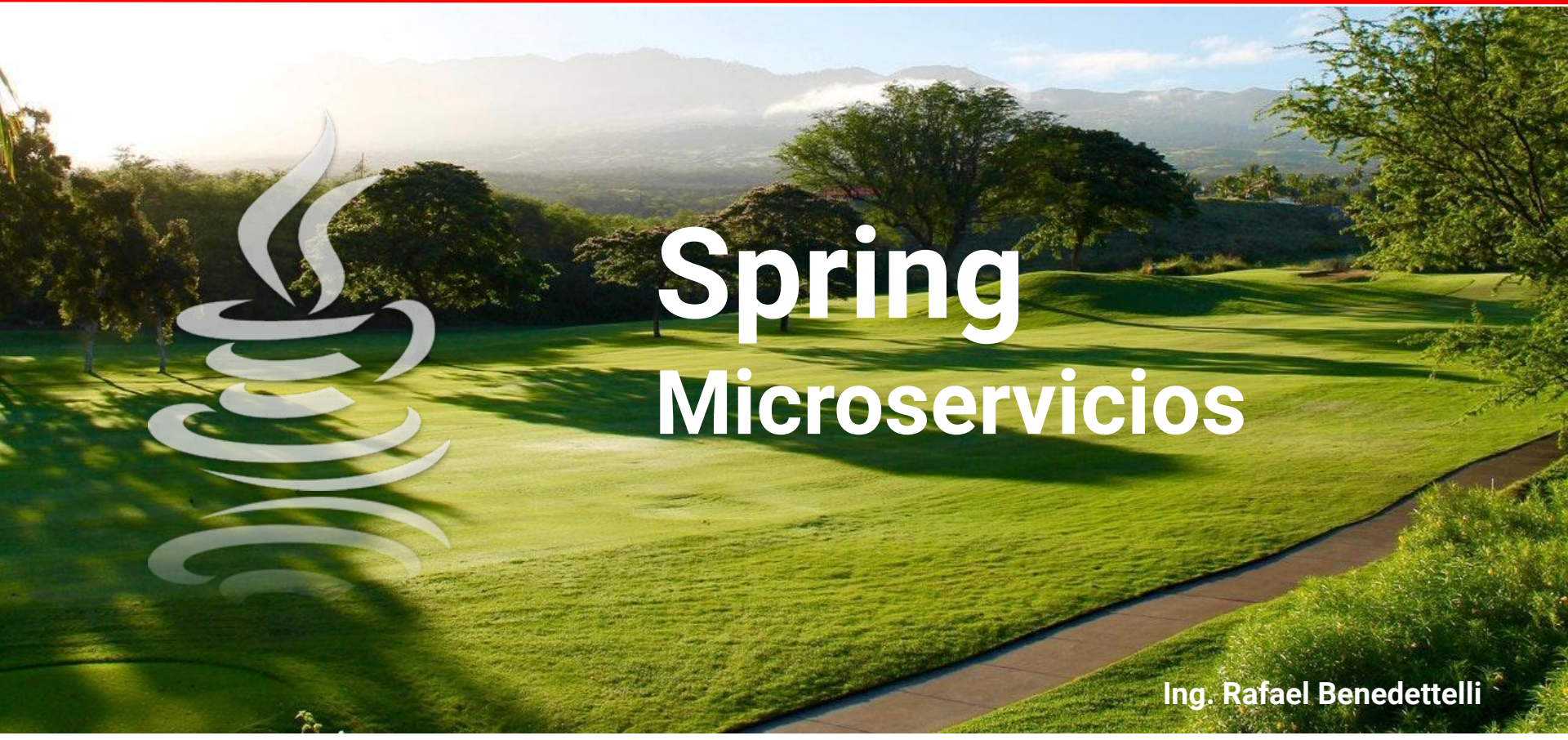

Spring
Microservicios

Ing. Rafael Benedettelli

CODIGOS DE RESPUESTAS HTTP

Rest Persona

spring boot

		Código Errores HTTP
getById	GET	200 - OK
listAll	GET	200 - OK
create	POST	201 - CREATED
update	PUT	200 o 204 - OK
delete	DELETE	200 o 204 - OK

Rest User (Not Found)

getById	→	GET	404 - OK
listAll	→	GET	404 - OK
create	→	POST	404 - OK
update	→	PUT	404 - OK
delete	→	DELETE	404 - OK

Obtener URI:

```
URI location = ServletUriComponentsBuilder.
                                        fromCurrentRequest()
                                        .path("/{id}")
                                        .buildAndExpand(userDTO.getId())
                                        .toUri();
```

Códigos de error HTTP ResponseEntity

200 - OK
201 - Created
400 - Bad Request
401 - Unauthorized
404 - Not Found
500 - Internal error server

Not Found
Recurso no encontrado

Anidación de recursos

users/{id}/accounts

Rest Buenas Prácticas

Rest Buenas Practicas

REST es orientado a resources

- Usar sustantivos (no verbos)
- Usar plural

POST **/usuarios** 

/crearUsuarios 

Rest Buenas Practicas

REST concatenación de recursos de manera apropiada

GET	**/usuarios/**	(todos los usuarios)
GET	**/usuarios/{id}**	(el usuario con {id}
GET	**/usuarios/{id}/accounts/**	Todas la cuentas del usuario con id
GET	**/usuarios/{id}/accounts/{idAcc}**	La cuenta con id del usuario id

Rest Buenas Practicas

Entregar codigos HTTP adecuados

No mas de tres niveles de resources

equipos/{1}/jugadores/{1}/estadisticas/

JSON claros y no mas de 3 niveles

```
{
nombre: “Real Madrid”
jugadores: [
  {nombre: “Benzema”, estadisticas: [
                .....
        ]}
]
}
```

La cuenta con id del usuario id

Rest Buenas Practicas

Evitar!!!

/equipos/1/jugadores/1/partidos/2/goles/1/multimedia/1

/users/1/accounts/1/orders/1/products/1/

Rest Buenas Practicas

No entregar JSON “Enormes”

- Paginar
- Sortear
- Filtrar
- Crear un rest query con filters

Rest Buenas Practicas

- Documentar la API
  - Open API - Swagger
- Generar links de navegación
  - HATEOAS
- Asegurar las apis definiendo roles

Rest Buenas Practicas

- Exportar clientes de demostración
  - Ej: POSTMAN
- Entregar errores definidos y verbosos
- Validación de campos
- Versionado apropiado de la API

Spring Boot Rest
Aspectos Avanzados

Open API - Swagger
SpringDocs

Activar Swagger

1

```
<dependency>
  <groupId>org.springdoc</ groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</ artifactId>
  <version>2.0.2</version>
</dependency>
<dependency>
  <groupId>org.springdoc</ groupId>
  <artifactId>springdoc-openapi-ui</ artifactId>
  <version>1.6.14</version>
</dependency>
```

2

http://localhost:8080/swagger-ui/index.html

Agregar carátula

3

```
import io.swagger.v3.oas.annotations. OpenAPIDefinition;
import io.swagger.v3.oas.annotations.info. Contact;
import io.swagger.v3.oas.annotations.info. Info;
import io.swagger.v3.oas.annotations.info. License;
import io.swagger.v3.oas.annotations.servers. Server;

@OpenAPIDefinition (
        info = @Info(
                title = "Tienda Escuela IT API",
                description = "Tienda Escuela IT Microservicio",
                contact = @Contact(
                        name = "Rafael Benedettelli",
                        url = "http://esculeait.com",
                        email = "rblbene@gmail.com"
                ),
                license = @License(
                        name = "MIT Licence",
                        url = "https://github.com/thombergs/code-examples/blob/master/LICENSE"
        servers = @Server(url = "http://localhost:8080")
)
public class SwaggerConfiguration {

}
```

Documentar API

4

Documentar Rest Controller

```
@Tag(name = "API personas",
description = "CRUD de personas de
tienda")
```

Documentar API

5

Documentar Endpoint

```
@Operation(summary = "Recupera una persona
por Id", description = "Recupera unan
persona dado un id de tipo numérico")
```

Documentar API

5

Documentar Endpoint: Respuestas

```
@ApiResponse(responseCode = "200", description = "Operación exitosa")
@ApiResponse(responseCode = "400", description = "Error de petición")
@ApiResponse(responseCode = "404", description = "Recurso no encontrado")
```

Documentar API

7

Documentar Parámetros de endpoint

```
@Parameter(description="Id de persona.
Valor entero", required=true,example =
"1")
```

Documentar API

7 Documentar Body de endpoint

```
@io.swagger.v3.oas.annotations.parameters.RequestBody(descripti
on = "JSON de persona completo",content = @Content(schema =
@Schema(implementation = Persona.class)))
```

Documentar API

8 **Documentar Esquema - clase y Atributos**

```
@Schema(description = "Esta es una Persona.")
```

```
@Schema(description = "Identificador único de
persona.",
example = "1”)
```

Instructor Ing. Rafael Benedettelli

Java Bean Validation 2.0

JSR 380

Spring debemos explicitar la dependencia

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

JSR 380

Validators

@Validated

@NotNull
@NotBlank (“”)
@Size (string)
@Min (int)
@Max (int)
@Regex()
@AssertTrue (bool)

Nuevos Validators in JSR 380

@Validated

@Email
@Positive
@PositiveOrZero
@Negative
@NegativeOrZero
@PastOrPresent
@FutureOrPresent

Custom Error Messages

(RestControllerExceptionHandler)

```
@Override
protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
HttpHeaders headers, HttpStatus status, WebRequest request) {

   Map<String, Object> body = new LinkedHashMap<>();
   body.put("timestamp", LocalDateTime.now());
   body.put("message",
ex.getBindingResult().getFieldError().getDefaultMessage());
   body.put("Error", HttpStatus.BAD_REQUEST.toString());

   return new ResponseEntity<>(body, HttpStatus.BAD_REQUEST);
}
```

Custom Validators

ConstraintValidator

Annotation

Custom Validators

Instructor Ing. Rafael Benedettelli

```java
import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

public class CUITValidator implements
ConstraintValidator<Cuit, String>{

 @Override
 public boolean isValid(String value,
ConstraintValidatorContext context) {

    if (value == null) {
       return false;
    }

    return value.length()==13;
 }

}
```


```java
import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.*;

@Documented
@Constraint(validatedBy = CUITValidator.class)
@Target( { ElementType.FIELD })
@Retention(RetentionPolicy.RUNTIME)
public @interface Cuit {
  String message() default "Invalido número CUIT";
  Class<?>[] groups() default {};
  Class<? extends Payload>[] payload() default {};
}
```

Validación por grupos

```
public interface OnUpdate{}
public interface OnCreate{}
@Validate(OnUpdate.class)
@NotNull(groups=OnUpdate.class)
```

HATOEAS

```
EntityModel<Cliente> clienteEntityModel = EntityModel.of(cliente,

linkTo(methodOn(ClienteControllerRest.class).getClienteById(id)).withSelfRel(),

linkTo(methodOn(ClienteControllerRest.class).listAllClientes()).withRel("clientes"));
```

Cache

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-cache</artifactId>
  <version>2.4.0</version>
</dependency>
```

Cache

```
import org.springframework.cache.CacheManager;
import org.springframework.cache.annotation.EnableCaching;
import org.springframework.cache.concurrent.ConcurrentMapCacheManager;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableCaching
public class CachingConfig {

   @Bean
   public CacheManager cacheManager() {
       return new ConcurrentMapCacheManager("clientes");
   }
}
```

Cache

```
@Cacheable("clientes")
```

Vista JSON

```
@JsonProperty
@JsonIgnore
@JsonPropertyOrder
```