

Informe de Laboratorio 11

Tema: Proyecto Final

Nota

Estudiante	Escuela	Asignatura
- Turpo Huanca, Wilson Josue - Hanco Mullisaca, Sergio - Sarayasi Huanaco, Jeferson	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702124

Laboratorio	Tema	Duración
11	Proyecto Final	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 27 de junio 2024	Al 6 de julio 2024

Docente
CARLO JOSE LUIS CORRALES DELGADO

1. URL de Repositorio Github final

- URL del proyecto Final para clonar el Repositorio GitHub.
https://github.com/shanccom/PROYECTO_FINAL_PWEB.git
- URL del Repositorio GitHub del primer prototipo para clonar: Registro trabajo en equipo.
<https://github.com/wilsonjosue/AppGestionHotelera.git>

2. Urls De Archivos Pedidos

URL de la explicación del Proyecto Final - Sergio Hanco Mullisaca:
<https://youtu.be/gkB4M900UP4>

URL de la explicación del Proyecto Final - Wilson Turpo Huanca:
https://www.youtube.com/watch?v=k1dAAYtu_hA&t=1s

URL de Diapositivas:
https://www.canva.com/design/DAGLvGbigBk/2a_JU4IXinCInakGXKRGmg/edit

3. Objetivos del proyecto

- Integrar los conocimientos adquiridos en el curso.
- Trabajar en equipo.
- Desarrollar una aplicación Web para una empresa real, que incluya:
Backend (Django) y FrontEnd (Ajax / Framework JavaScript: Angular)

4. Propuesta de trabajo

- Título: Buscando agencia para el desarrollo de sitios web y aplicaciones móviles de operaciones para huéspedes del hotel.
- URL del pedido del cliente: <https://www.upwork.com/freelance-jobs/apply/Mobile-Apps-iOS-and-Android-a~01499deb6184b90334/>

Sobre el proyecto

 166 propuestas

 Proyecto remoto

Haga su oferta

Monto de la oferta

\$ 50

Dólar esta...

Dirección de correo electrónico

jane@freelancer.com

Oferta por el proyecto

5. Actividades

- Requerimientos Funcionales
1. Gestión de Reservas:
 - a) Crear, actualizar, listar, y eliminar reservas de habitaciones.

- b) Ver detalles de cada reserva.
 - c) Consultas que devuelvan datos en formato JSON.
- 2. Gestión de Clientes:
 - a) Registrar y administrar la información de los huéspedes.
 - b) Consultar detalles de cada huésped.
- 3. Gestión de Habitaciones:
 - a) Administrar la disponibilidad de las habitaciones.
 - b) Crear, actualizar, listar, y eliminar habitaciones.
 - c) Asignar habitaciones a las reservas.
- 4. Notificaciones y Comunicaciones:
 - a) Enviar correos de confirmación de reserva.
 - b) Notificaciones a los huéspedes sobre su estadía.
- 5. Reportes:
 - a) Generar y descargar informes en formato PDF sobre las reservas y ocupación del hotel.

6. Primer entorno de trabajo de trabajo

- Configuración del Backend (Django): creación e instalación previa del entorno de trabajo python y Django.

```
C:\Users\danie>cd C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHoteleria
C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHoteleria>python -m venv sistema
C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHoteleria>.\sistema\Scripts\activate
(sistema) C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHoteleria>pip install django
Collecting django
  Using cached Django-5.0.6-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.7.0 (from django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.5.0-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
  Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached Django-5.0.6-py3-none-any.whl (8.2 MB)
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.0-py3-none-any.whl (43 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.0.6 sqlparse-0.5.0 tzdata-2024.1

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

1. Crear el Proyecto Django:
django-admin startproject backend
cd backend

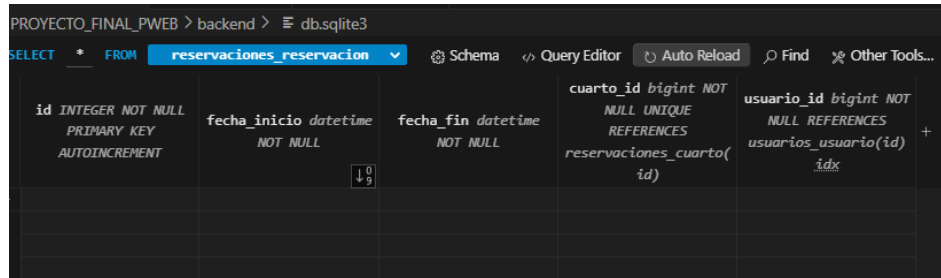
```
C:\Users\danie>cd C:\Users\danie\OneDrive\Documentos\AngularProjects\GestionHoteleria\PROYECTO_FINAL_PWEB
C:\Users\danie\OneDrive\Documentos\AngularProjects\GestionHoteleria\PROYECTO_FINAL_PWEB>django-admin startproject backend
```

2. Crear la Aplicación Django:

python manage.py startapp reservaciones

```
C:\Users\danie\OneDrive\Documentos\AngularProjects\GestionHotelera\PROYECTO_FINAL_PWEB\backend>python manage.py startapp reservaciones
```

3. Creamos la base de datos en sqlparse==



id	fecha_inicio	fecha_fin	cuarto_id	usuario_id
INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	datetime NOT NULL	datetime NOT NULL	bigint NOT NULL UNIQUE REFERENCES reservaciones_cuarto(id)	bigint NOT NULL REFERENCES usuarios_usuario(id)

4. Configuración previa de la Base de Datos en settings.py:

```
74 DATABASES = {  
75     'default': {  
76         'ENGINE': 'django.db.backends.sqlite3',  
77         'NAME': BASE_DIR / 'db.sqlite3',  
78     }  
79 }  
80
```

5. Instalar y Configurar Django Rest Framework:

pip install djangorestframework

```
(sistema) C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHotelera\hotel_management>pip install djangorestframework  
Collecting djangorestframework  
  Downloading djangorestframework-3.15.2-py3-none-any.whl.metadata (10 kB)  
Requirement already satisfied: django>4.2 in c:\users\danie\onedrive\documentos\angularprojects\lab_pw2_appgestionhotelera\sistema\lib\site-packages (from djangorestframework) (5.0.6)  
Requirement already satisfied: asgiref<4, >=3.7.0 in c:\users\danie\onedrive\documentos\angularprojects\lab_pw2_appgestionhotelera\sistema\lib\site-packages (from django>4.2->djangorestframework) (3.8.1)  
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\danie\onedrive\documentos\angularprojects\lab_pw2_appgestionhotelera\sistema\lib\site-packages (from django>4.2->djangorestframework) (0.5.0)  
Requirement already satisfied: tzdata in c:\users\danie\onedrive\documentos\angularprojects\lab_pw2_appgestionhotelera\sistema\lib\site-packages (from django>4.2->djangorestframework) (2024.1)  
Downloading djangorestframework-3.15.2-py3-none-any.whl (1.1 MB)  
----- 1.1/1.1 MB 646.9 kB/s eta 0:00:00  
Installing collected packages: djangorestframework  
Successfully installed djangorestframework-3.15.2  
  
[notice] A new release of pip is available: 24.0 -> 24.1.1  
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Agregar 'rest_framework', y 'reservaciones', a INSTALLED_APPS en settings.py:

```
22
23  INSTALLED_APPS = [
24      'django.contrib.admin',
25      'django.contrib.auth',
26      'django.contrib.contenttypes',
27      'django.contrib.sessions',
28      'django.contrib.messages',
29      'django.contrib.staticfiles',
30
31      ## APPS PROPIAS
32      'usuarios',
33      'reservaciones',
34      'rest_framework',
35      'rest_framework.authtoken',
36      'corsheaders',
37  ]
38
```

7. Código para el backend(Django)

1. Definir los Modelos en reservaciones/models.py:

TipoCuarto(models.Model):

Cuarto(models.Model):

Reservacion(models.Model):

```
PROYECTO_FINAL_PWEB > backend > reservaciones > models.py > Reservacion > __str__
1  from django.db import models
2  from django.conf import settings
3  from django.utils import timezone
4
5  class TipoCuarto(models.Model):
6      nombre = models.CharField(max_length=100)
7      descripcion = models.TextField()
8      precio_por_noche = models.DecimalField(max_digits=8, decimal_places=2)
9
10     def __str__(self):
11         return self.nombre
12
13     class Cuarto(models.Model):
14         tipo = models.ForeignKey(TipoCuarto, on_delete=models.CASCADE)
15         numero = models.CharField(max_length=10, unique=True)
16
17         def __str__(self):
18             return f"{self.tipo.nombre} - {self.numero}"
19
```

```
19
20 class Reservacion(models.Model):
21     usuario = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
22     cuarto = models.OneToOneField(Cuarto, on_delete=models.CASCADE)
23     fecha_inicio = models.DateTimeField()
24     fecha_fin = models.DateTimeField()
25
26     def __str__(self):
27         return f"Reservación de {self.usuario} en {self.cuarto} desde {self.fecha_inicio} hasta {self.fecha_fin}"
28
29     @property
30     def duracion(self):
31         return (self.fecha_fin - self.fecha_inicio).days
32
33     @property
34     def costo_total(self):
35         precio_por_noche = self.cuarto.tipo.precio_por_noche
36         return precio_por_noche * self.duracion
```

2. Instala pycpg2:

pip install pycpg2-binary

```
(sistema) C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHotelera\hotel_management>pip install psycpg2-binary
Collecting psycpg2-binary
  Downloading psycpg2_binary-2.9.9-cp312-cp312-win_amd64.whl.metadata (4.6 kB)
  Downloading psycpg2_binary-2.9.9-cp312-cp312-win_amd64.whl (1.2 MB)
-----
1.2/1.2 MB 2.3 MB/s eta 0:00:00
Installing collected packages: psycpg2-binary
Successfully installed psycpg2-binary-2.9.9

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(sistema) C:\Users\danie\OneDrive\Documentos\AngularProjects\Lab_PW2_AppGestionHotelera\hotel_management>pip list
Package            Version
-----
asgiref            3.8.1
Django             5.0.6
django-rest-framework 3.15.2
pip                24.0
psycpg2-binary     2.9.9
sqlparse           0.5.0
tzdata             2024.1

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

3. Crear y Aplicar Migraciones:

```
python manage.py makemigrations
python manage.py migrate
```

```
C:\Users\danie\OneDrive\Documentos\AngularProjects\GestionHotelera\PROYECTO_FINAL_PWEB\backend>python manage.py makemigrations
```

4. Definir Serializadores en reservaciones/serializers.py

```
PROYECTO_FINAL_PWEB > backend > reservaciones > serializers.py > ...
1  from rest_framework import serializers
2  from .models import Cuarto, TipoCuarto, Reservacion
3
4  class CuartoSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Cuarto
7          fields = '__all__'
8
9  class TipoCuartoSerializer(serializers.ModelSerializer):
10     class Meta:
11         model = TipoCuarto
12         fields = '__all__'
13
14     class ReservacionSerializer(serializers.ModelSerializer):
15         class Meta:
16             model = Reservacion
17             fields = ['id', 'usuario', 'cuarto', 'fecha_inicio', 'fecha_fin']
```

5. Crear Vistas para la API en reservaciones/views.py

```
PROYECTO_FINAL_PWEB > backend > reservaciones > serializers.py > ...
1  from rest_framework import serializers
2  from .models import Cuarto, TipoCuarto, Reservacion
3
4  class CuartoSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Cuarto
7          fields = '__all__'
8
9  class TipoCuartoSerializer(serializers.ModelSerializer):
10     class Meta:
11         model = TipoCuarto
12         fields = '__all__'
13
14     class ReservacionSerializer(serializers.ModelSerializer):
15         class Meta:
16             model = Reservacion
17             fields = ['id', 'usuario', 'cuarto', 'fecha_inicio', 'fecha_fin']
```

6. Configurar URLs en reservaciones/urls.py

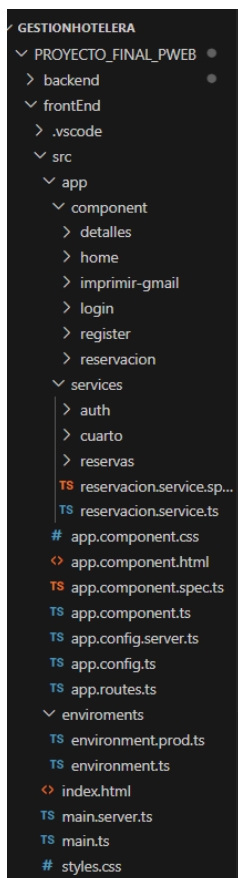
```
PROYECTO_FINAL_PWEB > backend > reservaciones > urls.py > ...
1  from django.urls import path, include
2  from rest_framework.routers import DefaultRouter
3  from .views import CuartoViewSet, TipoCuartoViewSet, ReservacionViewSet
4
5  router = DefaultRouter()
6  router.register(r'cuartos', CuartoViewSet)
7  router.register(r'tipos', TipoCuartoViewSet)
8  router.register(r'reservaciones', ReservacionViewSet)
9
10 urlpatterns = [
11     path('', include(router.urls)),
12     path('', include)
13 ]
```

7. Configurar URLs en backend/urls.py:

```
PROYECTO_FINAL_PWEB > backend > backend > urls.py > ...
1  from django.contrib import admin
2  from django.urls import path, include
3
4  urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('usuarios/', include('usuarios.urls')),
7     path('reservaciones/', include('reservaciones.urls'))
8 ]
9
```


8. Código para el frontend(Angular)

1. Frontend, sus componentes y sus servidores



2. Componente detalles donde se llama a la base de datos, tabla tipos cuartos, donde se muestra en específico la información de un cuarto.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > detalles > TS detalles.component.ts > DetallesComponent
1  import { Component } from '@angular/core';
2  import { AuthService } from '../services/auth/auth.service';
3  import { OnInit } from '@angular/core';
4  import { CommonModule } from '@angular/common';
5  import { ReservasService } from '../services/reservas/reservas.service';
6  import { CuartoService } from '../services/cuarto/cuarto.service';
7  import { RouterModule, RouterOutlet } from '@angular/router';
8  import { Router } from '@angular/router';
9  import { ActivatedRoute } from '@angular/router';
10 import { ReservacionComponent } from '../reservacion/reservacion.component';
11
12 @Component({
13   selector: 'app-detalles',
14   standalone: true,
15   imports: [CommonModule, RouterModule, RouterOutlet, ReservacionComponent],
16   templateUrl: './detalles.component.html',
17   styleUrls: ['./detalles.component.css']
18 })
19 export class DetallesComponent implements OnInit {
20   public tiposDeCuartos: any[] = [];
21   public cuartosDisponibles: any[] = [];
22   public elid: number = 0;
23   isAuthenticated = false;
24   reservas: any[] = [];
25
26   constructor(private authService: AuthService, private cuartoService: CuartoService, private router:
27
```

```

PROYECTO_FINAL_PWEB > frontend > src > app > component > detalles > TS detalles.component.ts > DetallesComponent > ngOnInit
19 export class DetallesComponent implements OnInit {
20   public tiposDeCuartos: any[] = [];
21   public cuartosDisponibles: any[] = [];
22   public elid: number = 0;
23   isAuthenticated = false;
24   reservas: any[] = [];
25
26   constructor(private authService: AuthService, private cuartoService: CuartoService, private router: Router, private route: ActivatedRoute) {}
27
28   ngOnInit(): void {
29     this.authService.isLoggedIn().subscribe(isLoggedIn => {
30       this.isAuthenticated = isLoggedIn;
31       if (this.isAuthenticated) {
32         this.loadTiposDeCuartos();
33       }
34     });
35     this.route.params.subscribe(params => {
36       this.elid = params['id'];
37       this.loadTiposDeCuartos = this.tiposDeCuartos.find(t => t.id === this.elid);
38     });
39   }
40   logout(): void {
41     this.authService.logout();
42   }
43
44   private loadTiposDeCuartos(): void {
45     this.cuartoService.getTiposDeCuartos().subscribe(
46       (data: any[]) => {
47         console.log('Datos obtenidos:', data);
48         this.tiposDeCuartos = data;
49       },
50       (error) => {
51         console.error('Error al obtener tipos de cuartos:', error);
52       }
53     );
54   }
55
56   obtenerID(): number {
57     return this.elid;
58   }
59 }

```

3. Componente home donde se podrá visualizar todos los cuartos de la base de datos/tabla tipode-cuartos.

```

PROYECTO_FINAL_PWEB > frontend > src > app > component > home > TS home.component.ts > ...
1 import { Component, Output, EventEmitter } from '@angular/core';
2 import { AuthService } from '../services/auth/auth.service';
3 import { OnInit } from '@angular/core';
4 import { CommonModule } from '@angular/common';
5 import { ReservasService } from '../services/reservas/reservas.service';
6 import { CuartoService } from '../services/cuarto/cuarto.service';
7 import { RouterModule, RouterOutlet } from '@angular/router';
8 import { Router } from '@angular/router';
9
10
11 @Component({
12   selector: 'app-home',
13   standalone: true,
14   imports: [CommonModule, RouterModule, RouterOutlet],
15   templateUrl: './home.component.html',
16   styleUrls: ['./home.component.css']
17 })

```

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > home > TS home.component.ts > ...
18 export class HomeComponent implements OnInit {
19   public tiposDeCuartos: any[] = [];
20   public cuartosDisponibles: any[] = [];
21   public selectedTipoId: number | null = null;
22   public cuartoSeleccionado: any | null = null;
23   dato: number = 0;
24
25   isAuthenticated = false;
26   reservas: any[] = [];
27
28   constructor(private authService: AuthService, private cuartoService: CuartoService, private router:
29
30   ngOnInit(): void {
31     this.authService.isLoggedIn().subscribe(isLoggedIn => {
32       this.isAuthenticated = isLoggedIn;
33       if (this.isAuthenticated) {
34         this.loadTiposDeCuartos();
35       }
36     });
37   }
38   logout(): void {
39     this.authService.logout();
40   }
41
42   private loadTiposDeCuartos(): void {
43     this.cuartoService.getTiposDeCuartos().subscribe(
44       (data: any[]) => {
45         console.log('Datos obtenidos:', data);
46         this.tiposDeCuartos = data;
47       },
48       (error) => {
49         console.error('Error al obtener tipos de cuartos:', error);
50       }
51     );
52   }
53
54   onTipoSeleccionado(id: number) {
55     this.router.navigate(['/detalles', id]);
56   }
57 }
58
59
```

4. Componente Imprimir-gmail se basa principalmente en imprimir un PDF donde se ve el usuario, el cuarto, fecha de inicio y fecha de fin.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > imprimir-gmail > TS imprimir-gmail.component.ts > ...
1 import { Component } from '@angular/core';
2 import { RouterModule } from '@angular/router';
3 import jsPDF from 'jspdf';
4 import { ReservacionService } from '../services/reservacion.service';
5 import { ActivatedRoute } from '@angular/router';
6 import { ReservacionComponent } from '../reservacion/reservacion.component';
7
8 @Component({
9   selector: 'app-imprimir',
10  standalone: true,
11  imports: [RouterModule, ReservacionComponent],
12  templateUrl: './imprimir-gmail.component.html',
13  styleUrls: ['./imprimir-gmail.component.css']
14 })
15 export class ImprimirComponent {
16
17   // Datos que se mostrarán en el PDF
18   data: any = {
19     titulo: 'Reserva de Cuarto',
20     usuario: '',
21     cuarto: '',
22     fecha_inicio: '',
23     fecha_fin: ''
24   };
25 }
```

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > imprimir-gmail > TS imprimir-gmail.component.ts > ...
15 export class ImprimirComponent {
26   constructor(private reservacionService: ReservacionService, private route: ActivatedRoute) {}
27
28   ngOnInit(): void {
29     this.route.paramMap.subscribe(params => {
30       const reservacionId = params.get('id');
31       console.log('ID de reservación:', reservacionId);
32
33       if (reservacionId && !isNaN(Number(reservacionId))) {
34         this.reservacionService.getReservacionById(+reservacionId).subscribe({
35           next: data => {
36             console.log('Datos de la reservación:', data);
37
38             this.data = {
39               titulo: 'Reserva de Cuarto',
40               fecha_inicio: data.fecha_inicio,
41               fecha_fin: data.fecha_fin
42             };
43
44             if (data.usuario) {
45               this.reservacionService.getUserById(data.usuario).subscribe({
46                 next: user => {
47                   console.log('Datos del usuario:', user);
48                   this.data.usuario = user.username;
49                 },
50                 error: error => {
51                   console.error('Error al obtener el usuario', error);
52                 }
53               });
54             } else {
55               console.error('ID de usuario no proporcionado');
56             }
57           }
58         });
59       }
60     });
61   }
62 }
```

```
58   if (data.cuarto) {
59     this.reservacionService.getCuartoById(data.cuarto).subscribe({
60       next: cuarto => {
61         console.log('Datos del cuarto:', cuarto);
62         this.data.cuarto = cuarto.numero; // Asegúrate de que `numero` sea el campo correcto
63       },
64       error: error => {
65         console.error('Error al obtener el cuarto', error);
66       }
67     });
68   } else {
69     console.error('ID de cuarto no proporcionado');
70   }
71 },
72 error: error => {
73   console.error('Error al obtener los datos de la reservación', error);
74 },
75 });
76 } else {
77   console.error('ID de reservación inválido');
78 }
79 });
80 }
81
82 generatePDF() {
83   const doc = new jsPDF();
84
85   doc.text(this.data.titulo, 10, 10);
86   doc.text('Usuario: ${this.data.usuario}', 10, 20);
87   doc.text('Cuarto: ${this.data.cuarto}', 10, 30);
88   doc.text('Fecha de Inicio: ${this.data.fecha_inicio}', 10, 40);
89   doc.text('Fecha de Fin: ${this.data.fecha_fin}', 10, 50);
90
91   doc.save('reserva.pdf');
92 }
93 }
```

```
93  
94 private getPDFBase64(): Promise<string> {  
95     return new Promise((resolve, reject) => {  
96         const doc = new jsPDF();  
97         doc.text(this.data.titulo, 10, 10);  
98         doc.text(`Usuario: ${this.data.usuario}`, 10, 20);  
99         doc.text(`Cuarto: ${this.data.cuarto}`, 10, 30);  
100        doc.text(`Fecha de Inicio: ${this.data.fecha_inicio}`, 10, 40);  
101        doc.text(`Fecha de Fin: ${this.data.fecha_fin}`, 10, 50);  
102  
103        const pdfOutput = doc.output('datauristring');  
104        const base64PDF = pdfOutput.split(',')[1]; // Extrae la parte base64 del data URI  
105        resolve(base64PDF);  
106    });  
107 }  
108  
109 gmail({})  
110  
111 }  
112  
113
```

5. Componente login se utiliza para iniciar sesión, es necesario tener un usuario para poder ingresar a la página.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > login > Ts login.component.ts > ...  
1  import { Component } from '@angular/core';  
2  import { FormBuilder, FormGroup, Validators } from '@angular/forms';  
3  import { AuthService } from '../../services/auth/auth.service';  
4  import { CommonModule } from '@angular/common';  
5  import { Router } from '@angular/router';  
6  import { ReactiveFormsModule } from '@angular/forms';  
7  
8  @Component({  
9      selector: 'app-login',  
10     standalone: true,  
11     templateUrl: './login.component.html',  
12     imports: [CommonModule, ReactiveFormsModule ],  
13     styleUrls: ['./login.component.css']  
14 })  
15
```

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > login > ts login.component.ts > ...
15
16 export class LoginComponent {
17   loginForm: FormGroup;
18   errorMessage: string | null = null;
19
20   constructor(
21     private fb: FormBuilder,
22     private authService: AuthService,
23     private router: Router
24   ) {
25     this.loginForm = this.fb.group({
26       username: ['', Validators.required],
27       password: ['', Validators.required]
28     });
29   }
30
31   // Método para manejar el envío del formulario
32   onSubmit(): void {
33     if (this.loginForm.valid) {
34       this.authService.login(this.loginForm.value).subscribe({
35         next: (response) => {
36           // Supongamos que el token está en response.token
37           this.authService.setSession(response.token);
38           this.router.navigate(['/home']);
39         },
40         error: (err) => {
41           // Manejar errores aquí
42           this.errorMessage = 'Inicio de sesión fallido. Verifique sus credenciales.';
43         }
44       });
45     }
46   }
47
48   goToRegister(): void {
49     this.router.navigate(['/register']);
50   }
51 }
```

6. Componente register es parte del inicio de sección ya que si no se tiene un usuario se puede crear una.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > register > TS register.component.ts > ...
1 import { Component } from '@angular/core';
2 import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';
3 import { AuthService } from '../../services/auth/auth.service';
4 import { Router } from '@angular/router';
5 import { CommonModule } from '@angular/common';
6
7 @Component({
8   selector: 'app-register',
9   standalone: true,
10  imports: [CommonModule, ReactiveFormsModule],
11  templateUrl: './register.component.html',
12  styleUrls: ['./register.component.css']
13 })
14
15 export class RegisterComponent {
16   registerForm: FormGroup;
17
18   constructor(private fb: FormBuilder, private authService: AuthService, private router: Router) {
19     this.registerForm = this.fb.group({
20       username: ['', Validators.required],
21       email: ['', [Validators.required, Validators.email]],
22       password: ['', Validators.required],
23       password2: ['', Validators.required]
24     });
25   }
26
27   onSubmit() {
28     if (this.registerForm.valid) {
29       this.authService.register(this.registerForm.value).subscribe(
30         response => {
31           console.log('Registration successful', response);
32           // Guardar el token y actualizar el estado de autenticación
33           this.authService.setSession(response.token);
34           this.router.navigate(['/home']);
35         },
36         error => {
37           console.error('Registration error', error);
38         }
39       );
40     }
41   }
42 }
```

7. Componente reservacion para obtener todos los datos necesarios para la reservacion o crear una reservacion

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > component > reservacion > TS reservacion.component.ts > ...
1 import { Component } from '@angular/core';
2 import { ReservacionService } from '../../services/reservacion.service';
3 import { FormsModule } from '@angular/forms';
4 import { RouterModule } from '@angular/router';
5 import { Router } from '@angular/router';
6 import { HttpClient } from '@angular/common/http';
7
8 @Component({
9   selector: 'app-reservacion',
10  standalone: true,
11  imports: [FormsModule, RouterModule],
12  templateUrl: './reservacion.component.html',
13  styleUrls: ['./reservacion.component.css']
14 })
15
16 export class ReservacionComponent {
17   reservacion = {
18     usuario: '',
19     cuarto: '',
20     fecha_inicio: '',
21     fecha_fin: ''
22   };
23
24   constructor(private reservacionService: ReservacionService, private router: Router, private http: HttpClient) {}
25
26   onSubmit() {
27     this.reservacionService.createReservacion(this.reservacion).subscribe(response => {
28       console.log('Reservación creada', response);
29       this.router.navigate(['/imprimir', response.id]);
30     }, error => {
31       console.error('Error al crear la reservación', error);
32     });
33   }
34 }
35
36 }
```

8. PARA LA CREACION DE LA SERVICIOS: Creacion de auth: se obtiene datos de los usuarios.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > services > auth > TS auth.service.ts > AuthService > register
1  import { Injectable, Inject, PLATFORM_ID } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Router } from '@angular/router';
4  import { BehaviorSubject, Observable } from 'rxjs';
5  import { isPlatformBrowser } from '@angular/common';
6  import { tap } from 'rxjs/operators';
7
8  @Injectable({
9    providedIn: 'root'
10 })
11 export class AuthService {
12   private apiUrl = 'http://localhost:8000/usuarios/';
13   private isAuthenticated = new BehaviorSubject<boolean>(false);
14   private isBrowser: boolean;
15
16   constructor(
17     private http: HttpClient,
18     private router: Router,
19     @Inject(PLATFORM_ID) private platformId: Object
20   ) {
21     this.isBrowser = isPlatformBrowser(this.platformId);
22     if (this.isBrowser) {
23       this.isAuthenticated.next(!localStorage.getItem('token'));
24     }
25   }
26
27   // Registro de usuario
28   register(userData: { username: string; email: string; password: string; password2: string }): Observable<any> {
29     return this.http.post<any>(this.apiUrl + 'register/', userData).pipe(
30       tap(response => {
31         if (response && response.token) {
32           this.setSession(response.token);
33         }
34       })
35     );
36   }
37 }
```

```
37
38   // Inicio de sesión
39   login(credentials: { username: string; password: string }): Observable<any> {
40     return this.http.post<any>(this.apiUrl + 'login/', credentials).pipe(
41       tap(response => {
42         if (response && response.token) {
43           this.setSession(response.token);
44         }
45       })
46     );
47   }
48
49   // Guardar el token en localStorage y actualizar el estado de autenticación
50   public setSession(token: string): void {
51     if (this.isBrowser) {
52       localStorage.setItem('token', token);
53       this.isAuthenticated.next(true);
54     }
55   }
56
57   // Cerrar sesión
58   logout(): void {
59     if (this.isBrowser) {
60       localStorage.removeItem('token');
61       this.isAuthenticated.next(false);
62       this.router.navigate(['/login']);
63     }
64   }
65
66   // Verificar si el usuario está autenticado
67   isLoggedIn(): Observable<boolean> {
68     return this.isAuthenticated.asObservable();
69   }
70 }
```


9. PARA LA CREACION DE LA SERVICIOS: Creacion de cuarto: Se obtiene datos de los cuartos.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > services > cuarto > TS cuarto.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class CuartoService {
9    private apiUrl = 'http://localhost:8000/reservaciones/';
10    private tiposApiUrl = `${this.apiUrl}tipos/`;
11    private cuartosApiUrl = `${this.apiUrl}cuartos/`;
12
13
14    constructor(private http: HttpClient) {}
15
16    // Obtener tipos de cuartos
17    getTiposDeCuartos(): Observable<any[]> {
18      return this.http.get<any[]>(this.tiposApiUrl);
19    }
20
21    // Obtener cuartos disponibles para reservar según el tipo
22    getCuartosDisponibles(tipoId: number): Observable<any[]> {
23      return this.http.get<any[]>(`${this.cuartosApiUrl}?tipo=${tipoId}`);
24    }
25  }
26
27
```

10. PARA LA CREACION DE LA SERVICIOS: Creacion de reservas: Se obtiene datos de las reservaciones.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > services > Ts reservacion.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class ReservacionService {
9
10     private apiUrl = 'http://127.0.0.1:8000/reservaciones/reservaciones/';
11
12     constructor(private http: HttpClient) { }
13
14     // Método para obtener todas las reservaciones
15     getReservaciones(): Observable<any[]> {
16       return this.http.get<any[]>(this.apiUrl);
17     }
18
19     // Método para crear una nueva reservación
20     createReservacion(reservacion: any): Observable<any> {
21       return this.http.post<any>(this.apiUrl, reservacion);
22     }
23
24     getReservacionById(id: number): Observable<any> {
25       const url = `${this.apiUrl}${id}/`;
26       return this.http.get(url);
27     }
28
29     getUserById(userId: number): Observable<any> {
30       return this.http.get<any>(`http://127.0.0.1:8000/usuarios/usuarios/${userId}/`); // Asegúrate de c
31     }
32
33     getCuartoById(tipoId: number): Observable<any> {
34       return this.http.get<any>(`http://127.0.0.1:8000/reservaciones/cuartos/${tipoId}/`); // Asegúrate
35     }
36
37   }
38
39
```

12. Componente ts de app.

```
PROYECTO_FINAL_PWEB > frontEnd > src > app > Ts app.component.ts > ...
1  import { Component } from '@angular/core';
2  import { RouterOutlet } from '@angular/router';
3
4
5  @Component({
6    selector: 'app-root',
7    standalone: true,
8    imports: [RouterOutlet],
9    templateUrl: './app.component.html',
10    styleUrls: ['./app.component.css']
11  })
12  export class AppComponent {
13    title = 'frontend';
14  }
15
16
```

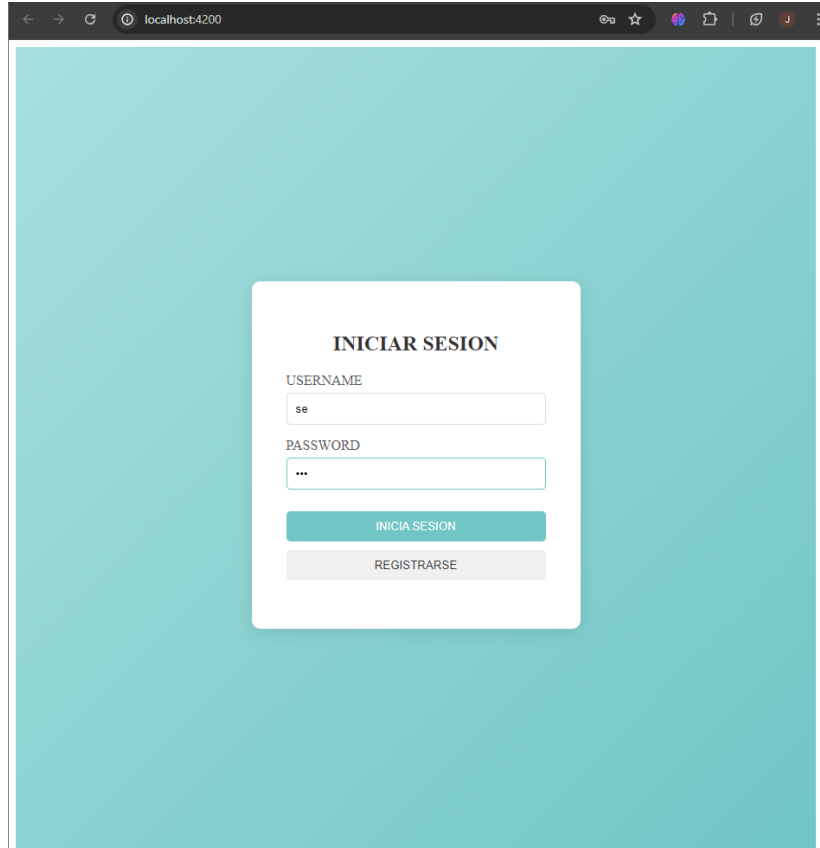
13. Rutas environment

```
PROYECTO_FINAL_PWEB > frontEnd > src > enviroments > TS environment.prod.ts > ...  
1 export const environment = {  
2   production: true,  
3   apiUrlReservas: 'https://api.miapp.com/reservaciones',  
4   apiUrlUsuarios: 'https://api.miapp.com/usuarios'  
5 };
```

```
PROYECTO_FINAL_PWEB > frontEnd > src > enviroments > TS environment.ts > ...  
1 export const environment = {  
2   production: false,  
3   apiUrlReservas: 'http://localhost:8000/reservaciones',  
4   apiUrlUsuarios: 'http://localhost:8000/usuarios',  
5 };
```

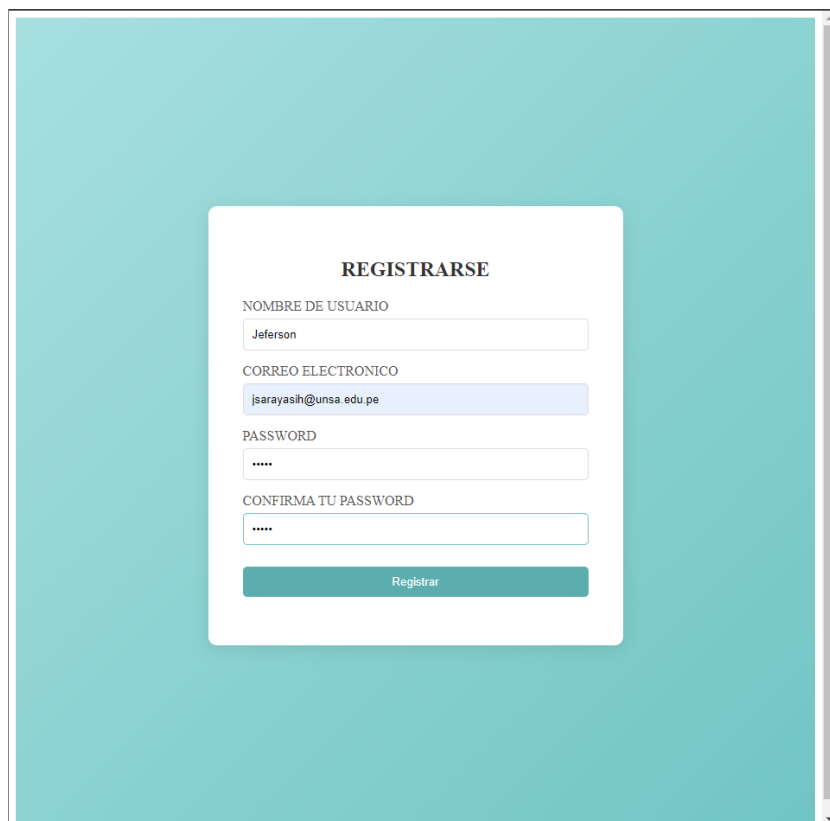
9. Corriendo el servidor

1. Prueba de Conexion - Login



The screenshot shows a web browser window with the address bar displaying 'localhost:4200'. The main content area has a teal background. In the center, there is a white card with the title 'INICIAR SESION'. Below the title, there are two input fields: 'USERNAME' with the text 'se' and 'PASSWORD' with three dots. Below these fields are two buttons: a teal button labeled 'INICIA SESION' and a light gray button labeled 'REGISTRARSE'.

2. Prueba de Conexion - Register



REGISTRARSE

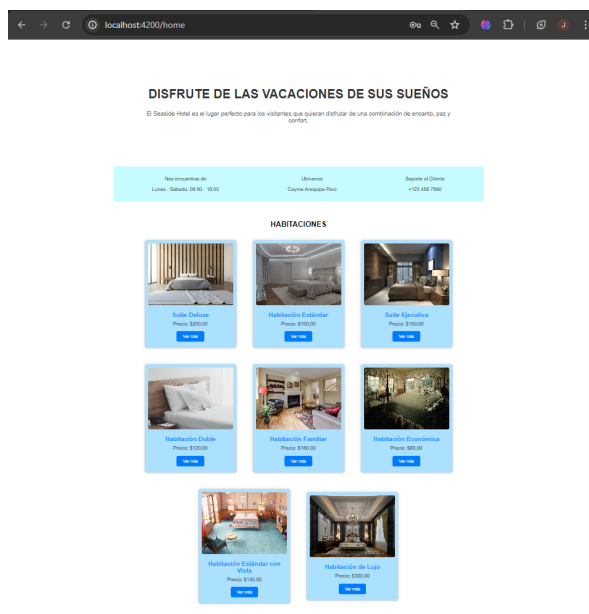
NOMBRE DE USUARIO

CORREO ELECTRONICO


PASSWORD

CONFIRMA TU PASSWORD

3. Prueba de Conexion - home



4. Prueba de Conexion - detalles



Habitación Familiar

Descripción: Habitación espaciosa para toda la familia, con una cama doble y dos individuales.

Precio: 180.00

Usuarios:

ID	Nombre de Usuario
1	ser
2	se
3	Referenc

INGRESE EL ID DEL USUARIO


FECHA DE INICIO

FECHA DE FIN

HACER RESERVACION

ATRÁS

5. Prueba de Conexion - reservacion



Habitación Familiar

Descripción: Habitación espaciosa para toda la familia, con una cama doble y dos individuales.

Precio: 180.00

Usuarios:

ID	Nombre de Usuario
1	ser
2	se
3	Referenc

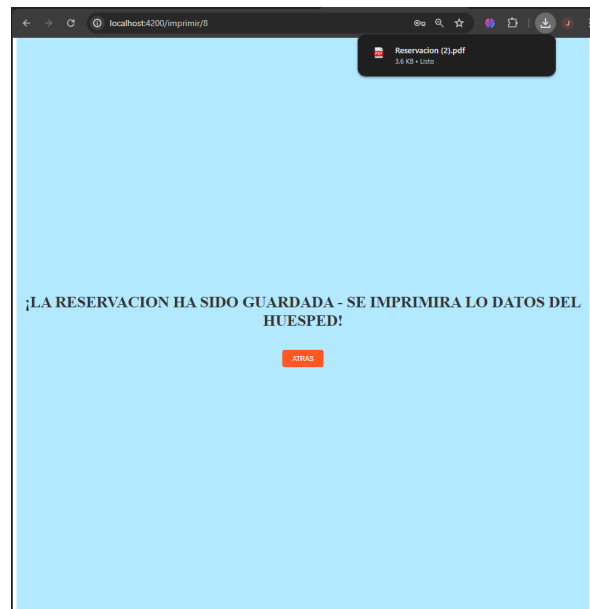
agosto de 2024

L	M	X	J	V	S	D	23	24
29	30	31	1	2	3	4	25	26
5	6	7	8	9	10	11	27	28
12	13	14	15	16	17	18	29	30
19	20	21	22	23	24	25	01	02
26	27	28	29	30	31	1	03	04
2	3	4	5	6	7	8	05	06

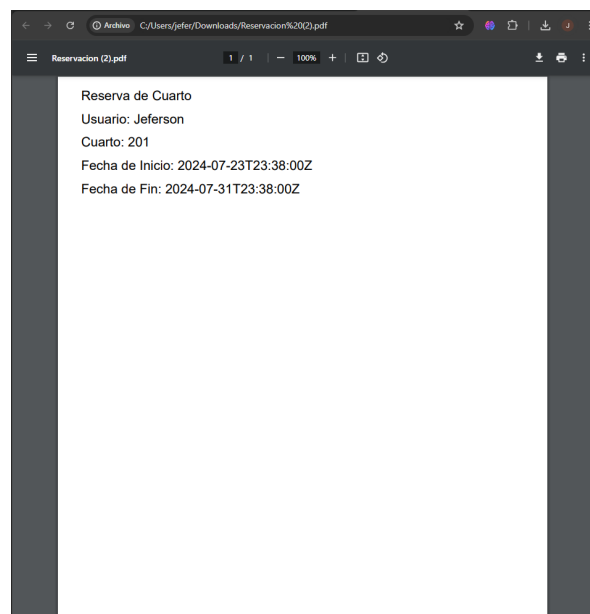
HACER RESERVACION

ATRÁS

6. Prueba de Conexion - imprimir



7. Prueba de Conexion - PDF descargable



10. Referencias

- Aprende ANGULAR 17 desde cero para principiantes GRATIS <https://www.youtube.com/watch?v=f7unUpshmpA&t=180s>
- Aprende Angular en 15 Minutos <https://www.youtube.com/watch?v=1x9Yy6Vp0tg>