TEC.AZUAY
INSTITUTO UNIVERSITARIO

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

TENDENCIAS ACTUALES DE LA PROGRAMACIÓN

Ciclo:

Nombres Completos:

CRITERIO	DESCRIPCIÓN	EVIDENCIA(FOTOCOPIA)
Define la Entidad y un Repositorio	Crea la entidad con los atributos validados correctamente	

```java
package com.examenfinal.kevinexamenfinal.modelo;

import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.List;


@Document(collection ="restaurante")
@Data
public class Restaurante {

    @Id
    private Long id;
    private String nombre;
    private String telefono;
    private String correo;
    private String direccion;
    private List<Cliente> clienteList;
    private Carta carta;


}
```

```java
package com.examenfinal.kevinexamenfinal.modelo;

import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "plato")
@Data
public class Plato {

    @Id
    private Long idPlato;
    private String nombrePlato;
    private String descripcion;
    private double precio;

}
```

```java
package com.examenfinal.kevinexamenfinal.modelo;

import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "cliente")
@Data
public class Cliente {

    @Id
    private Long idCliente;
    private String nombreCliente;
    private String apellidoCliente;
    private String cedula;
    private String correo;

}
```

```java
package com.examenfinal.kevinexamenfinal.modelo;

import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.List;

@Document(collection = "carta")
@Data
public class Carta {

    @Id
    private Long idCarta;
    private String nombreCarta;
    private String categoria;
    private int cantidad;
    private List<Plato> platoList;
}
```

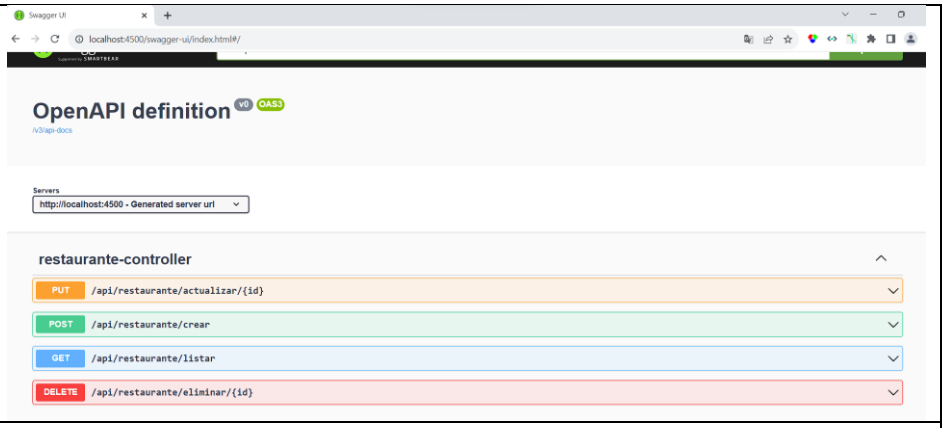
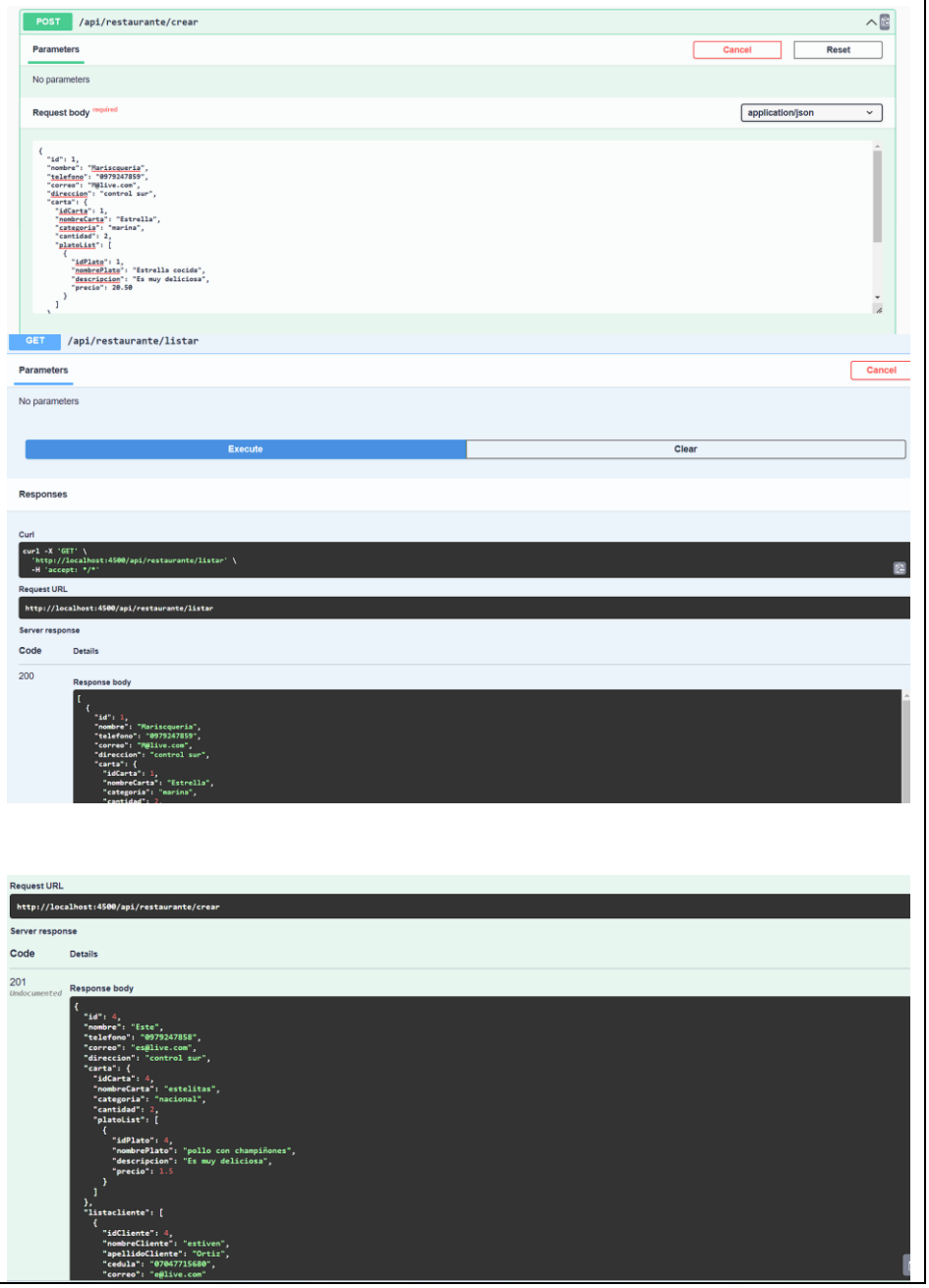
	Define y extiende la interfaz del Repositorio correctamente	

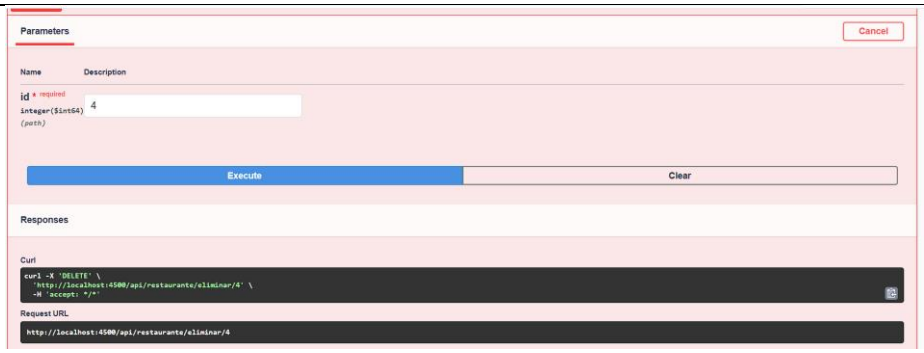
```java
package com.examenfinal.kevinexamenfinal.repository;

import com.examenfinal.kevinexamenfinal.modelo.Restaurante;
import org.springframework.data.mongodb.repository.MongoRepository;

public interface RestauranteRepository extends MongoRepository<Restaurante,Long> {

}
```

Define la Clase Controladora	Crea la clase controlador con los métodos GET, POST, PUT, DELETE correctamente	

```java
 */
@RestController
@RequestMapping("/api/restaurante")
public class RestauranteController {

    @Autowired
    RestauranteServiceImpl restauranteService;

    @GetMapping("/listar")
    public ResponseEntity<List<Restaurante>> listarProductos() {
        return new ResponseEntity<>(restauranteService.findByAll(),
                HttpStatus.OK);
    }

    @PostMapping("/crear")
    public ResponseEntity<Restaurante> crearProducto(
            @RequestBody Restaurante p) {
        return new ResponseEntity<>(restauranteService.save(p),
                HttpStatus.CREATED);
    }

    @PutMapping("/actualizar/{id}")
    public ResponseEntity<Restaurante> actualizarProducto(@PathVariable Long id, @RequestBody Restaurante p) {
        Restaurante restaurante = restauranteService.findById(id);
        if (restaurante == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        } else {
            try {
                restaurante.setNombre(p.getNombre());
                restaurante.setTelefono(p.getTelefono());
                restaurante.setCorreo(p.getCorreo());
                restaurante.setDireccion(p.getDireccion());
                restaurante.setListacliente(p.getListacliente());
                restaurante.setCarta(p.getCarta());

                return new ResponseEntity<>(restauranteService.save(p), HttpStatus.OK);
            } catch (DataAccessException e) {
                return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
            }
        }
    }

    @DeleteMapping("/eliminar/{id}")
    public ResponseEntity<Restaurante> eliminarProducto(@PathVariable Long id) {
        restauranteService.delete(id);
        return new ResponseEntity<>(HttpStatus.OK);
    }

}
```

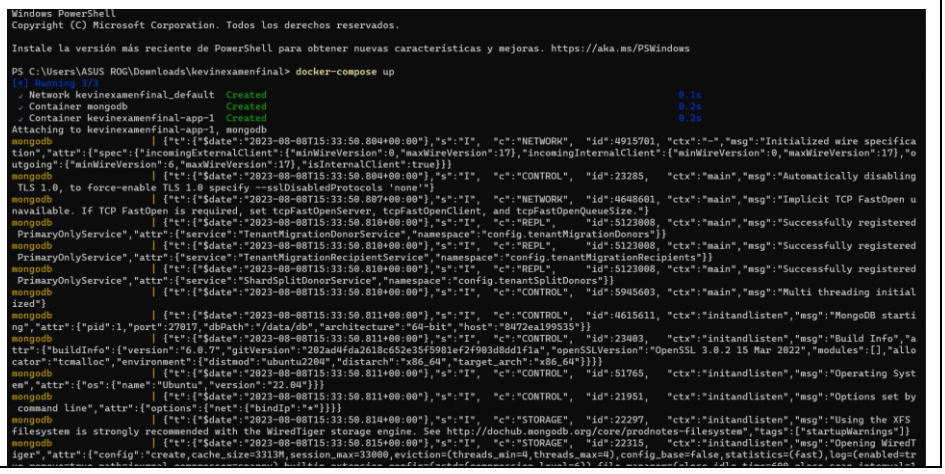
Puerto del swagger: 4500

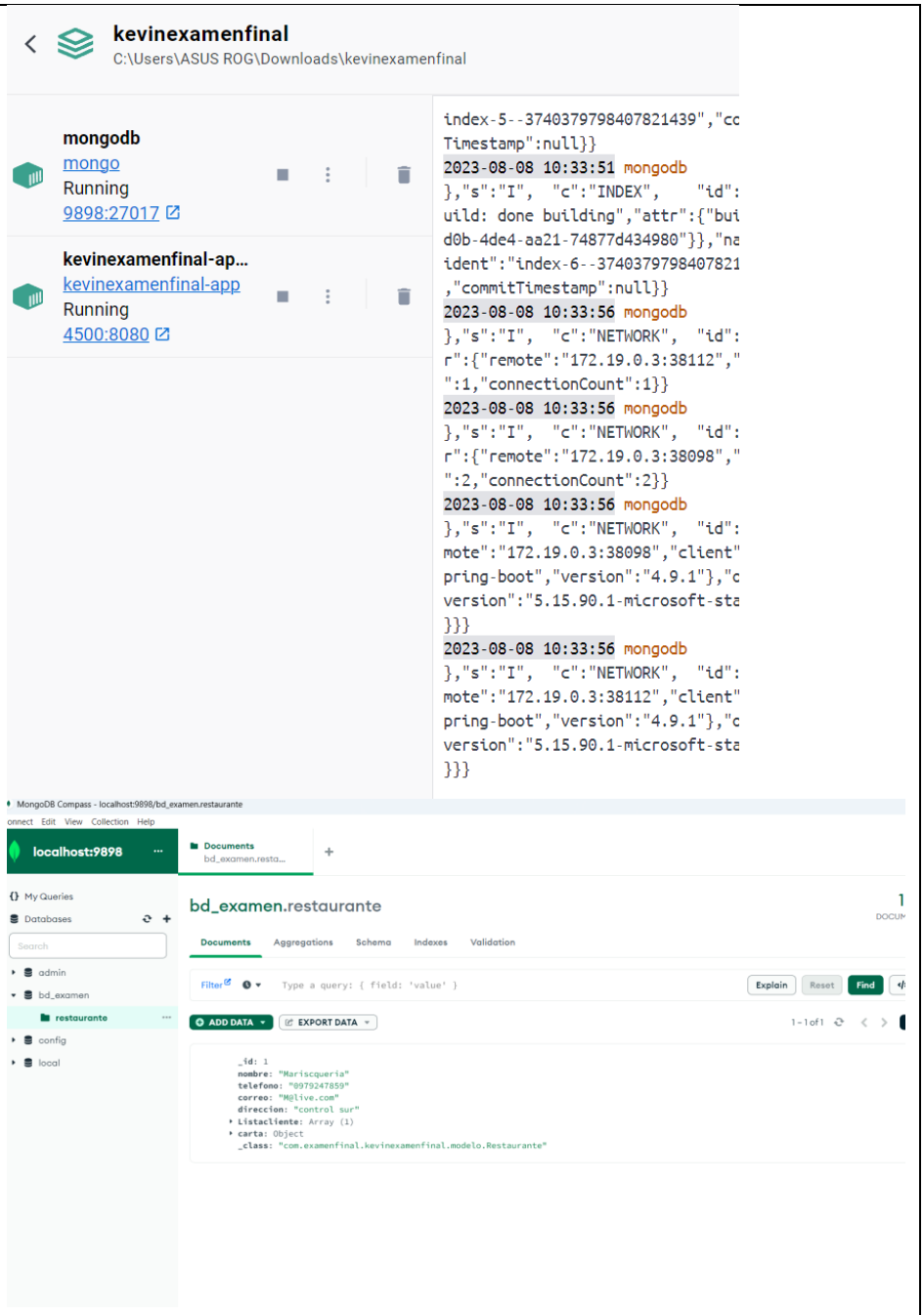
		
Crea el microservicio	Crea el microservicio e implementa los métodos de las operaciones CRUD correctamente	

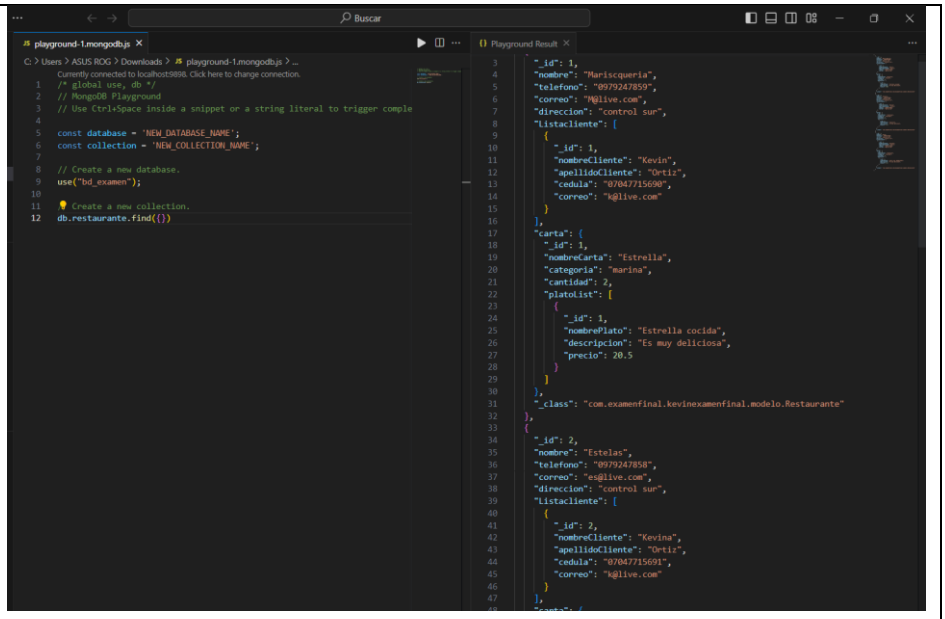
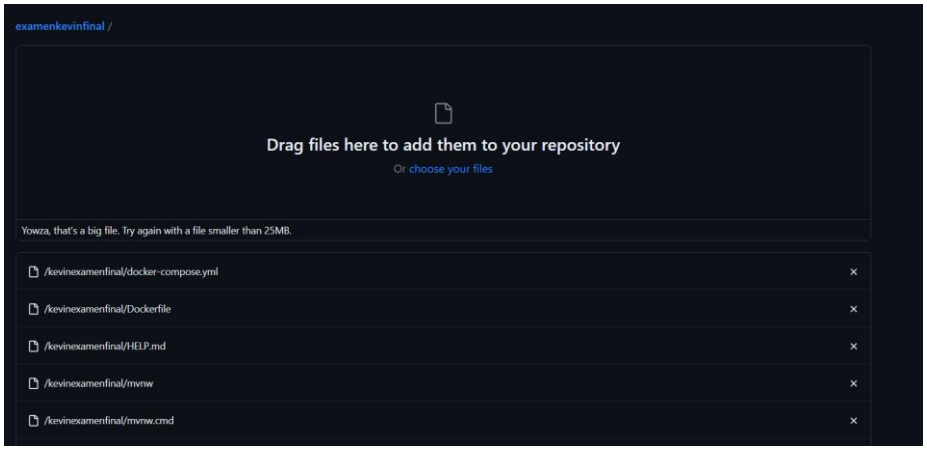
		 Parameters Cancel Name Description id * required integer($int64) (path) 4 Execute Clear Responses Curl curl -X 'DELETE' \ 'http://localhost:4500/api/restaurante/eliminar/4' \ -H 'accept: */*' Request URL http://localhost:4500/api/restaurante/eliminar/4

Docker, Mongo y consultas	Realiza la creación de la Bd, y las consultas NoSql respectivas, con el puerto respectivo.	

```yaml
version: '3.3'
services:
  app:
    build: .
    ports:
      - 4500:8080
    depends_on:
      - mongo
    environment:
      SPRING_DATA_MONGODB_HOST: mongo
      SPRING_DATA_MONGODB_PORT: 27017
      SPRING_DATA_MONGODB_DATABASE: bd_examen

  mongo:
    image: mongo
    container_name: mongodb
    restart: always

    ports:
      -  9898:27017


volumes:
  mongo-data:
```



		
Scripts de Docker y PlayGround	Construir los archivos de docker, y Generar el código de consulta MngoDB.	

		
Documenta las pruebas del microservicio en Swagger	Documenta las pruebas del microservicio en swagger con los métodos de validación solicitados	

Despliega el código fuente en GitHub y en Docker Hub	Despliega el código fuente en GitHub y en Docker Hub	