkM&t=1413s (Tutorial 2 de react)

<https://www.youtube.com/watch?v=bvxm389cYVI> (Tutorial 3 de react)

<https://www.youtube.com/watch?v=rn2LCOeNPds> (Curso de Laravel)

<https://www.youtube.com/watch?v=pvj7UyDZQ0g&t=35s> (Curso de Laravel 2)

<https://tailwindcss.com/plus/ui-blocks/marketing/feedback/404-pages-templates> (Tailwindcss)

Anexos

Manual de Usuario

Iniciar Sesión y Registrarte

Para acceder a la página tendríamos que acceder a la [url](#) y en caso de que no tengamos cuenta daríamos en registrarse

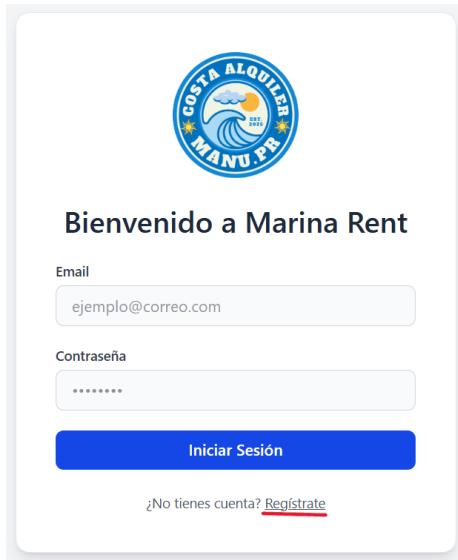


Ilustración 369 Manual de Usuario I

Luego tendríamos que llenar los campos correctamente, al hacer esto daremos en registrarse.

Nombre	<input type="text" value="Manuel"/>
Apellidos	<input type="text" value="Gonzalez Perez"/>
Email	<input type="text" value="manuel5365.mgl@gmail.com"/>
Fecha de Nacimiento	<input type="text" value="04/06/2025"/>
Contraseña	<input type="password" value="*****"/>
Seguridad:	<div style="width: 100px; background-color: green; height: 10px;"></div> Fuerte
<ul style="list-style-type: none"> ✓ Mínimo 6 caracteres ✓ Al menos una mayúscula ✓ Al menos una minúscula ✓ Al menos un número ✓ Al menos un carácter especial 	
Confirmar Contraseña	<input type="password" value="*****"/>
Regístrate	

Ilustración 370 Manual de Usuario II

Al dar en registrate ya tendríamos iniciada la sesión



Ilustración 371 Manual de Usuario III

En caso de que ya tengamos cuenta lo que haremos será rellenar los datos correctamente con su email y contraseña y daremos en iniciar sesión.

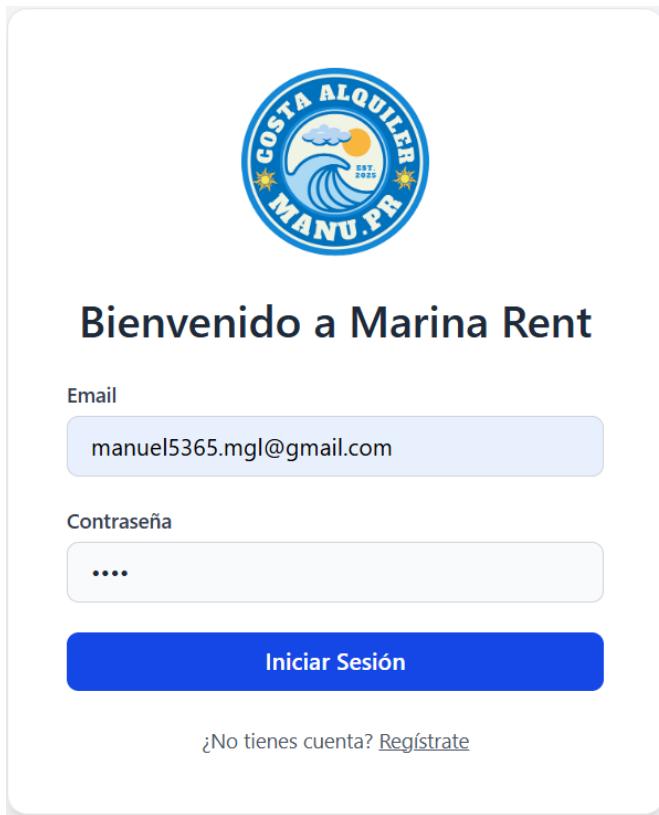


Ilustración 372 Manual de Usuario IV

Consultar publicaciones y añadir al carrito

Para consultar las publicaciones, en primer lugar tenemos un nav menú el cual nos dejara movernos entre los siguientes apartados que son los necesarios para ver las publicaciones.



Ilustración 373 Consultar Publicaciones I

Nos moveremos en 3 apartados para consultar las publicaciones:

-Home

En el home al hacer scroll para abajo tendremos un apartado que es de publicaciones destacadas, el cual al dar click en una de las publicaciones de las cuales son aleatorias por lo que puede ser alquilable o informativo , podremos ver sus datos.



Ilustración 374 Consultar Publicaciones Home II

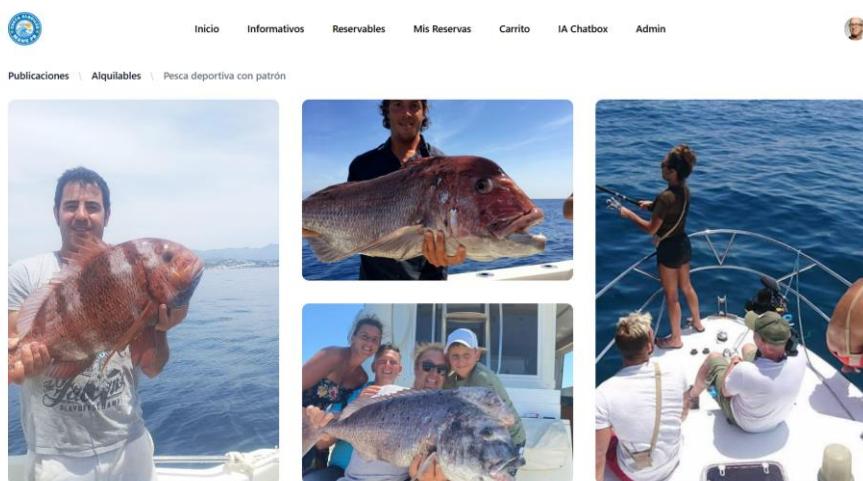


Ilustración 375 Consultar Publicaciones Home III

Pesca deportiva con patrón

Excursión de pesca para grupos

4 de julio de 2025

Highlights

- 💡 Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🌐 Portal de compra 100% seguro
- 🔴 Cancela cuando Quieras

Detalles

Disfruta de tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

\$90
Momento del Día

 15:00 PM

08:00 15:00 22:00

Personas

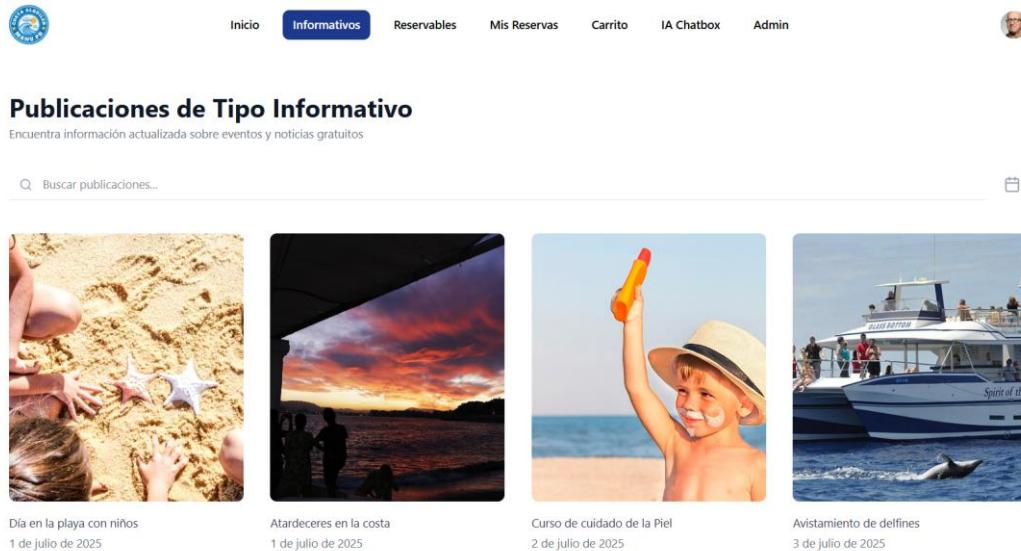
1 2 3 

Añadir al Carrito

Ilustración 376 Consultar Publicaciones Home IV

-Informativos

En el apartado de informativos podremos consultar las publicaciones de tipo informativo, para poder hacer eso tendremos que hacer click en el nav menú de arriba en el apartado de informativos.



The screenshot shows a navigation bar with links: Inicio, Informativos (highlighted), Reservables, Mis Reservas, Carrito, IA Chatbox, Admin, and a user profile icon. Below the navigation is a search bar with placeholder text 'Buscar publicaciones...'. Four news items are displayed in a grid:

- Día en la playa con niños** (1 de julio de 2025) - An image of children playing in sand with starfish.
- Atardeceres en la costa** (1 de julio de 2025) - An image of a sunset over the ocean.
- Curso de cuidado de la Piel** (2 de julio de 2025) - An image of a child holding a sun cream bottle.
- Avistamiento de delfines** (3 de julio de 2025) - An image of a boat on the water with a dolphin.

Ilustración 377 Consultar Publicaciones Informativos I

Debajo del todo tendremos un botón para cambiar de página para poder ver los productos de la siguiente página y al dar click cambiaríamos de página para ver mas productos



< Página 1 de 2 >

Ilustración 378 Consultar Publicaciones Informativos II

Para ver los datos de la publicación lo que haremos será dar click en una publicación de tipo informativo para leer su información

Publicaciones \ Informativos \ Día en la playa con niños

-
-
-

Ilustración 379 Consultar Publicaciones Informativos III

Las publicaciones de tipo informativo no se pueden reservar ya que solo son informativas.

Día en la playa con niños

Consejos para disfrutar un día en familia junto al mar

1 de julio de 2025

Este producto no está disponible para alquiler

Este artículo es de tipo informativo

Highlights

- ☀ Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🌐 Portal de compra 100% seguro
- 🚫 Cancela cuando Quieras

Detalles

Disfruta de tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

Ilustración 380 Consultar Publicaciones Informativos IV

-Reservables

En el apartado de reservables tendremos la posibilidad de consultar las publicaciones de tipo reservable.

Imagen	Título	Fecha	Precio
	Tour en barco por la bahía	1 de julio de 2025	60 €
	Moto de agua 30 min	2 de julio de 2025	45 €
	Paseo en velero privado	3 de julio de 2025	150 €
	Pesca deportiva con patrón	4 de julio de 2025	90 €

Ilustración 381 Consultar Publicaciones Reservables I

Debajo del todo tendremos un botón para cambiar de página para poder ver los productos de la siguiente página y al dar click cambiaríamos de página para ver mas productos



Clase de paddle surf
4 de julio de 2025

25 €

Kayak por acantilados
5 de julio de 2025

30 €

Snorkel guiado
5 de julio de 2025

35 €

Bautismo de buceo
6 de julio de 2025

70 €

< Página 1 de 2 >

Ilustración 382 Consultar Publicaciones Reservables II

Estas publicaciones si se pueden reservar, para ello lo que haremos será hacer click en una publicación.

Ilustración 383 Consultar Publicaciones Reservables III

Lo que haremos para realizar una reserva será seleccionar una franja horaria y un número de personas y daremos en añadir al carrito.

Con esto ya habríamos realizado la reserva

Pesca deportiva con patrón

Excursión de pesca para grupos

4 de julio de 2025

Highlights

- ☀️ Te lo Pasarás de Lujo
- 💵 100% Reembolsable
- 🌐 Portal de compra 100% seguro
- 🔴 Cancela cuando Quieras

Detalles

Disfruta tu día en la playa con nuestros artículos de alquiler, pensados para ofrecerte la máxima comodidad y funcionalidad. Todos nuestros productos están cuidadosamente seleccionados para adaptarse a tus necesidades, ya sea que busques relajarte bajo el sol, practicar actividades acuáticas o disfrutar de un paseo junto al mar. Reserva el tuyo con antelación y asegúrate de tener lo mejor para tu experiencia costera.

\$180

Momento del Día

 11:00 AM

08:00 15:00 22:00

Personas

1 2 3 4

Añadir al Carrito

Ilustración 384 Consultar Publicaciones Reservables IV

Realizar reservas y consultar las reservas

Para realizar las reservas tendremos que dar click en los apartados Carrito y Mis reservas para consultarlas.



Ilustración 385 Nav Menu Realizar Reservas y Consultar Reservas

Realizar reservas

Para realizar una reserva lo que haremos será ir al apartado de carrito y daremos en realizar pedido en caso de que se pueda. Y con esto ya estará realizada la reserva ☺.

Order summary	
Seguro de Riesgos	\$99.00
Taxas estimadas	\$8.32
Precio Reserva	\$112.32
Realizar Pedido	

Ilustración 386 Realizar reserva

Consultar Reservas

Para consultar reservas en el nav menú daremos click en el apartado de Mis Reservas, una vez aquí podremos ver las reservas que tenemos realizadas.

En esta página podremos cambiar de día para ver reservas de diferentes meses.

Ilustración 387 Consultar Reservas I

Si hacemos click en un día podremos cancelar la reserva y entonces esta reserva quedará libre para otra persona.

Ilustración 388 Consultar Reservas II / Ilustración 389 Consultar Reservas III

Restaurar Contraseña

Para restaurar la contraseña en caso de que se nos olvide, lo que tendremos que hacer será fallar la contraseña 3 veces.

Tras esto nos aparecerá un mensaje que dice Recuperar Contraseña

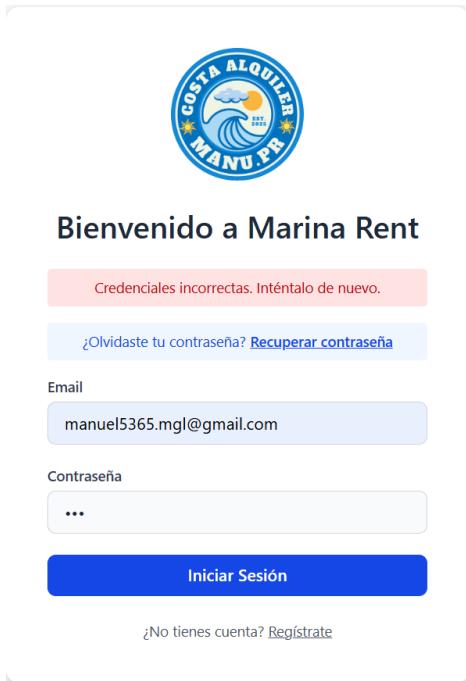


Ilustración 390 Restaurar Contraseña I

Después de esto nos llevará a una página donde tendremos que introducir el correo que queremos recuperar, lo que haremos será poner nuestro correo y dar en el botón de enviar restablecimiento

Ilustración 391 Restaurar Contraseña II

Ilustración 392 Restaurar Contraseña III

Y nos habrá llegado el correo



Ilustración 393 Restaurar Contraseña IV email I

Al abrirlo nos mostrará el siguiente enlace donde haremos click.

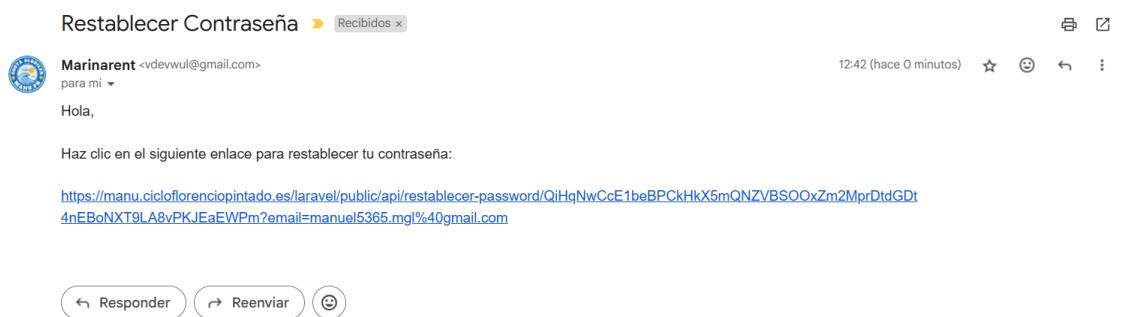


Ilustración 394 Restaurar Contraseña V email II

Y nos llevará a la página de restaurar password donde pondremos los datos de la nueva contraseña



Ilustración 395 Restaurar Contraseña VI

Y nos mostrará el siguiente mensaje dejándonos claro que podemos cerrar la ventana



Ilustración 396 Restaurar Contraseña VII

*Charlar con la IA *NEW**

Para charlar con la IA lo que tendremos que hacer es pinchar en el menú de navegación en el apartado de IA y mediante el input de abajo le podremos realizar preguntas y este nos responderá



Ilustración 397 IA Chatbox