@Entity =This annotation indicates that the class is a JPA entity representing a table in the database.

@EntityListeners(AuditingEntityListener.class): Specifies that the entity should be audited using the AuditingEntityListener class, which helps manage created and last modified dates.

@Id: Indicates that the field following this annotation is the primary key of the entity.

@GeneratedValue(strategy = GenerationType.IDENTITY): Specifies the strategy for generating primary key values. In this case, it uses an identity column (auto-increment) to generate unique values.

@CreatedDate: Indicates that this field will store the creation timestamp of the entity.

@LastModifiedDate: Indicates that this field will store the timestamp of the last modification to the entity.

@Temporal(TemporalType.TIMESTAMP): Specifies the temporal type of the date fields. In this case, it represents a full timestamp including date and time.

@Column(name = "created\_at", nullable = false, updatable = false): Maps the field to the "created\_at" column, which represents the creation timestamp. It is marked as not nullable and not updatable.

@Repository:indicate that it's a Spring-managed repository component.

@service : marks this class as spring service, allow to use spring framework

@Autowired : is used for automatic dependency injection

@Bean: This annotation is applied to the myBean method, indicating that it will create and configure a Spring bean.

@RestController: Indicates that this class RESTFULL controller that handles incoming HTTP requests and produces responses.

@RequestMapping: Maps HTTP requests to controller methods.

SecurityContextHolder is used to retrieve authentication information.

@RequestBody annotation is used to accept json data in the request body

**Value**: Injects values from property files or environment variables.

**@Configuration**: Indicates that a class contains Spring bean configuration.

**@Service**, **@Repository**, **@Component**: Annotate classes to be automatically discovered and registered as Spring beans.

@EnableDiscoveryClient :Allows the services to register themselves to the consul agent

@EnableCircuitBreaker : enables a circuit breaker functionality for handling service failure.

@FeignClient : creates feign client for making http request to other microservices

@LoadBanced : enables client-side load balancing

@EnableZuulProxy : Enables API gateway functionality using zuul

@EnableConfiServer : enable spring cloud config server

@EnableEurekeServer : Enables eureka server functionality

@EnableEurekaClient : enable as a eureka server and microservices will register itself with eureka server

@RefreshScope : allows dynamic reloading when configuration properties changes

@Runwith and @SpringRunner :used for Junit testing

**@Slf4j**, **@Log4j2**, **@Log**: Used for adding logging capabilities to classes.