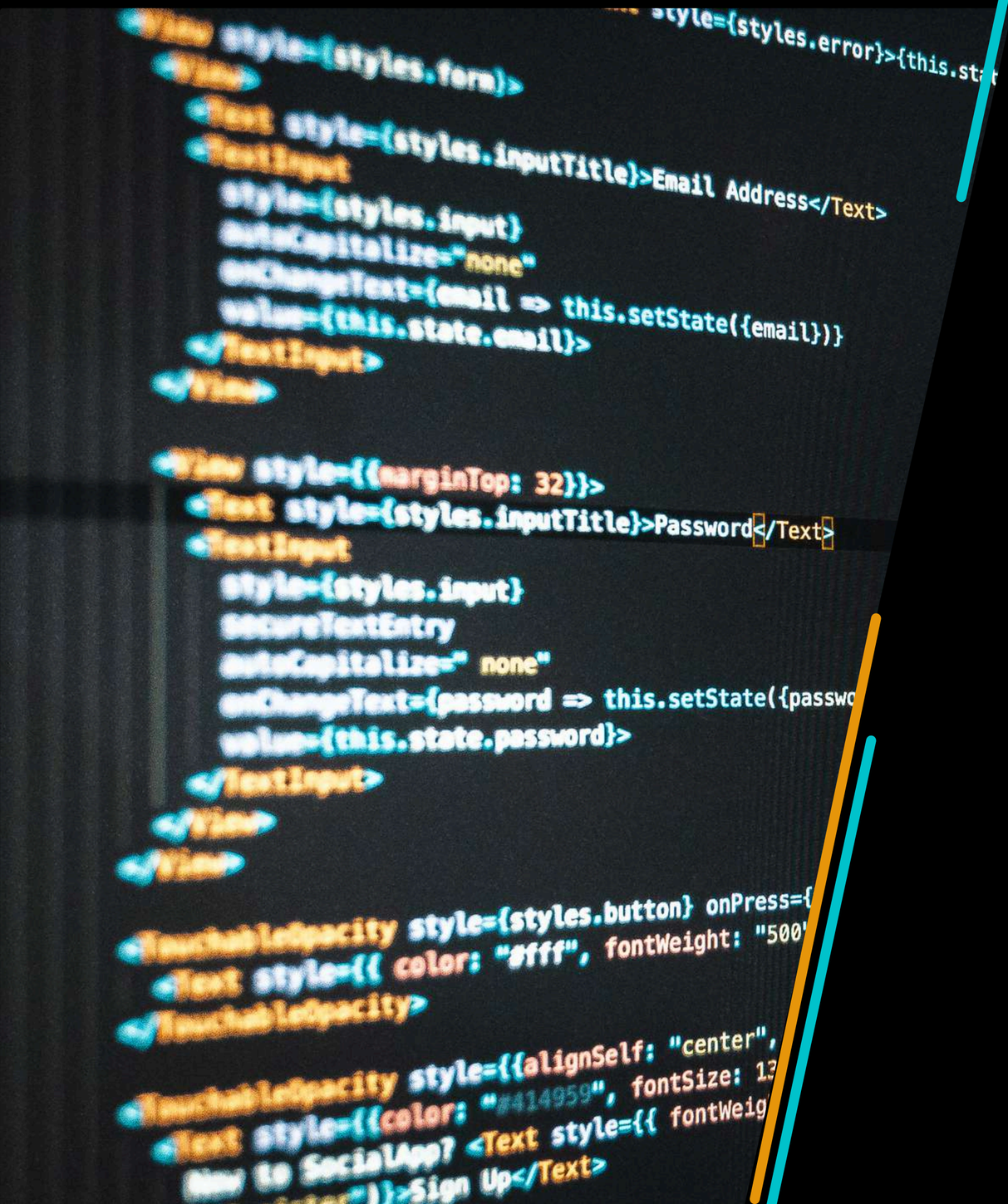
# VALIDATION API

PROGRAMACIÓN 3

PROF. MANGO EDUARDO




# CONTENIDO

# Introduccion


## ¿QUE ES?

En aplicaciones web desarrolladas con Spring Boot, la validación de datos es un aspecto fundamental para garantizar la integridad y seguridad de la información. Spring Web soporta la validación mediante la Java Validation API (Jakarta Bean Validation), que proporciona anotaciones para validar objetos de entrada.

Spring Boot integra esta API a través de la implementación **Hibernate Validator**, que permite definir reglas de validación de manera declarativa en las clases de modelo o DTOs.

# Dependencia


# Hibernate Validator

Si bien Spring Web ya incluye el soporte de validaciones, no incluye una implementación concreta de estas, por lo que debemos importar una dependencia que lo haga por nosotros. Por suerte, Spring nos ofrece un Starter de Validation, que incluye Hibernate Validation.

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

# Validaciones comunes

La API de validación proporciona diversas anotaciones para definir restricciones en los atributos de una clase. Algunas de las más utilizadas son:

- @NotNull: El campo no puede ser nulo.
- @NotEmpty: No permite valores null ni cadenas vacías.
- @NotBlank: No permite valores null, cadenas vacías ni espacios en blanco.
- @Size(min = X, max = Y): Define la longitud mínima y máxima de una cadena.
- @Min(value), @Max(value): Define valores mínimo y máximo para números.
- @Positive, @PositiveOrZero: Restringe valores numéricos positivos o cero.
- @Negative, @NegativeOrZero: Restringe valores numéricos negativos o cero.
- @Email: Valida que el campo tenga un formato de correo electrónico válido.
- @Pattern(regexp = "regex"): Valida que el campo cumpla con una expresión regular.

# Uso en DTO

```java
@Schema(description = "Data transfer object for creating a new user.")
public class NewUserDTO {

    @Schema(description = "Username for the new user", example = "john_doe", required = true)
    @NotBlank(message = "Username is required")
    @Length(min = 6, message = "Username must be at least 6 characters long")
    private String username;

    @Schema(description = "Email address of the user", example = "john.doe@example.com", required = true)
    @Email(message = "Email format is not valid",
            regexp = "^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^.-]+@[a-zA-Z0-9.-]+$")
    //Estandar RFC 5322 para validar emails
    @NotBlank(message = "Email is required")
    private String email;

    @Schema(description = "Password for the new user", example = "password123", required = true)
    @NotBlank(message = "Password is required")
    @Length(min = 6, message = "Password must be at least 6 characters long")
    private String password;
}
```

# Uso en Controladores

Para habilitar la validación en un controlador Spring, se usa la anotación @Valid o @Validated en los métodos que reciben datos del cliente.

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioController {

    @PostMapping
    public ResponseEntity<String> crearUsuario(@Valid @RequestBody
UsuarioDTO usuario) {
        return ResponseEntity.ok("Usuario creado exitosamente");
    }
}
```

# Validación con grupos

A veces, es necesario aplicar diferentes reglas de validación según el contexto (por ejemplo, creación vs. actualización de usuario). Para ello, se usan grupos de validación con @Validated.

Esto nos reduce considerablemente la cantidad de codigo, ya que hace que no sea necesario crear un DTO para cada caso, sino que se puede adaptar el mismo según el contexto.

# A nivel DTO

```java
public interface Crear {}
public interface Actualizar {}

public class UsuarioDTO {

    @NotNull(groups = Actualizar.class)
    private Long id;

    @NotBlank(groups = {Crear.class, Actualizar.class})
    private String nombre;
}
```

# A nivel Controlador

```java
@PostMapping
public ResponseEntity<String> crearUsuario(@Validated(Crear.class)
@RequestBody UsuarioDTO usuario) {
    return ResponseEntity.ok("Usuario creado");
}

@PutMapping
public ResponseEntity<String>
actualizarUsuario(@Validated(Actualizar.class) @RequestBody UsuarioDTO
usuario) {
    return ResponseEntity.ok("Usuario actualizado");
}
```

# VALIDATION API

PROGRAMACIÓN 3

PROF. MANGO EDUARDO



Universidad Tecnológica Nacional de Mar del Plata