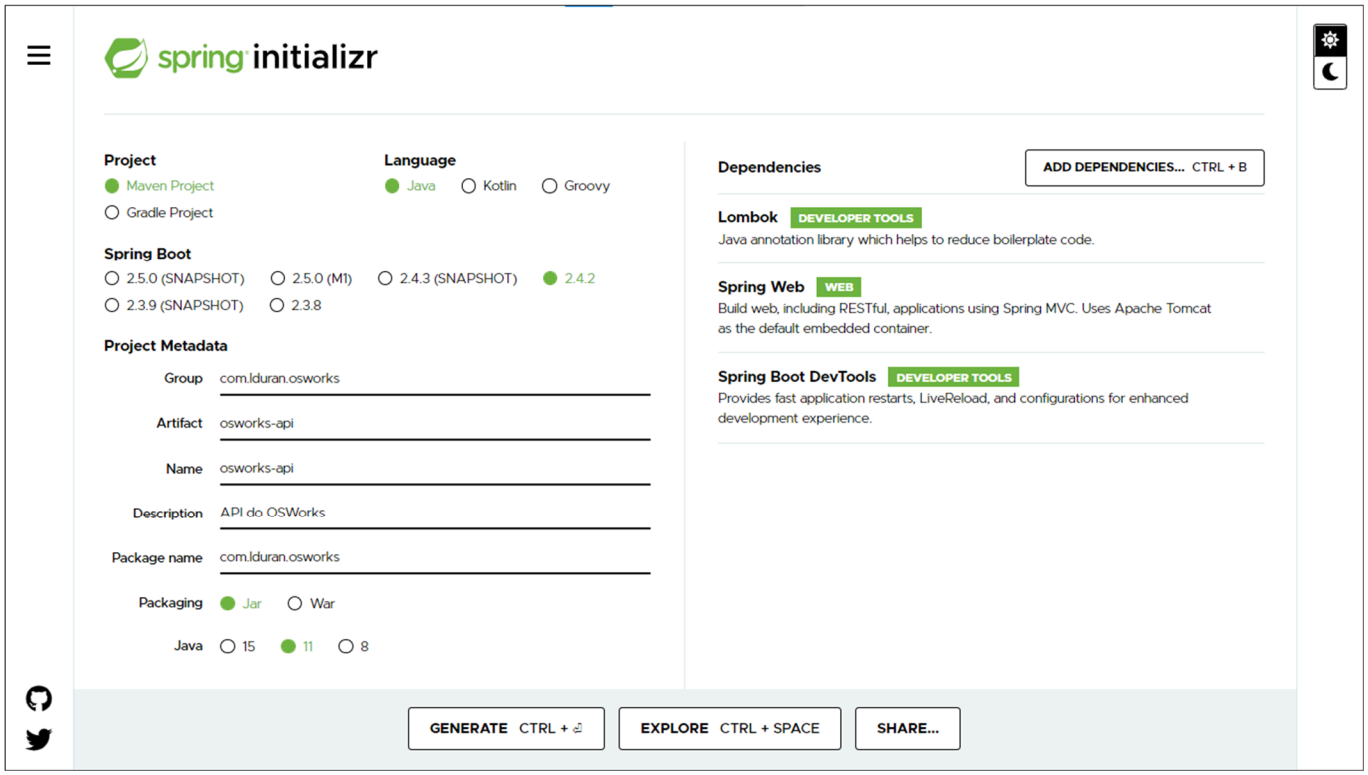
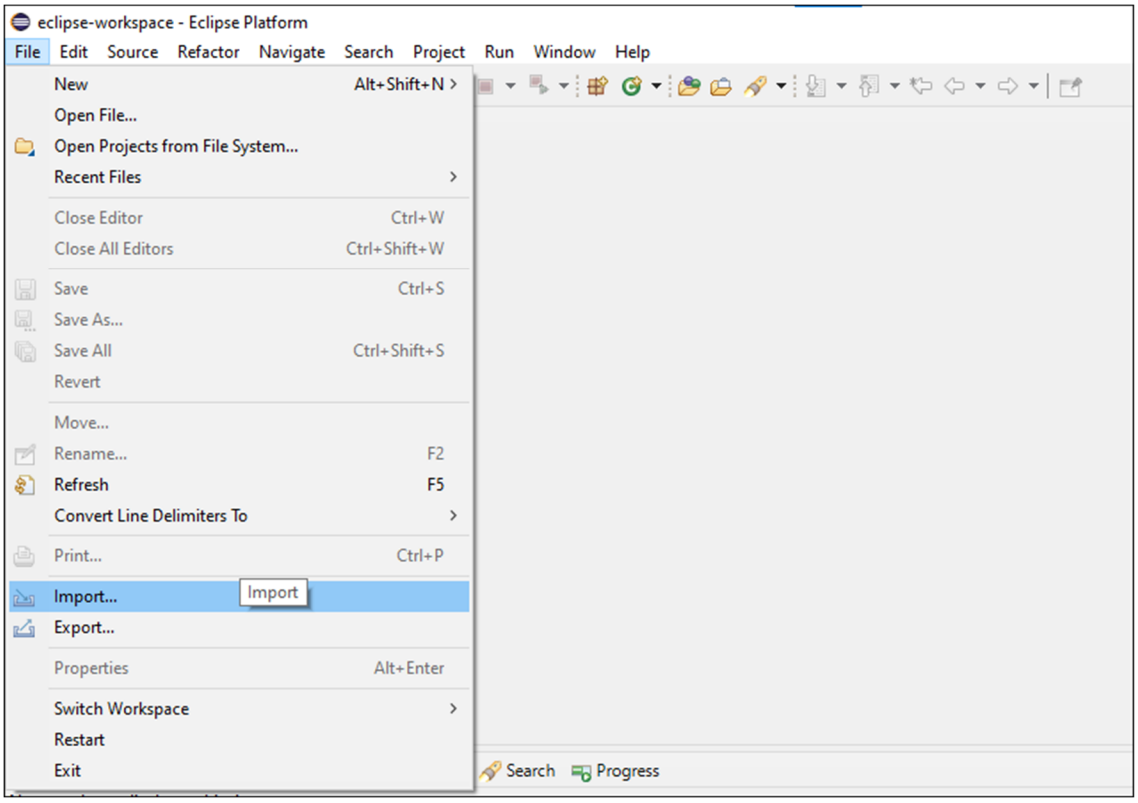
Spring REST para Iniciantes

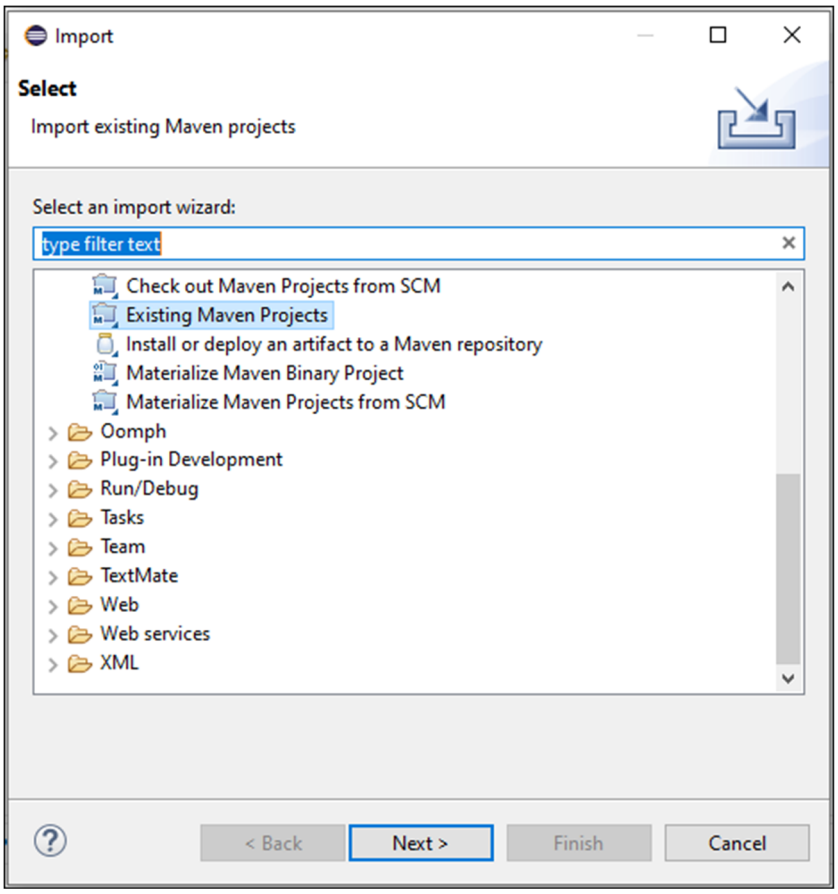
Implementando uma REST API com Spring

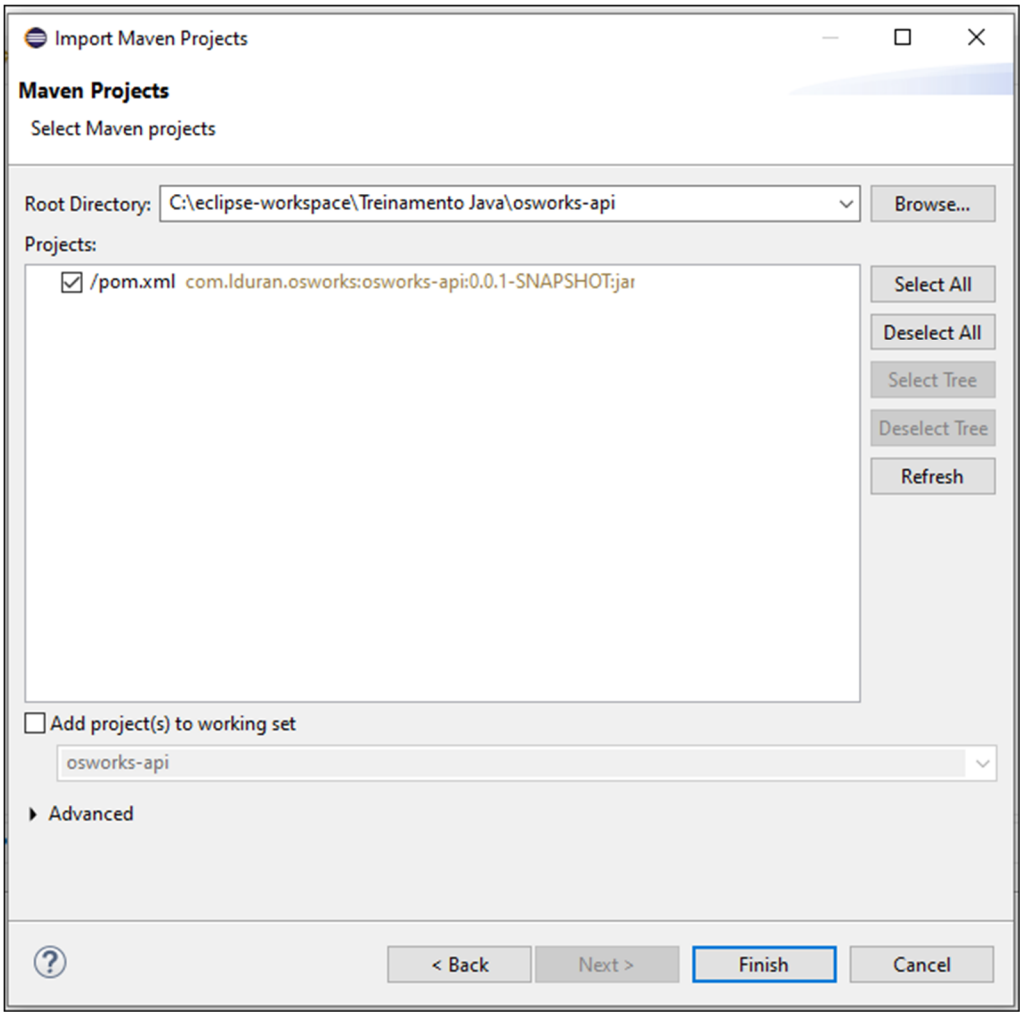
Criar o projeto inicial utilizando o [spring initializr]():



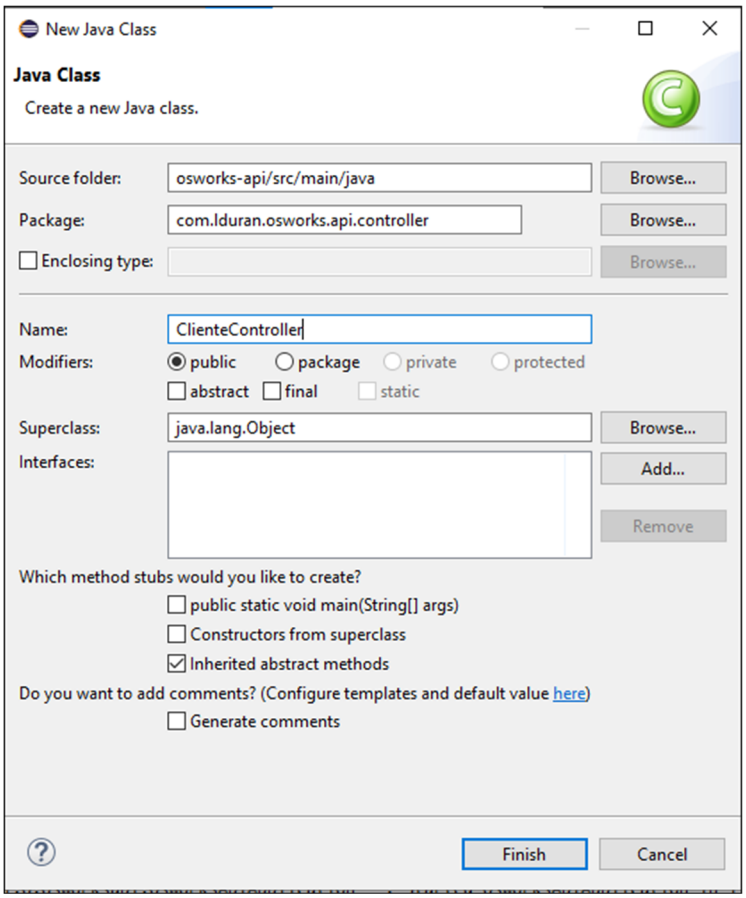
Gerar o pacote do projeto e importar no Eclipse:




Import
Select
Import existing Maven projects
Select an import wizard:
type filter text
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Binary Project
Materialize Maven Projects from SCM
Oomph
Plug-in Development
Run/Debug
Tasks
Team
TextMate
Web
Web services
XML
< Back
Next >
Finish
Cancel


Import Maven Projects
Maven Projects
Select Maven projects
Root Directory:
C:\eclipse-workspace\Treinamento Java\osworks-api
Browse...
Projects:
/pom.xml com.lduran.osworks:osworks-api:0.0.1-SNAPSHOT:jar
Select All
Deselect All
Select Tree
Deselect Tree
Refresh
Add project(s) to working set
osworks-api
Advanced
< Back
Next >
Finish
Cancel

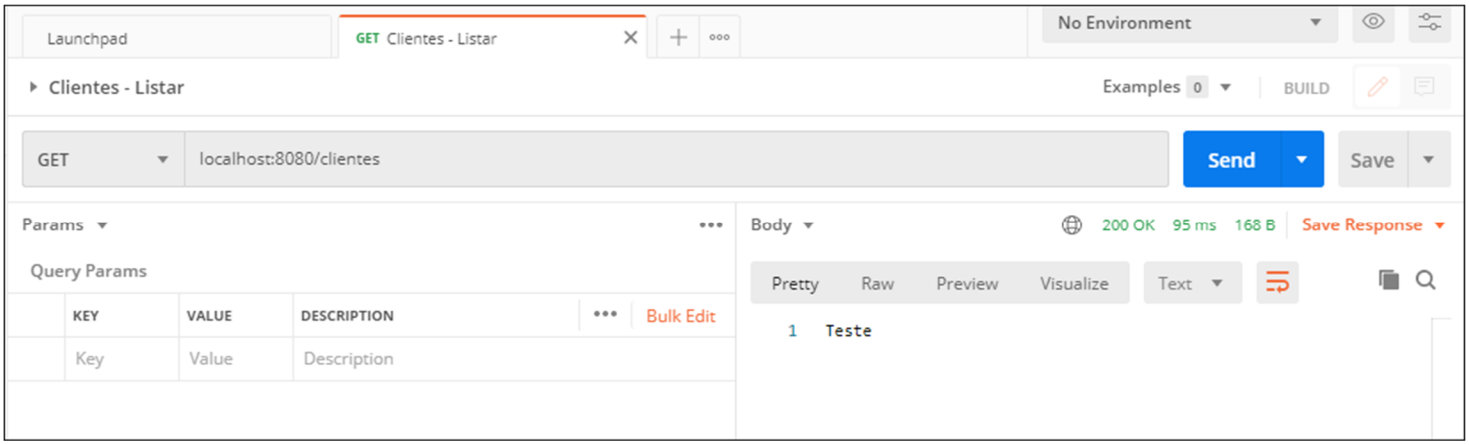
Criar o ClienteController:



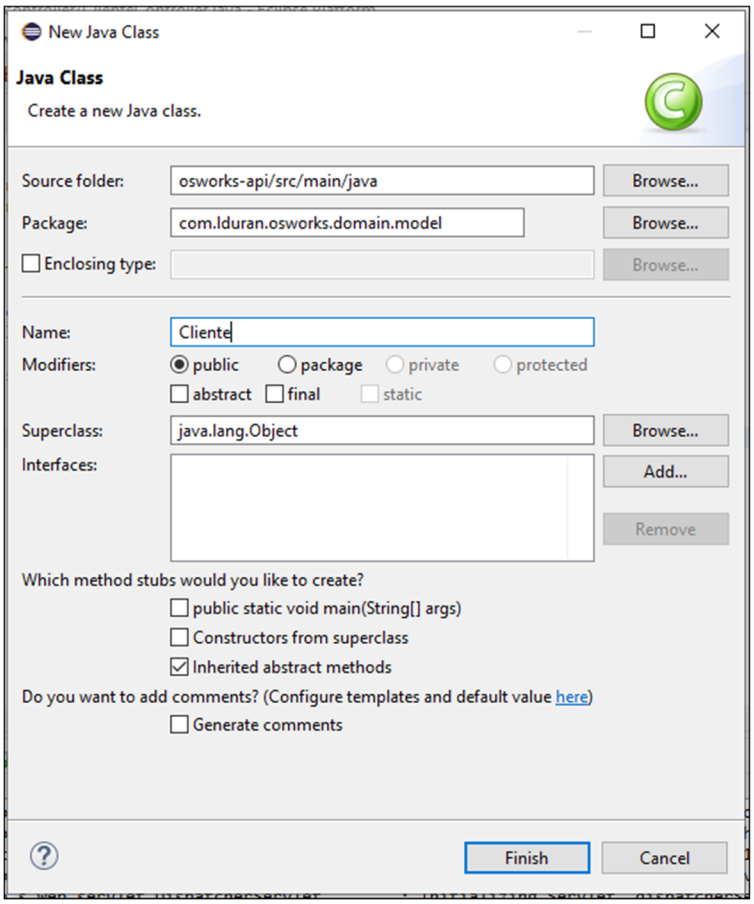
Criar um End-Point para listar clientes:

```
package com.lduran.osworks.api.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ClienteController
{
	@GetMapping("/clientes")
	public String listar()
	{
		return "Teste";
	}
}
```

Executar o projeto e testar no Postman:



Criar a entidade Cliente:



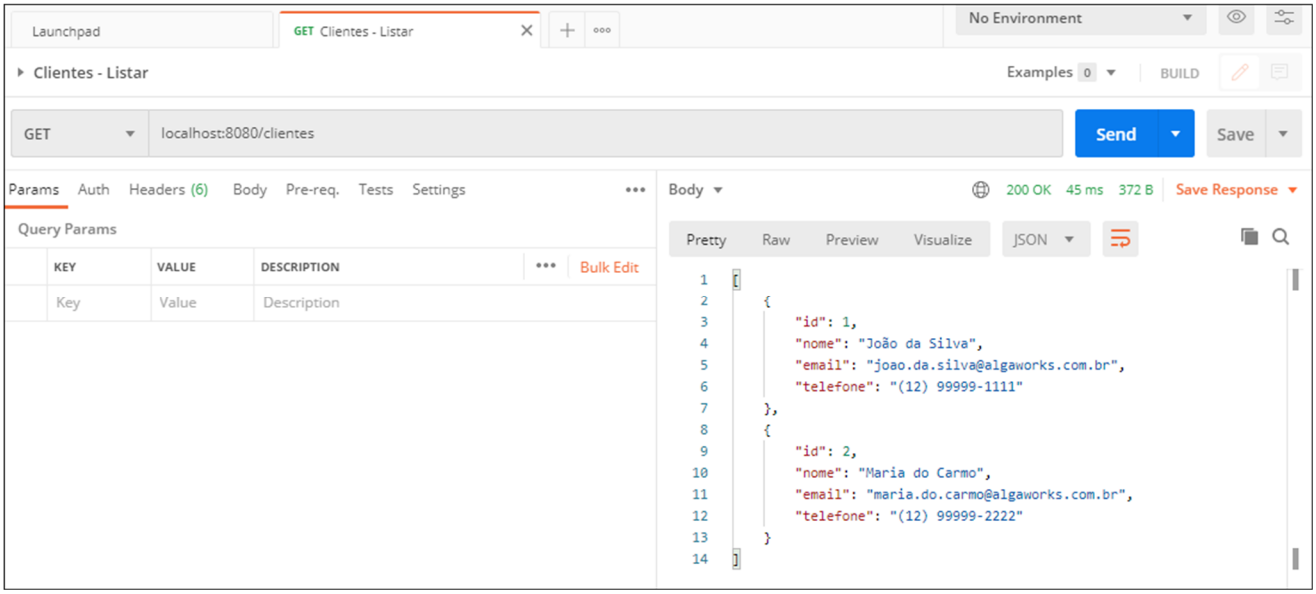

```java
package com.lduran.osworks.domain.model;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class Cliente
{
	private Long id;
	private String nome;
	private String email;
	private String telefone;
}
```

Criar uma lista de Clientes no Controller:

```java
package com.lduran.osworks.api.controller;

import java.util.Arrays;
import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;

@RestController
public class ClienteController
{
	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		var cliente1 = new Cliente();
		cliente1.setId(1L);
		cliente1.setNome("João da Silva");
		cliente1.setEmail("joao.da.silva@algaworks.com.br");
		cliente1.setTelefone("(12) 99999-1111");
```

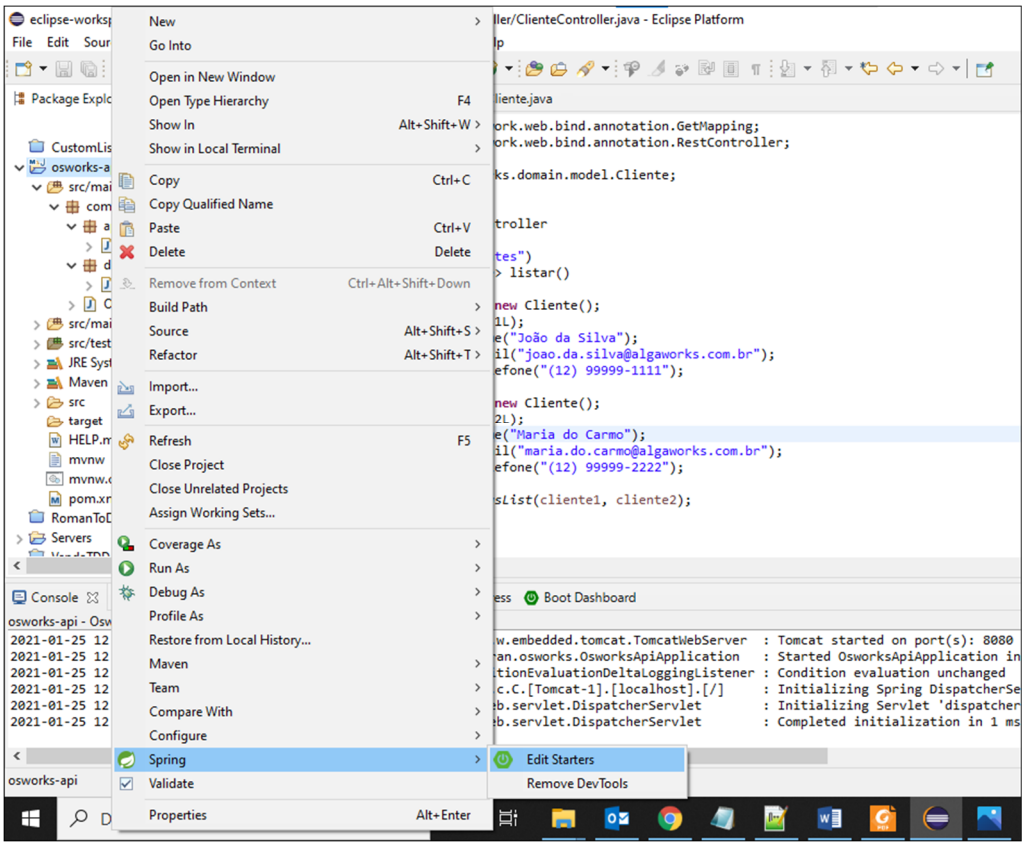
```
                var cliente2 = new Cliente();
                cliente2.setId(2L);
                cliente2.setNome("Maria do Carmo");
                cliente2.setEmail("maria.do.carmo@algaworks.com.br");
                cliente2.setTelefone("(12) 99999-2222");

                return Arrays.asList(cliente1, cliente2);
        }
}
```

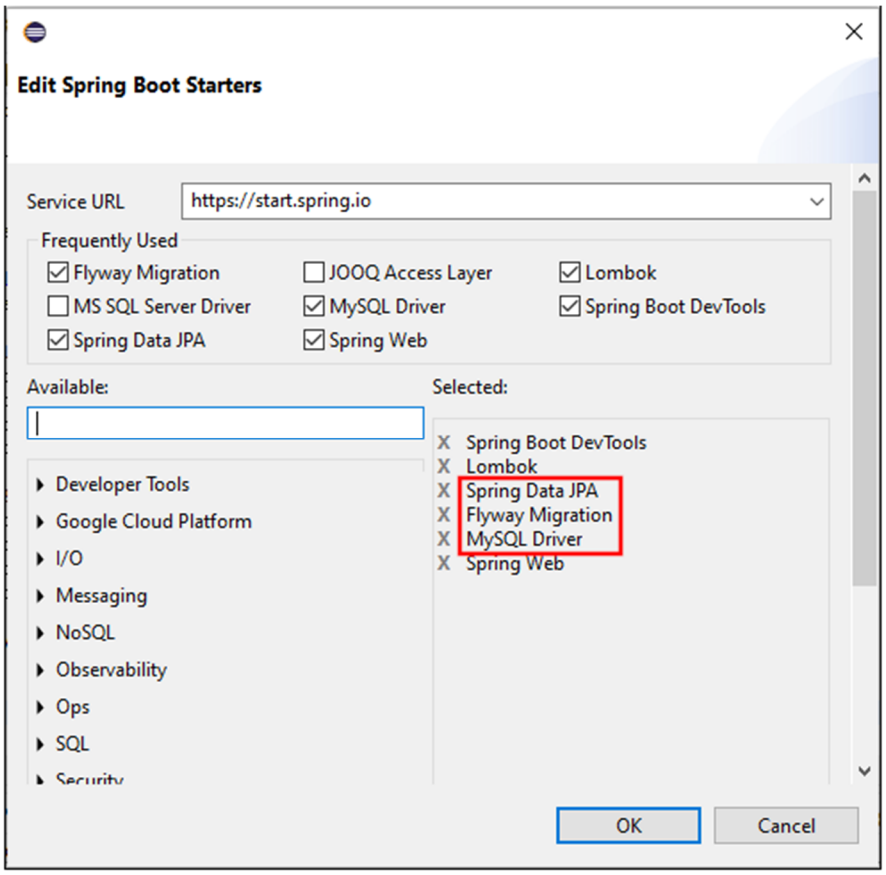
Testar o End-Point no Postman:



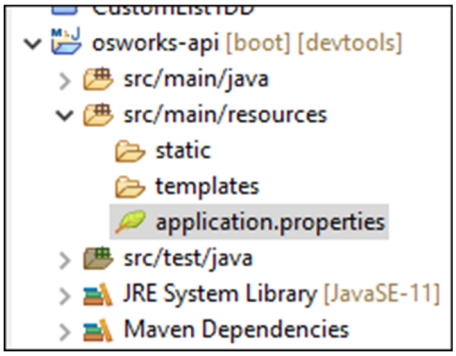
Persistência, Bean Validation e Exception Handler

Instalar o Spring-Jpa, o driver do MySQL e o Flyway Migration através do Edit Starters:



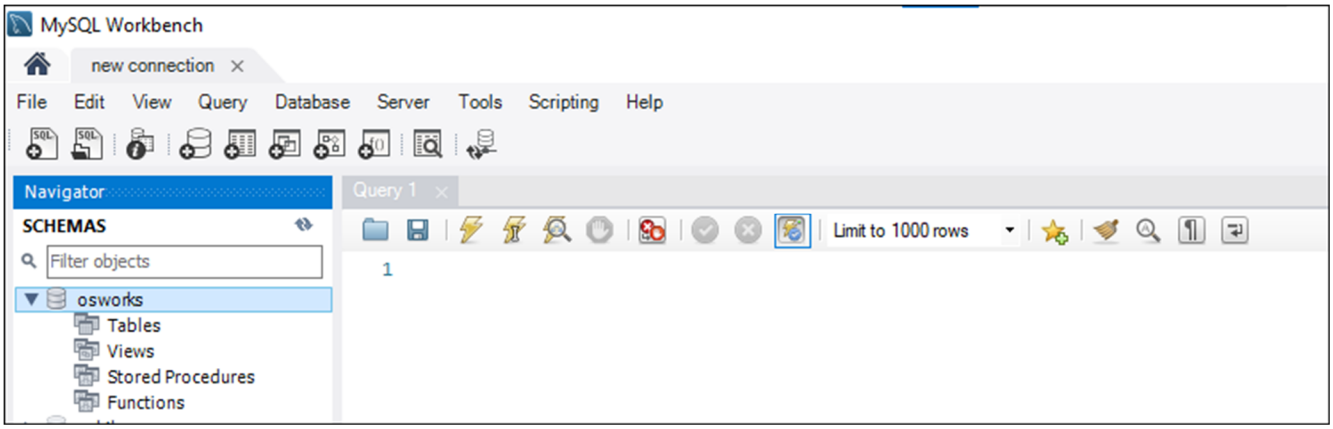


Configurar a conexão ao banco de dados no arquivo **application.properties**:



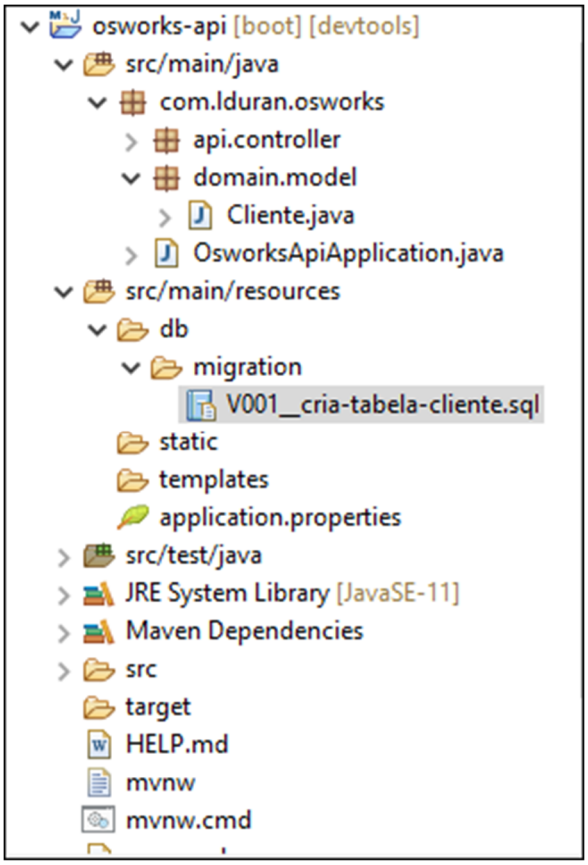

```
spring.datasource.url=jdbc:mysql://localhost/osworks?createDatabaseIfNotExist=true&serverTimezone=UTC
spring.datasource.username= root
spring.datasource.password= VCK4AGzFOnc@
```

Schema criado pelo Spring.Jpa no MySQL:



Configurando o Flyway Migration:

Criar uma pasta db/migration dentro de src/main/resources e criar o script sql inicial **V001__cria-tabela-cliente.sql**



Alterar o conteúdo do script sql inicial **V001__cria-tabela-cliente.sql** com o script de criação da tabela Cliente:

```
create table cliente
(
    id bigint not null auto_increment,
    nome varchar(60) not null,
    email varchar(255) not null,
    telefone varchar(20) not null,

    primary key (id)
);
```

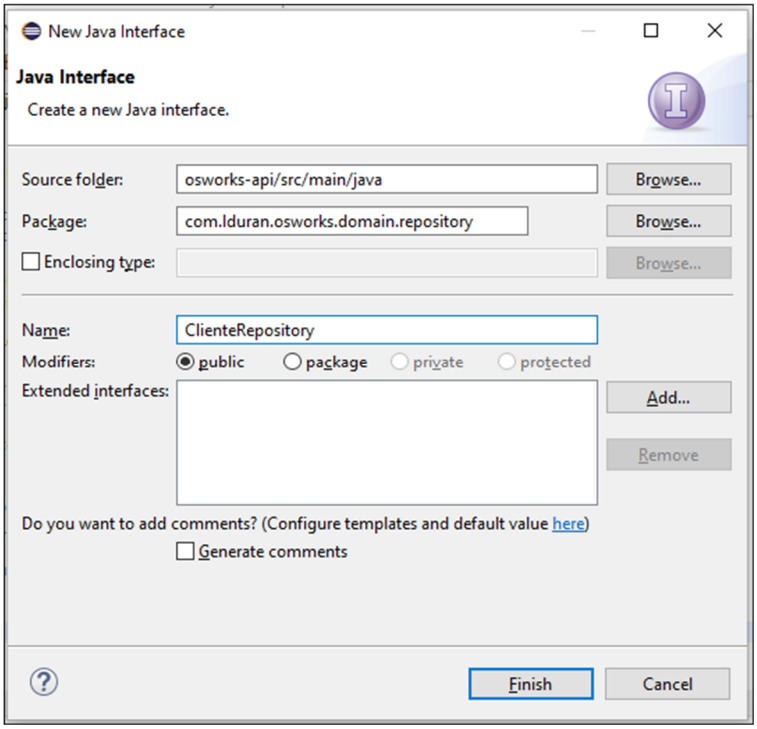
Fazer o Mapeamento Objeto Relacional da entidade Cliente:

```
package com.lduran.osworks.domain.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
public class Cliente
{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;
        private String nome;
        private String email;
        private String telefone;
}
```

Configurar o EntityManager no ClienteController para acessar os dados do banco:

```
package com.lduran.osworks.api.controller;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;

@RestController
public class ClienteController
{
	@PersistenceContext
	private EntityManager manager;

	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		return this.manager.createQuery("from Cliente", Cliente.class).getResultList();
	}
}
```

Criar a interface Repositório para o Cliente:




```
package com.lduran.osworks.domain.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.lduran.osworks.domain.model.Cliente;

@Repository
public interface ClienteRepository extends JpaRepository<Cliente, Long>
{

}
```

Utilizar a interface ClienteRepository no ClienteController:

```
package com.lduran.osworks.api.controller;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;

@RestController
public class ClienteController
{
	@Autowired
	private ClienteRepository clienteRepository;

	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		return this.clienteRepository.findAll();
	}
}
```

Criar os métodos findByNome(String nome) e findByNomeContaining(String nome) na interface Repositório do Cliente:

```
package com.lduran.osworks.domain.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.lduran.osworks.domain.model.Cliente;

@Repository
public interface ClienteRepository extends JpaRepository<Cliente, Long>
{
	List<Cliente> findByNome(String nome);
	List<Cliente> findByNomeContaining(String nome);

}
```

Criação de BuscaPorNome, BuscaPorNomeParcial e Busca usando o clienteId:

```
package com.lduran.osworks.api.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;

@RestController
public class ClienteController
{
	@Autowired
	private ClienteRepository clienteRepository;

	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		return this.clienteRepository.findAll();
	}
```

```
	@GetMapping("/clientes/buscando-nome")
	public List<Cliente> buscarPorNome()
	{
		return this.clienteRepository.findByNome("Maria de Gouvea");
	}

	@GetMapping("/clientes/buscando-nome-parcial")
	public List<Cliente> buscarPorNomeParcial()
	{
		return this.clienteRepository.findByNomeContaining("João");
	}

	@GetMapping("/clientes/{clienteId}")
	public Cliente buscar(@PathVariable Long clienteId)
	{
		return this.clienteRepository.findById(clienteId).orElse(null);
	}
}
```

Melhoria na resposta do método buscar(Long clienteId):

```
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;

@RestController
public class ClienteController
{
	@Autowired
	private ClienteRepository clienteRepository;

	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		return this.clienteRepository.findAll();
	}

	@GetMapping("/clientes/{clienteId}")
	public ResponseEntity<Cliente> buscar(@PathVariable Long clienteId)
	{
		Optional<Cliente> cliente = this.clienteRepository.findById(clienteId);
		if(cliente.isPresent())
		{
			return ResponseEntity.ok(cliente.get());
		}

		return ResponseEntity.notFound().build();
	}
}
```

Configurar o **Jakarta Bean Validation** no pom.xml:

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

Implementar o Bean Validation na entidade Cliente:

```
package com.lduran.osworks.domain.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

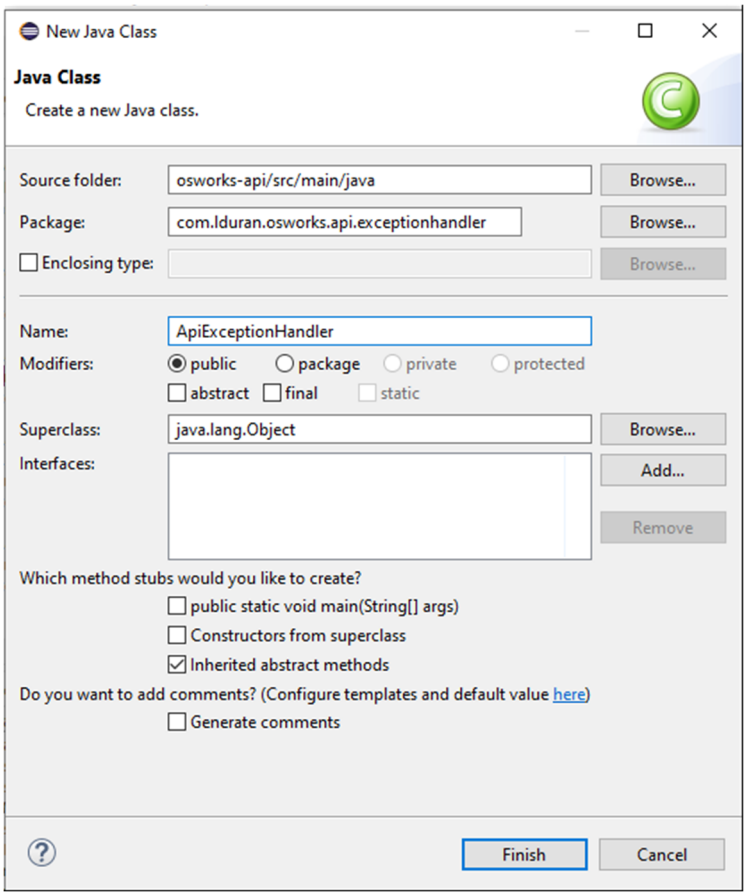
```java
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class Cliente
{
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@EqualsAndHashCode.Include
	private Long id;

	@NotBlank
	@Size(max=60)
	private String nome;

	@NotBlank
	@Email
	@Size(max=255)
	private String email;

	@NotBlank
	@Size(max=20)
	private String telefone;
}
```

Implementar o Bean Validation no ClienteController:

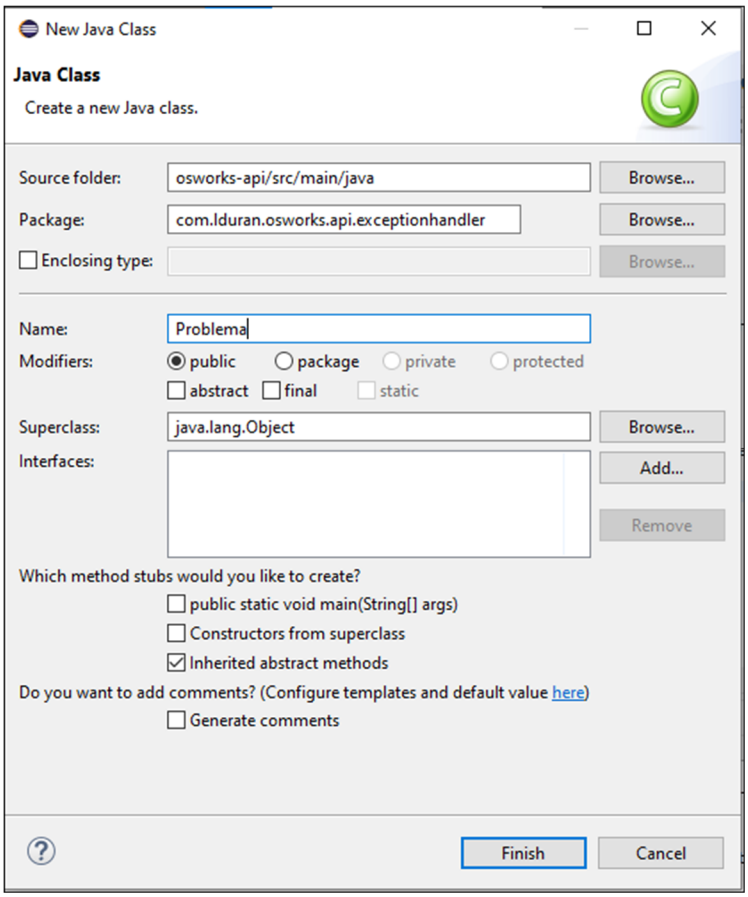
```java
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;

@RestController
public class ClienteController
{
	@Autowired
	private ClienteRepository clienteRepository;

	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		return this.clienteRepository.findAll();
	}

	@GetMapping("/clientes/{clienteId}")
	public ResponseEntity<Cliente> buscar(@PathVariable Long clienteId)
	{
		Optional<Cliente> cliente = this.clienteRepository.findById(clienteId);
		if(cliente.isPresent())
		{
			return ResponseEntity.ok(cliente.get());
		}
```

```
                return ResponseEntity.notFound().build();
        }

        @PostMapping("/clientes")
        @ResponseStatus(HttpStatus.CREATED)
        public Cliente adicionar(@Valid @RequestBody Cliente cliente)
        {
                return this.clienteRepository.save(cliente);
        }

        @PutMapping("/clientes/{clienteId}")
        public ResponseEntity<Cliente> atualizar(@Valid @PathVariable Long clienteId, @RequestBody Cliente cliente)
        {
                if(!this.clienteRepository.existsById(clienteId))
                {
                        return ResponseEntity.notFound().build();
                }

                cliente.setId(clienteId);
                cliente = this.clienteRepository.save(cliente);
                return ResponseEntity.ok(cliente);
        }

        @DeleteMapping("/clientes/{clienteId}")
        public ResponseEntity<Void> remover(@PathVariable Long clienteId)
        {
                if(!this.clienteRepository.existsById(clienteId))
                {
                        return ResponseEntity.notFound().build();
                }

                this.clienteRepository.deleteById(clienteId);

                return ResponseEntity.noContent().build();
        }
}
```

Criar a classe ApiExceptionHandler() para tratar as exceções:



Criar classe Problema() para representar o problema:




```java
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class Problema
{
	private int status;
	private LocalDateTime dataHora;
	private String titulo;
}
```

Instanciar um Problema() na classe ApiExceptionHandler():

```java
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice
public class ApiExceptionHandler extends ResponseEntityExceptionHandler
{
	@Override
	protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
			HttpHeaders headers, HttpStatus status, WebRequest request)
	{
		var problema = new Problema();
```

```
                problema.setStatus(status.value());

                problema.setTitulo("Um ou mais campos estão inválidos. " +
                                "Faça o preenchimento correto e tente novamente.");

                problema.setDataHora(LocalDateTime.now());

                return super.handleExceptionInternal(ex, problema, headers, status, request);
        }
}
```

Criar uma classe Campo():

```
package com.lduran.osworks.api.exceptionhandler;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter @Setter
@NoArgsConstructor @AllArgsConstructor
public class Campo
{
        public String nome;
        public String mensagem;
}
```

Criar uma lista de Campo() em Problema():

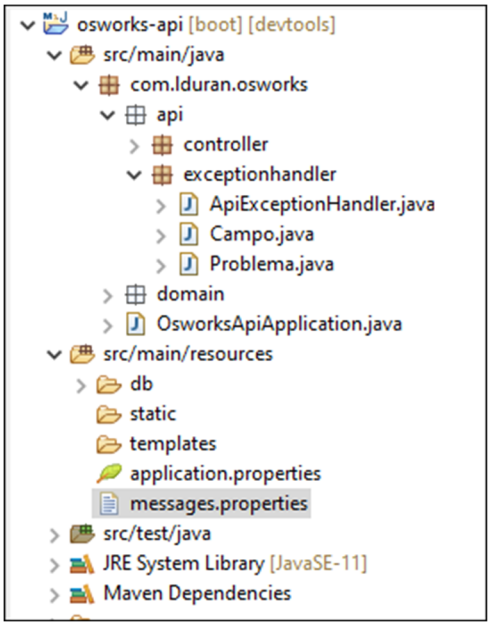
```
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;
import java.util.List;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class Problema
{
        private int status;
        private LocalDateTime dataHora;
        private String titulo;
        private List<Campo> campos;
}
```

Obter os campos com erro na classe ApiExceptionHandler():

```
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;
import java.util.ArrayList;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice
public class ApiExceptionHandler extends ResponseEntityExceptionHandler
{
        @Override
        protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
                        HttpHeaders headers, HttpStatus status, WebRequest request)
        {
                var campos = new ArrayList<Campo>();

                for(ObjectError error : ex.getBindingResult().getAllErrors())
```

```
                {
                        String nome = ((FieldError)error).getField();
                        String mensagem = error.getDefaultMessage();

                        campos.add(new Campo(nome,mensagem));
                }

                var problema = new Problema();

                problema.setStatus(status.value());

                problema.setTitulo("Um ou mais campos estão inválidos. " +
                                "Faça o preenchimento correto e tente novamente.");

                problema.setDataHora(LocalDateTime.now());

                problema.setCampos(campos);

                return super.handleExceptionInternal(ex, problema, headers, status, request);
        }
}
```

Criar um arquivo messages.properties dentro de src/main/resources:



Personalizar as mensagens de erro no arquivo messages.properties:

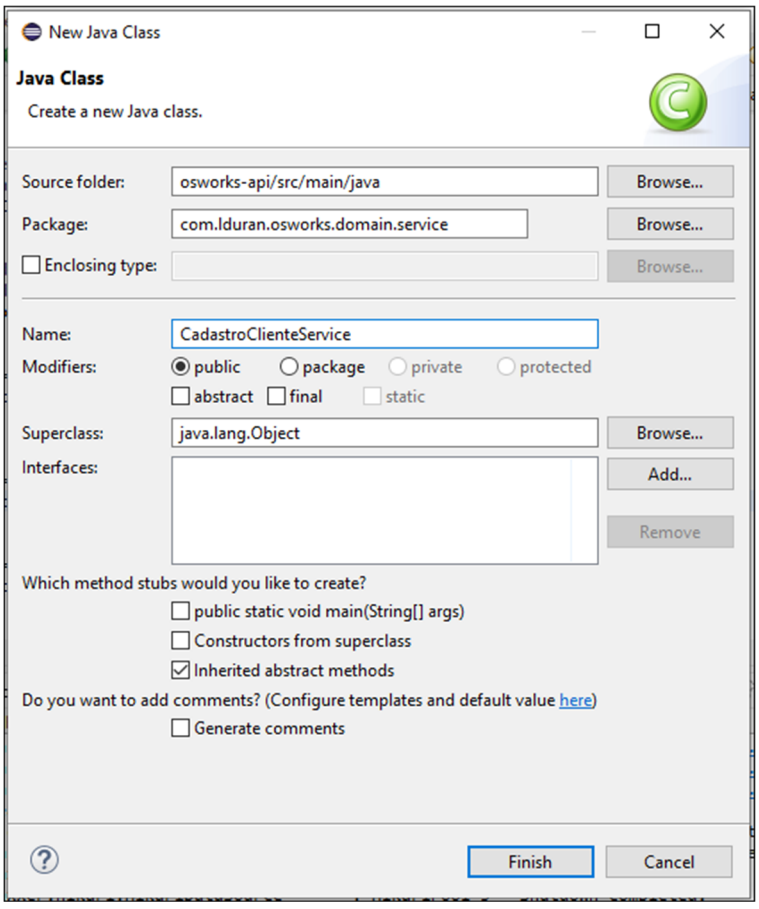
```
NotBlank=é obrigatório
Size=deve ter no mínimo {2} e no máximo {1}
Email=deve ser um e-mail válido
```

Instanciar a classe MessageSource() em ApiExceptionHandler() para utilizar as mensagens personalizadas:

```
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;
import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
```

```java
@ControllerAdvice
public class ApiExceptionHandler extends ResponseEntityExceptionHandler
{
	@Autowired
	private MessageSource messageSource;

	@Override
	protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
			HttpHeaders headers, HttpStatus status, WebRequest request)
	{
		var campos = new ArrayList<Campo>();

		for(ObjectError error : ex.getBindingResult().getAllErrors())
		{
			String nome = ((FieldError)error).getField();
			String mensagem = this.messageSource.getMessage(error,LocaleContextHolder.getLocale());

			campos.add(new Campo(nome,mensagem));
		}

		var problema = new Problema();

		problema.setStatus(status.value());

		problema.setTitulo("Um ou mais campos estão inválidos. " +
				"Faça o preenchimento correto e tente novamente.");

		problema.setDataHora(LocalDateTime.now());

		problema.setCampos(campos);

		return super.handleExceptionInternal(ex, problema, headers, status, request);
	}
}
```

Criação da camada de serviço do Cliente():



```java
package com.lduran.osworks.domain.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;

@Service
public class CadastroClienteService
{
	@Autowired
	private ClienteRepository clienteRepository;

	public Cliente salvar(Cliente cliente)
	{
		return this.clienteRepository.save(cliente);
	}

	public void excluir(Long clienteId)
	{
		this.clienteRepository.deleteById(clienteId);
	}
}
```

Refatorar a classe ClienteController(), utilizando os métodos salvar() e excluir() de CadastroClienteService():

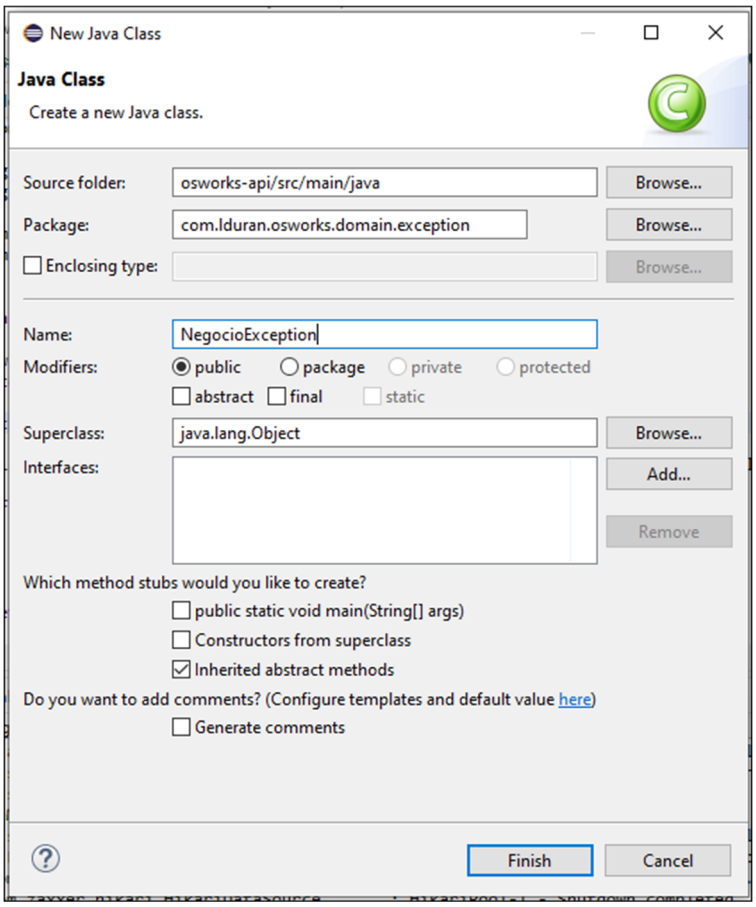
```java
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;
import com.lduran.osworks.domain.service.CadastroClienteService;

@RestController
public class ClienteController
{
	@Autowired
	private ClienteRepository clienteRepository;

	@Autowired
	private CadastroClienteService cadastroCliente;

	@GetMapping("/clientes")
	public List<Cliente> listar()
	{
		return this.clienteRepository.findAll();
	}

	@GetMapping("/clientes/{clienteId}")
	public ResponseEntity<Cliente> buscar(@PathVariable Long clienteId)
	{
		Optional<Cliente> cliente = this.clienteRepository.findById(clienteId);
		if(cliente.isPresent())
		{
			return ResponseEntity.ok(cliente.get());
		}

		return ResponseEntity.notFound().build();
	}
```

```
	@PostMapping("/clientes")
	@ResponseStatus(HttpStatus.CREATED)
	public Cliente adicionar(@Valid @RequestBody Cliente cliente)
	{
		return this.cadastroCliente.salvar(cliente);
	}

	@PutMapping("/clientes/{clienteId}")
	public ResponseEntity<Cliente> atualizar(@Valid @PathVariable Long clienteId, @RequestBody Cliente cliente)
	{
		if(!this.clienteRepository.existsById(clienteId))
		{
			return ResponseEntity.notFound().build();
		}

		cliente.setId(clienteId);
		cliente = this.cadastroCliente.salvar(cliente);
		return ResponseEntity.ok(cliente);
	}

	@DeleteMapping("/clientes/{clienteId}")
	public ResponseEntity<Void> remover(@PathVariable Long clienteId)
	{
		if(!this.clienteRepository.existsById(clienteId))
		{
			return ResponseEntity.notFound().build();
		}

		this.cadastroCliente.excluir(clienteId);

		return ResponseEntity.noContent().build();
	}
}
```

Criar busca de cliente utilizando o campo email:

```
package com.lduran.osworks.domain.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.lduran.osworks.domain.model.Cliente;

@Repository
public interface ClienteRepository extends JpaRepository<Cliente, Long>
{
	List<Cliente> findByNome(String nome);
	List<Cliente> findByNomeContaining(String nome);
	Cliente findByEmail(String email);
}
```

Criar uma classe NegocioException():




```java
package com.lduran.osworks.domain.exception;

public class NegocioException extends RuntimeException
{
	private static final long serialVersionUID = 1L;

	public NegocioException(String message)
	{
		super(message);
	}
}
```

Criar verificação de cliente já cadastrado:

```java
package com.lduran.osworks.domain.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.exception.NegocioException;
import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.repository.ClienteRepository;

@Service
public class CadastroClienteService
{
	@Autowired
	private ClienteRepository clienteRepository;

	public Cliente salvar(Cliente cliente)
	{
		Cliente clienteExistente = this.clienteRepository.findByEmail(cliente.getEmail());

		if(clienteExistente != null && !clienteExistente.equals(cliente))
		{
			throw new NegocioException("Já existe um cliente cadastrado com este e-mail");
		}
```

```
                return this.clienteRepository.save(cliente);
        }

        public void excluir(Long clienteId)
        {
                this.clienteRepository.deleteById(clienteId);
        }
}
```

Tratar a mensagem de erro de “Cliente já cadastrado”:

```
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;
import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import com.lduran.osworks.domain.exception.NegocioException;

@ControllerAdvice
public class ApiExceptionHandler extends ResponseEntityExceptionHandler
{
        @Autowired
        private MessageSource messageSource;

        @ExceptionHandler(NegocioException.class)
        public ResponseEntity<Object> handleNegocio(NegocioException ex, WebRequest request)
        {
                var status = HttpStatus.BAD_REQUEST;

                var problema = new Problema();
                problema.setStatus(status.value());
                problema.setTitulo(ex.getMessage());
                problema.setDataHora(LocalDateTime.now());

                return super.handleExceptionInternal(ex, problema, new HttpHeaders(), status, request);
        }

        @Override
        protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
                        HttpHeaders headers, HttpStatus status, WebRequest request)
        {
                var campos = new ArrayList<Campo>();

                for(ObjectError error : ex.getBindingResult().getAllErrors())
                {
                        String nome = ((FieldError)error).getField();
                        String mensagem = this.messageSource.getMessage(error,LocaleContextHolder.getLocale());

                        campos.add(new Campo(nome,mensagem));
                }

                var problema = new Problema();

                problema.setStatus(status.value());

                problema.setTitulo("Um ou mais campos estão inválidos. " +
                                "Faça o preenchimento correto e tente novamente.");

                problema.setDataHora(LocalDateTime.now());

                problema.setCampos(campos);
```

```
                return super.handleExceptionInternal(ex, problema, headers, status, request);
        }
}
```

Alteração na classe Problema() para apenas retornar campos não nulos:

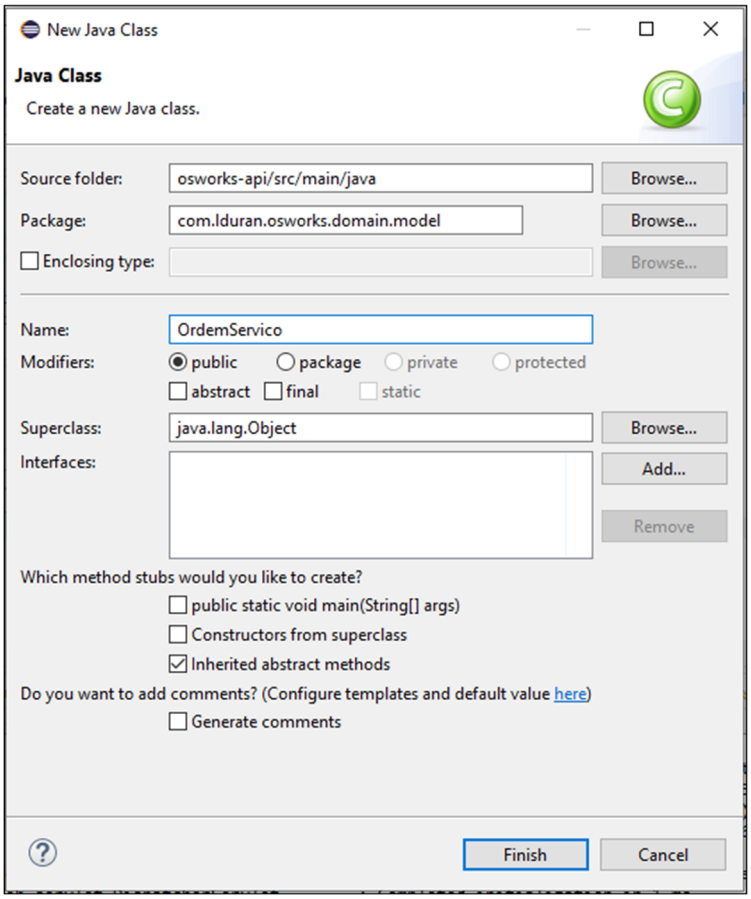
```
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.annotation.JsonInclude.Include;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
@JsonInclude(Include.NON_NULL)
public class Problema
{
        private int status;
        private LocalDateTime dataHora;
        private String titulo;
        private List<Campo> campos;
}
```

Criar um enum StatusOrdemServico:

```
package com.lduran.osworks.domain.model;

public enum StatusOrdemServico
{
        ABERTA, FINALIZADA, CANCELADA
}
```

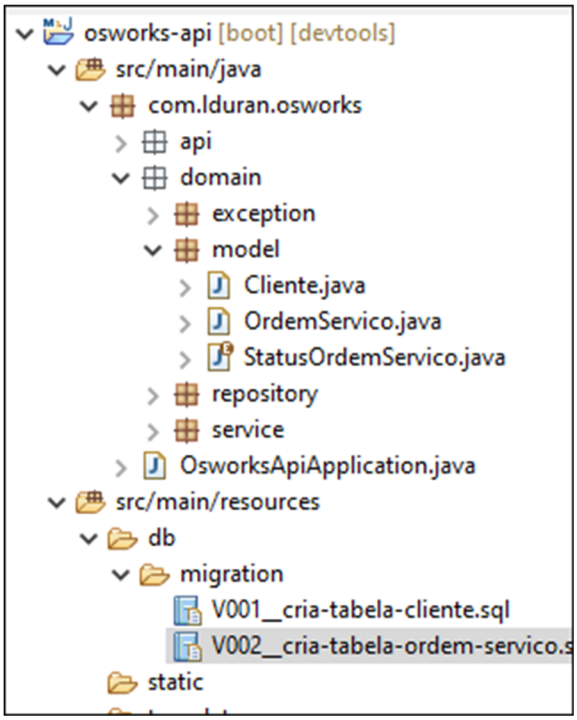
Técnicas e boas práticas

Criar a entidade OrdemServico:



```java
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.LocalDateTime;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@EqualsAndHashCode.Include
	private Long id;

	@ManyToOne
	private Cliente cliente;
	private String descricao;
	private BigDecimal preco;

	@Enumerated(EnumType.STRING)
	private StatusOrdemServico status;
	private LocalDateTime dataAbertura;
	private LocalDateTime dataFinalizacao;
}
```

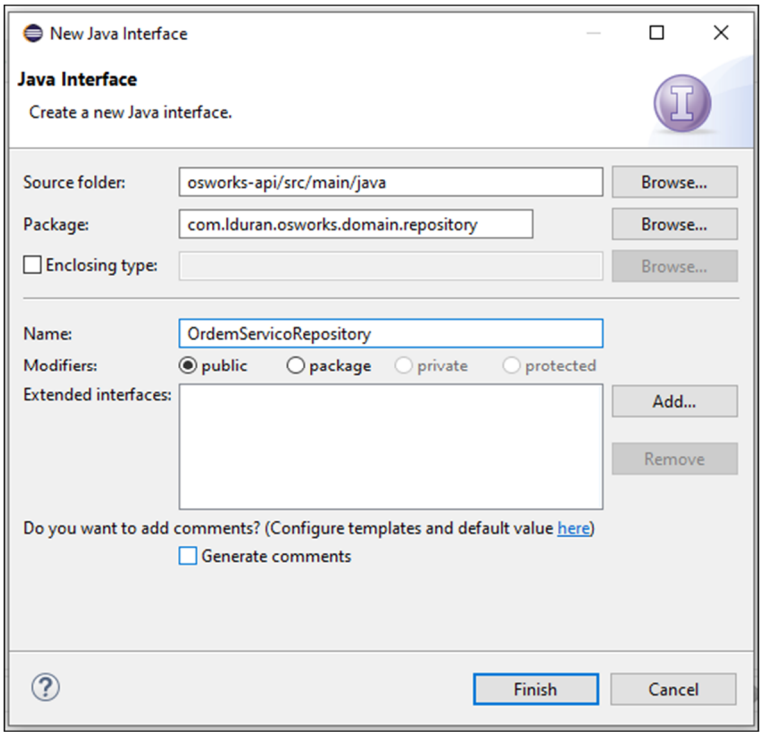
Criar o script sql inicial **V002__cria-tabela-ordem-servico.sql**:



Alterar o conteúdo do script sql inicial **V002__cria-tabela-ordem-servico.sql** com o script de criação da tabela Cliente:

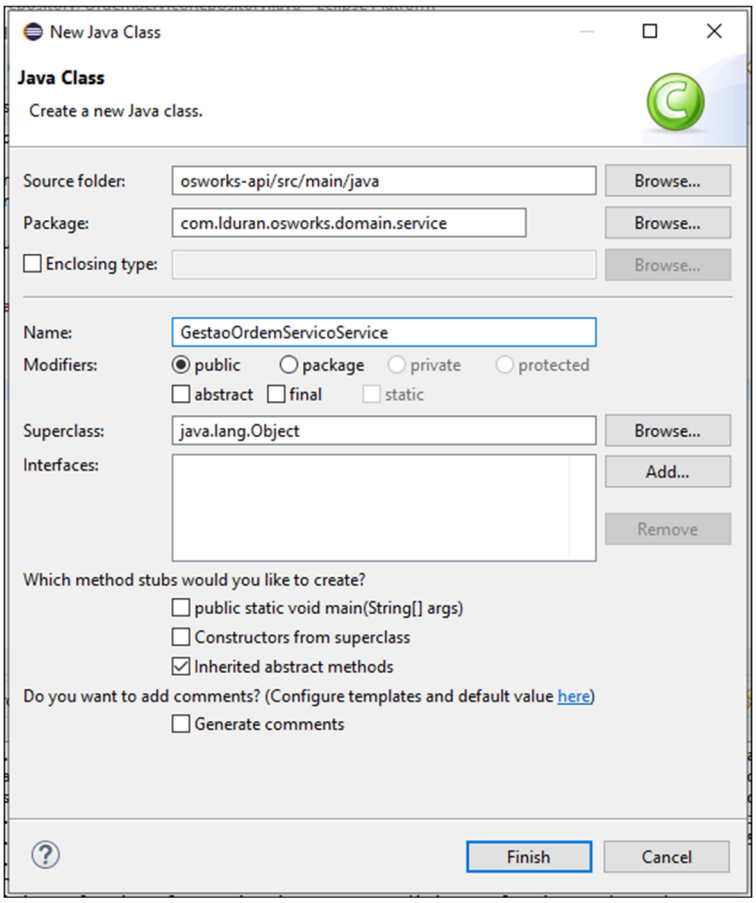
```
drop table if exists ordem_servico;

create table ordem_servico
(
    id bigint not null auto_increment,
    cliente_id bigint not null,
    descricao text not null,
    preco decimal(10,2) not null,
    status varchar(20) not null,
    data_abertura datetime not null,
    data_finalizacao datetime,

    primary key (id)
);

alter table ordem_servico add constraint fk_ordem_servico_cliente
foreign key (cliente_id) references cliente (id);
```

Criar a interface Repositório para a OrdermServico:



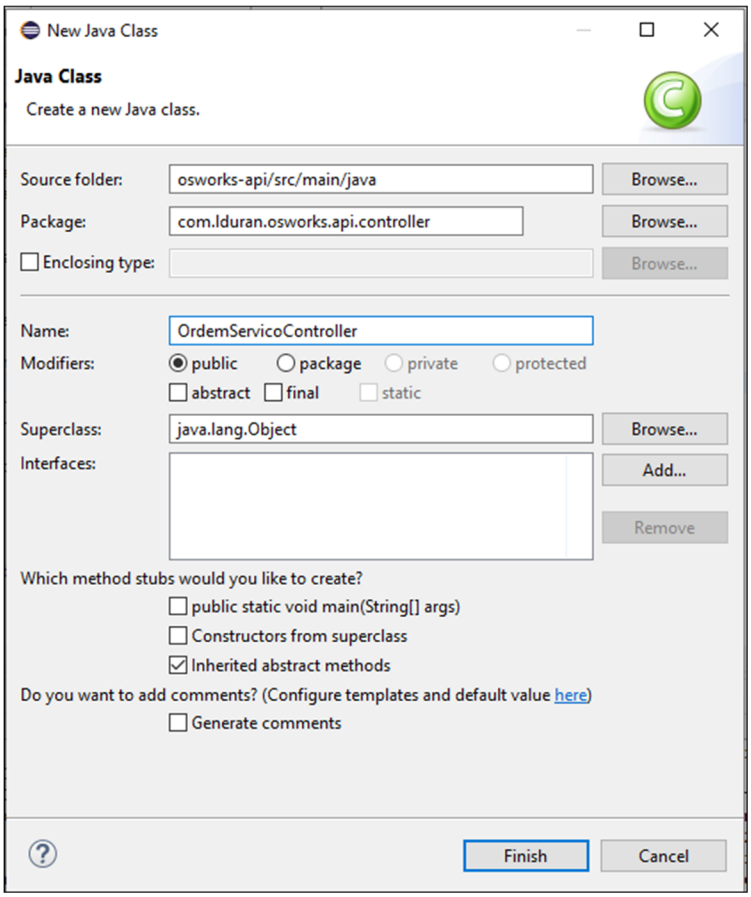

```
package com.lduran.osworks.domain.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.lduran.osworks.domain.model.OrdemServico;

@Repository
public interface OrdemServicoRepository extends JpaRepository<OrdemServico, Long>
{

}
```

Criar a classe de Serviço de Gestão para a OrdemServico:




```java
package com.lduran.osworks.domain.service;

import java.time.LocalDateTime;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.model.StatusOrdemServico;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;

@Service
public class GestaoOrdemServicoService
{
	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	public OrdemServico criar(OrdemServico ordemServico)
	{
		ordemServico.setStatus(StatusOrdemServico.ABERTA);
		ordemServico.setDataAbertura(LocalDateTime.now());

		return this.ordemServicoRepository.save(ordemServico);
	}

	public void excluir(Long ordemServicoId)
	{
		this.ordemServicoRepository.deleteById(ordemServicoId);
	}
}
```

Criar a classe OrdemServicoController():




```java
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;
import com.lduran.osworks.domain.service.GestaoOrdemServicoService;

@RestController
@RequestMapping("/ordens-servico")
public class OrdemServicoController
{
	@Autowired
	private GestaoOrdemServicoService gestaoOrdemServicoService;

	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	@GetMapping()
	public List<OrdemServico> listar()
	{
		return this.ordemServicoRepository.findAll();
	}
```

```
	@GetMapping("/{ordemServicoId}")
	public ResponseEntity<OrdemServico> buscar(@PathVariable Long ordemServicoId)
	{
		Optional<OrdemServico> ordemServico = this.ordemServicoRepository.findById(ordemServicoId);
		if(ordemServico.isPresent())
		{
			return ResponseEntity.ok(ordemServico.get());
		}

		return ResponseEntity.notFound().build();
	}

	@PostMapping
	@ResponseStatus(HttpStatus.CREATED)
	public OrdemServico criar(@Valid @RequestBody OrdemServico ordemServico)
	{
		return this.gestaoOrdemServicoService.criar(ordemServico);
	}
}
```

Ajuste na classe GestaoOrdemServicoService() para que traga o cliente da ordem de serviço:

```
package com.lduran.osworks.domain.service;

import java.time.LocalDateTime;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.exception.NegocioException;
import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.model.StatusOrdemServico;
import com.lduran.osworks.domain.repository.ClienteRepository;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;

@Service
public class GestaoOrdemServicoService
{
	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	@Autowired
	private ClienteRepository clienteRepository;

	public OrdemServico criar(OrdemServico ordemServico)
	{
		Cliente cliente = this.clienteRepository.findById(ordemServico.getCliente().getId())
				.orElseThrow(() -> new NegocioException("Cliente não encontrado."));

		ordemServico.setCliente(cliente);
		ordemServico.setStatus(StatusOrdemServico.ABERTA);
		ordemServico.setDataAbertura(LocalDateTime.now());

		return this.ordemServicoRepository.save(ordemServico);
	}

	public void excluir(Long ordemServicoId)
	{
		this.ordemServicoRepository.deleteById(ordemServicoId);
	}
}
```

Definir como Read Only os atributos da classe OrdemServico que só devem ser alteradas pelo sistema:

```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.LocalDateTime;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonProperty.Access;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@EqualsAndHashCode.Include
	private Long id;

	@ManyToOne
	private Cliente cliente;
	private String descricao;
	private BigDecimal preco;

	@Enumerated(EnumType.STRING)
	@JsonProperty(access = Access.READ_ONLY)
	private StatusOrdemServico status;

	@JsonProperty(access = Access.READ_ONLY)
	private LocalDateTime dataAbertura;

	@JsonProperty(access = Access.READ_ONLY)
	private LocalDateTime dataFinalizacao;
}
```

Definir as validações da Ordem de Serviço:

```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.LocalDateTime;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonProperty.Access;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@EqualsAndHashCode.Include
	private Long id;

	@NotNull
	@ManyToOne
	private Cliente cliente;

	@NotBlank
	private String descricao;
```

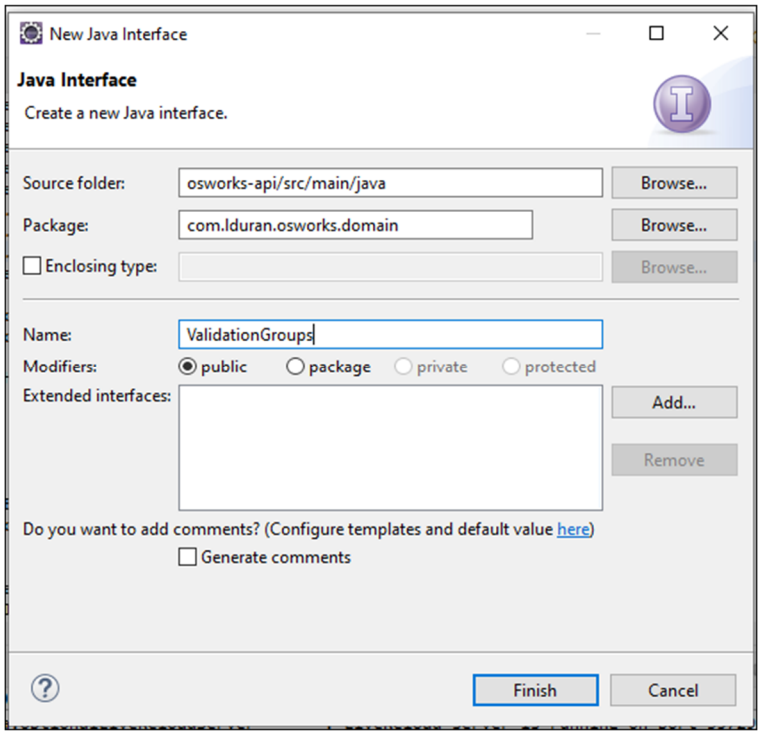
```
	@NotNull
	private BigDecimal preco;

	@Enumerated(EnumType.STRING)
	@JsonProperty(access = Access.READ_ONLY)
	private StatusOrdemServico status;

	@JsonProperty(access = Access.READ_ONLY)
	private LocalDateTime dataAbertura;

	@JsonProperty(access = Access.READ_ONLY)
	private LocalDateTime dataFinalizacao;
}
```

Definir as validações no Controller da Ordem de Serviço:

```
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;
import com.lduran.osworks.domain.service.GestaoOrdemServicoService;

@RestController
@RequestMapping("/ordens-servico")
public class OrdemServicoController
{
	@Autowired
	private GestaoOrdemServicoService gestaoOrdemServicoService;

	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	@GetMapping()
	public List<OrdemServico> listar()
	{
		return this.ordemServicoRepository.findAll();
	}

	@GetMapping("/{ordemServicoId}")
	public ResponseEntity<OrdemServico> buscar(@PathVariable Long ordemServicoId)
	{
		Optional<OrdemServico> ordemServico = this.ordemServicoRepository.findById(ordemServicoId);
		if(ordemServico.isPresent())
		{
			return ResponseEntity.ok(ordemServico.get());
		}

		return ResponseEntity.notFound().build();
	}

	@PostMapping
	@ResponseStatus(HttpStatus.CREATED)
	public OrdemServico criar(@Valid @RequestBody OrdemServico ordemServico)
	{
		return this.gestaoOrdemServicoService.criar(ordemServico);
	}

	@DeleteMapping("/{ordemServicoId}")
```

```
	public ResponseEntity<Void> remover(@PathVariable Long ordemServicoId)
	{
		if(!this.ordemServicoRepository.existsById(ordemServicoId))
		{
			return ResponseEntity.notFound().build();
		}

		this.gestaoOrdemServicoService.excluir(ordemServicoId);

		return ResponseEntity.noContent().build();
	}
}
```

Criar a interface ValidationGroups():




```
package com.lduran.osworks.domain;

public interface ValidationGroups
{
	public interface ClienteId {}
}
```

Cascatear a validação das propriedades de Cliente() à partir de OrdemServico():

```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.LocalDateTime;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.groups.ConvertGroup;
import javax.validation.groups.Default;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonProperty.Access;
import com.lduran.osworks.domain.ValidationGroups;
```

```java
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@EqualsAndHashCode.Include
	private Long id;

	@Valid
	@ConvertGroup(from=Default.class, to=ValidationGroups.ClienteId.class)
	@NotNull
	@ManyToOne
	private Cliente cliente;

	@NotBlank
	private String descricao;

	@NotNull
	private BigDecimal preco;

	@Enumerated(EnumType.STRING)
	@JsonProperty(access = Access.READ_ONLY)
	private StatusOrdemServico status;

	@JsonProperty(access = Access.READ_ONLY)
	private LocalDateTime dataAbertura;

	@JsonProperty(access = Access.READ_ONLY)
	private LocalDateTime dataFinalizacao;
}
```

```java
package com.lduran.osworks.domain.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import com.lduran.osworks.domain.ValidationGroups;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class Cliente
{
	@NotNull(groups=ValidationGroups.ClienteId.class)
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@EqualsAndHashCode.Include
	private Long id;

	@NotBlank
	@Size(max=60)
	private String nome;

	@NotBlank
	@Email
	@Size(max=255)
	private String email;

	@NotBlank
	@Size(max=20)
```

```
        private String telefone;
}
```

Usar na classe OrdemServico() data e hora no padrão 8601 com Offset (diferença entre UTC e o meridiano local):

```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.OffsetDateTime;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.groups.ConvertGroup;
import javax.validation.groups.Default;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonProperty.Access;
import com.lduran.osworks.domain.ValidationGroups;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @EqualsAndHashCode.Include
        private Long id;

        @Valid
        @ConvertGroup(from=Default.class, to=ValidationGroups.ClienteId.class)
        @NotNull
        @ManyToOne
        private Cliente cliente;

        @NotBlank
        private String descricao;

        @NotNull
        private BigDecimal preco;

        @Enumerated(EnumType.STRING)
        @JsonProperty(access = Access.READ_ONLY)
        private StatusOrdemServico status;

        @JsonProperty(access = Access.READ_ONLY)
        private OffsetDateTime dataAbertura;

        @JsonProperty(access = Access.READ_ONLY)
        private OffsetDateTime dataFinalizacao;
}
```

Usar na classe GestaoOrdemServicoService () data e hora no padrão 8601 com Offset (diferença entre UTC e o meridiano local):

```
package com.lduran.osworks.domain.service;

import java.time.OffsetDateTime;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.exception.NegocioException;
```

```java
import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.model.StatusOrdemServico;
import com.lduran.osworks.domain.repository.ClienteRepository;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;

@Service
public class GestaoOrdemServicoService
{
	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	@Autowired
	private ClienteRepository clienteRepository;

	public OrdemServico criar(OrdemServico ordemServico)
	{
		Cliente cliente = this.clienteRepository.findById(ordemServico.getCliente().getId())
				.orElseThrow(() -> new NegocioException("Cliente não encontrado."));

		ordemServico.setCliente(cliente);
		ordemServico.setStatus(StatusOrdemServico.ABERTA);
		ordemServico.setDataAbertura(OffsetDateTime.now());

		return this.ordemServicoRepository.save(ordemServico);
	}

	public void excluir(Long ordemServicoId)
	{
		this.ordemServicoRepository.deleteById(ordemServicoId);
	}
}
```

Usar na classe Problema() data e hora no padrão 8601 com Offset (diferença entre UTC e o meridiano local):

```java
package com.lduran.osworks.api.exceptionhandler;

import java.time.LocalDateTime;
import java.time.OffsetDateTime;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.annotation.JsonInclude.Include;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
@JsonInclude(Include.NON_NULL)
public class Problema
{
	private int status;
	private OffsetDateTime dataHora;
	private String titulo;
	private List<Campo> campos;
}
```

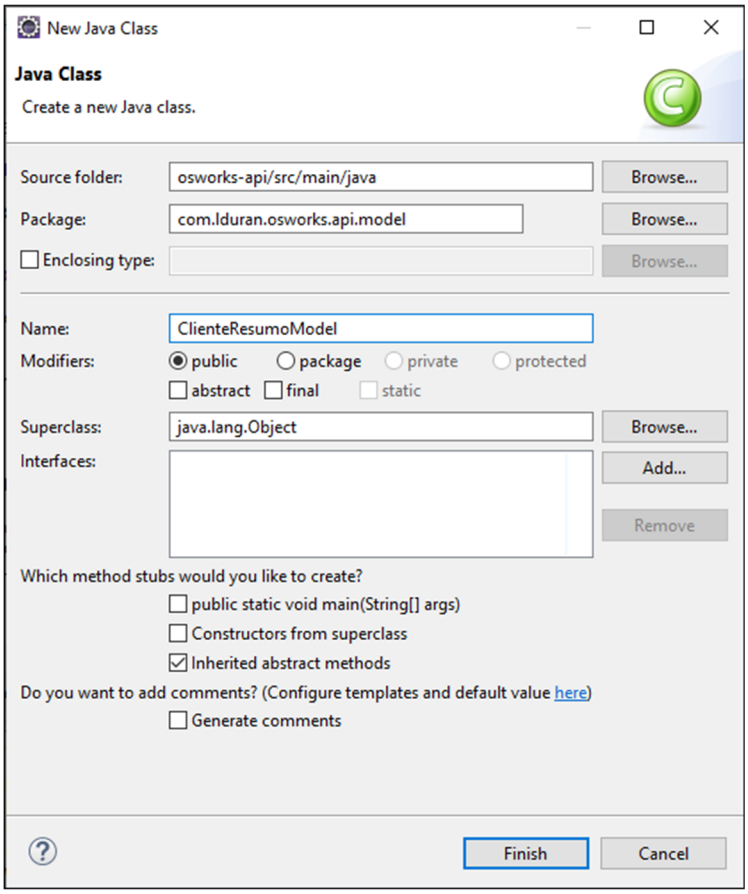
Usar na classe ApiExceptionHandler () data e hora no padrão 8601 com Offset (diferença entre UTC e o meridiano local):

```java
package com.lduran.osworks.api.exceptionhandler;

import java.time.OffsetDateTime;
import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
```

```java
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import com.lduran.osworks.domain.exception.NegocioException;

@ControllerAdvice
public class ApiExceptionHandler extends ResponseEntityExceptionHandler
{
	@Autowired
	private MessageSource messageSource;

	@ExceptionHandler(NegocioException.class)
	public ResponseEntity<Object> handleNegocio(NegocioException ex, WebRequest request)
	{
		var status = HttpStatus.BAD_REQUEST;

		var problema = new Problema();
		problema.setStatus(status.value());
		problema.setTitulo(ex.getMessage());
		problema.setDataHora(OffsetDateTime.now());

		return super.handleExceptionInternal(ex, problema, new HttpHeaders(), status, request);
	}

	@Override
	protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
			HttpHeaders headers, HttpStatus status, WebRequest request)
	{
		var campos = new ArrayList<Campo>();

		for(ObjectError error : ex.getBindingResult().getAllErrors())
		{
			String nome = ((FieldError)error).getField();
			String mensagem = this.messageSource.getMessage(error,LocaleContextHolder.getLocale());

			campos.add(new Campo(nome,mensagem));
		}

		var problema = new Problema();

		problema.setStatus(status.value());

		problema.setTitulo("Um ou mais campos estão inválidos. " +
				"Faça o preenchimento correto e tente novamente.");

		problema.setDataHora(OffsetDateTime.now());

		problema.setCampos(campos);

		return super.handleExceptionInternal(ex, problema, headers, status, request);
	}
}
```

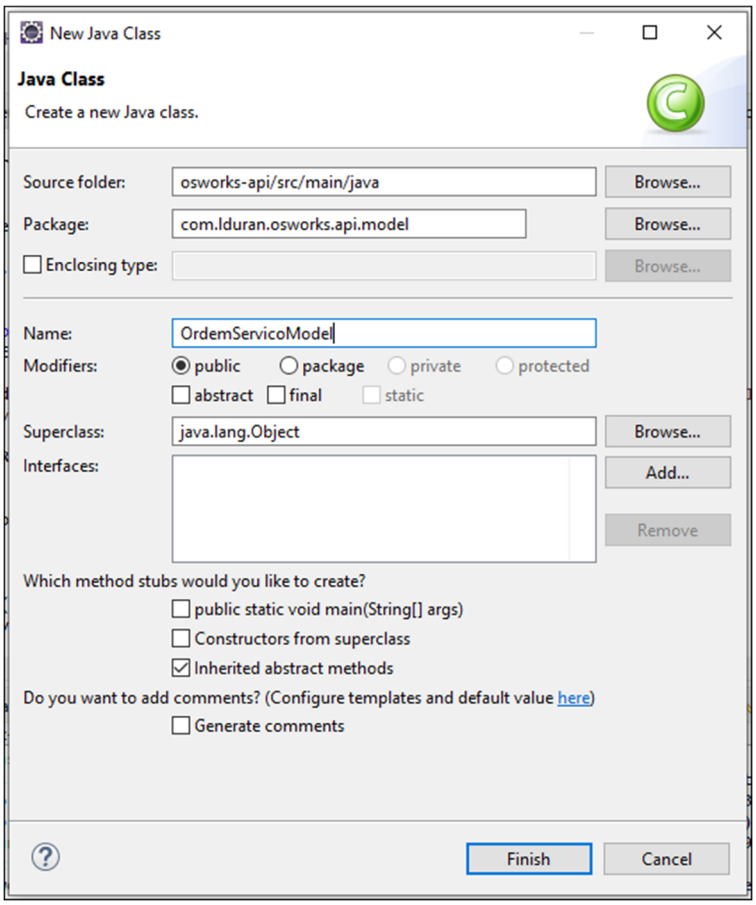
Realizar a separação do Domain Model do Representation Model criando classes de Transferência de Dados DTO (Data Transfer Object):

Criar a classe DTO para a entidade Cliente():



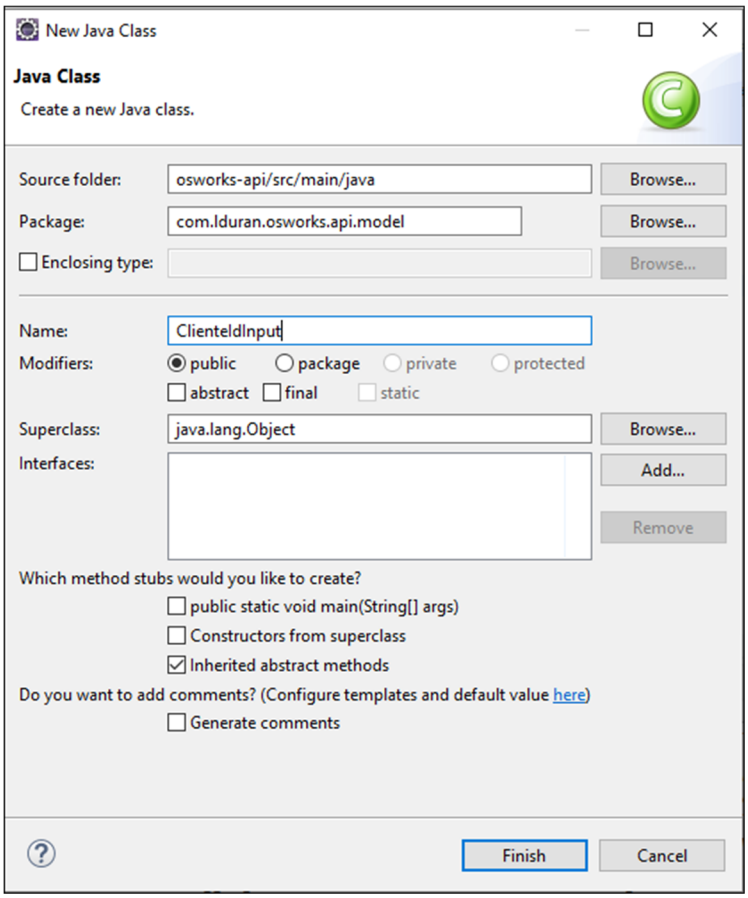

```java
package com.lduran.osworks.api.model;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class ClienteResumoModel
{
	private Long id;
	private String nome;
	private String telefone;
}
```

Criar a classe DTO para a entidade OrdemService():



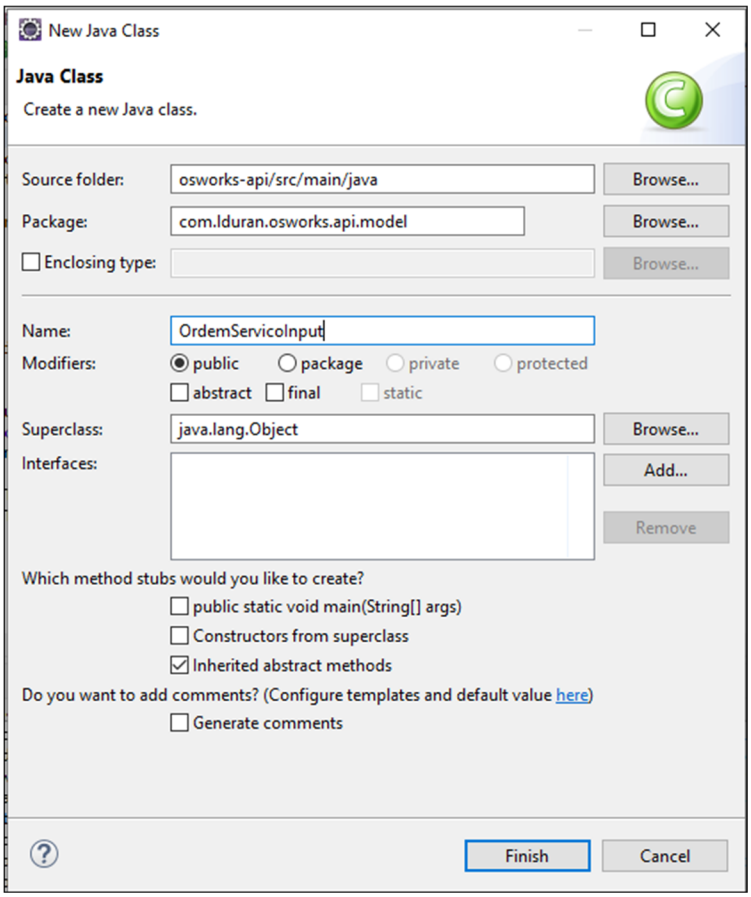

```java
package com.lduran.osworks.api.model;

import java.math.BigDecimal;
import java.time.OffsetDateTime;

import com.lduran.osworks.domain.model.StatusOrdemServico;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class OrdemServicoModel
{
	private Long id;
	private ClienteResumoModel cliente;
	private String descricao;
	private StatusOrdemServico status;
	private BigDecimal preco;
	private OffsetDateTime dataAbertura;
	private OffsetDateTime dataFinalizacao;
}
```

Criar a classe DTO para entrada de dados da entidade Cliente():



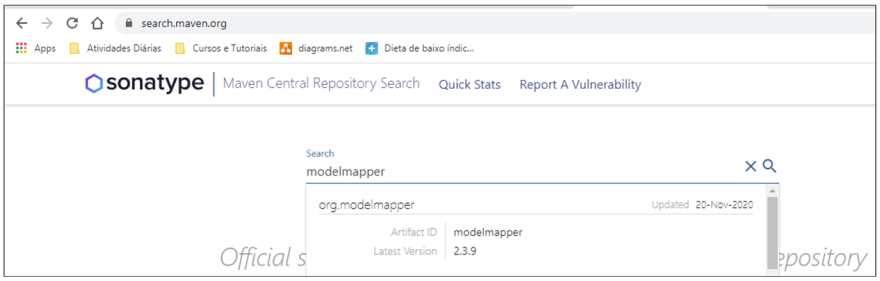

```java
package com.lduran.osworks.api.model;

import javax.validation.constraints.NotNull;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class ClienteIdInput
{
	@NotNull
	private Long id;
}
```

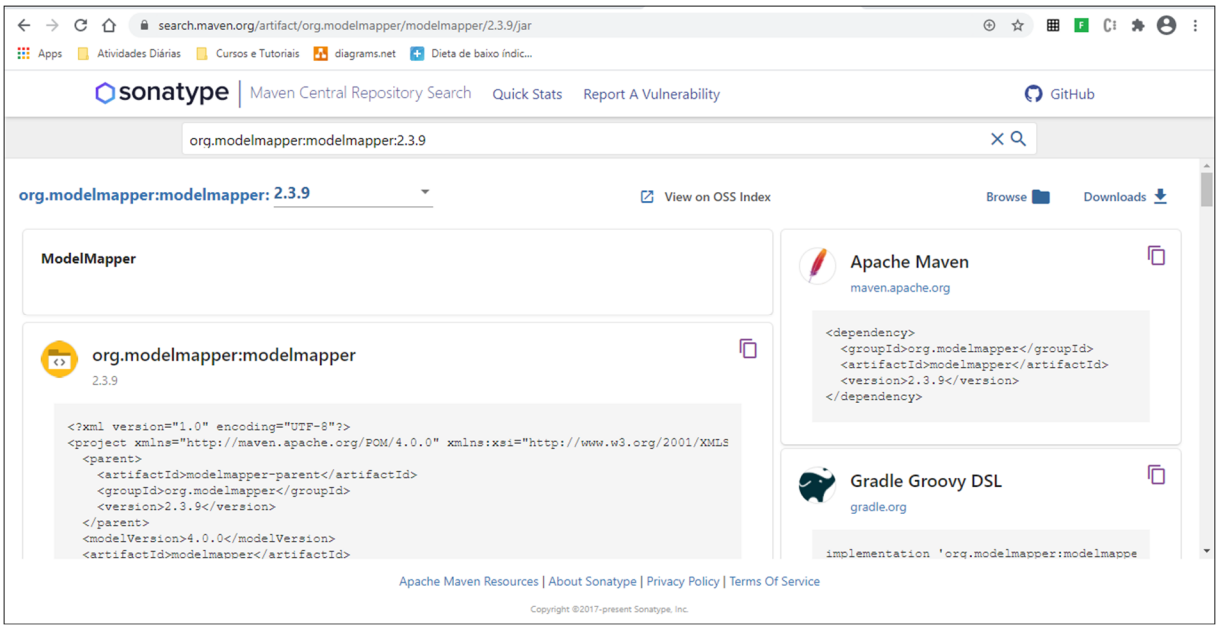
Criar a classe DTO para entrada de dados da entidade OrdemService():



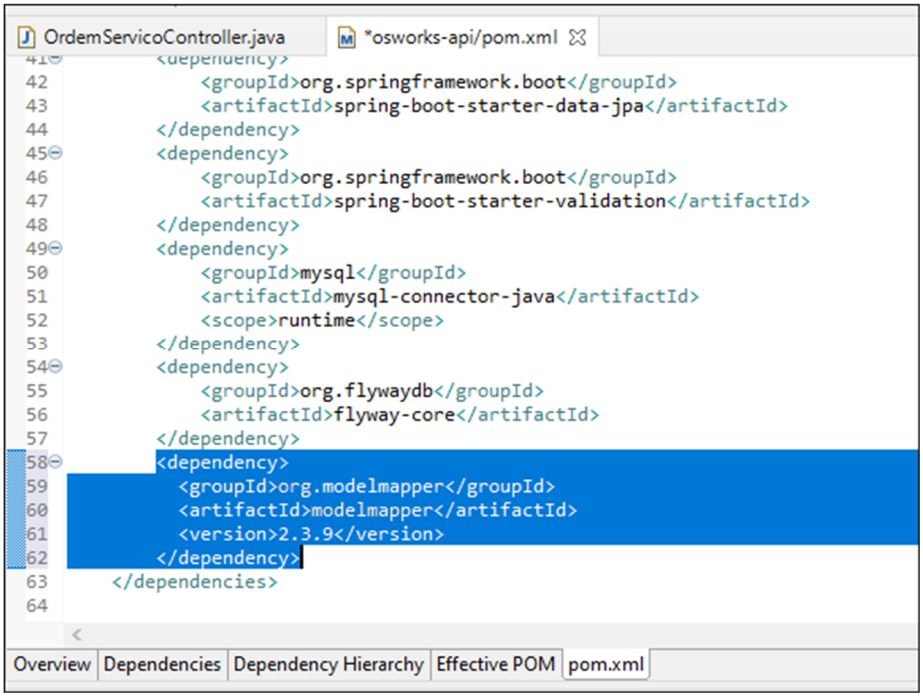

```java
package com.lduran.osworks.api.model;

import java.math.BigDecimal;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class OrdemServicoInput
{
	@Valid
	@NotNull
	private ClienteIdInput cliente;

	@NotBlank
	private String descricao;

	@NotNull
	private BigDecimal preco;
}
```

Buscar no repositório central do Maven o ModelMapper:

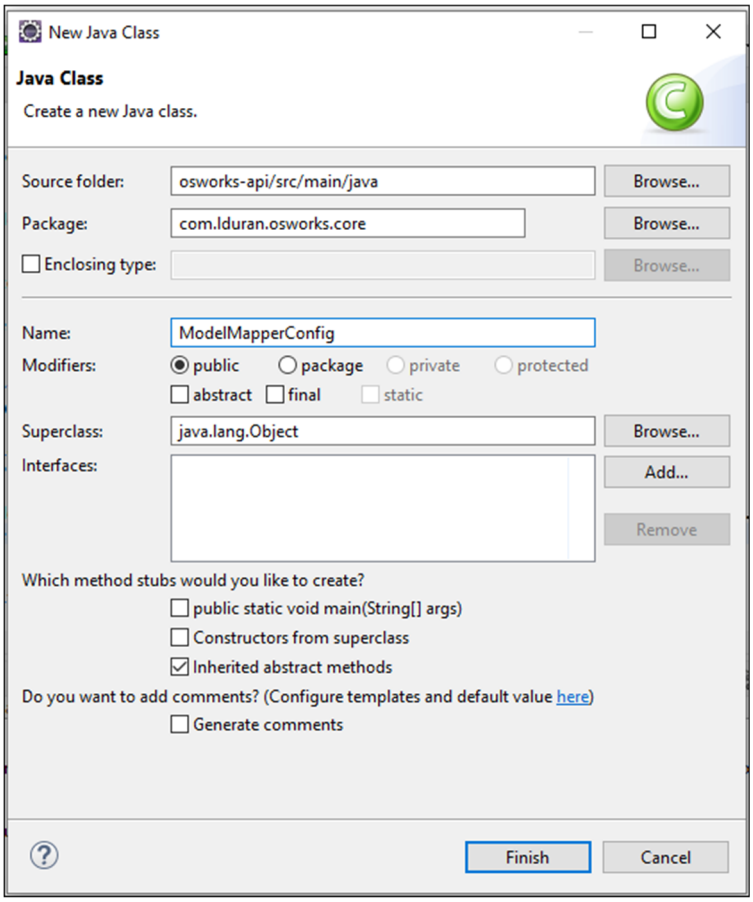




Copiar a configuração da dependência e inserir nas dependências do pom.xml:



Transformar o ModelMapper em uma instância gerenciada pelo Spring, criando uma classe ModelMapperConfig() que retornara uma instância do ModelMapper:




```java
package com.lduran.osworks.core;

import org.modelmapper.ModelMapper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ModelMapperConfig
{
	@Bean
	public ModelMapper modelMapper()
	{
		return new ModelMapper();
	}
}
```

Alterar o Controller da Ordem de Serviço para que trabalhe com as classes DTO da entidade OrdemService(), utilizando o ModelMapper:

```java
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import javax.validation.Valid;

import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
```

```java
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.api.model.OrdemServicoInput;
import com.lduran.osworks.api.model.OrdemServicoModel;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;
import com.lduran.osworks.domain.service.GestaoOrdemServicoService;

@RestController
@RequestMapping("/ordens-servico")
public class OrdemServicoController
{
	@Autowired
	private GestaoOrdemServicoService gestaoOrdemServicoService;

	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	@Autowired
	private ModelMapper modelMapper;

	@GetMapping()
	public List<OrdemServicoModel> listar()
	{
		return this.toCollectionModel(this.ordemServicoRepository.findAll());
	}

	@GetMapping("/{ordemServicoId}")
	public ResponseEntity<OrdemServicoModel> buscar(@PathVariable Long ordemServicoId)
	{
		Optional<OrdemServico> ordemServico = this.ordemServicoRepository.findById(ordemServicoId);
		if(ordemServico.isPresent())
		{
			OrdemServicoModel ordemServicoModel = this.toModel(ordemServico.get());
			return ResponseEntity.ok(ordemServicoModel);
		}

		return ResponseEntity.notFound().build();
	}

	@PostMapping
	@ResponseStatus(HttpStatus.CREATED)
	public OrdemServicoModel criar(@Valid @RequestBody OrdemServicoInput ordemServicoInput)
	{
		OrdemServico ordemServico = this.toEntity(ordemServicoInput);
		return this.toModel(this.gestaoOrdemServicoService.criar(ordemServico));
	}

	@DeleteMapping("/{ordemServicoId}")
	public ResponseEntity<Void> remover(@PathVariable Long ordemServicoId)
	{
		if(!this.ordemServicoRepository.existsById(ordemServicoId))
		{
			return ResponseEntity.notFound().build();
		}

		this.gestaoOrdemServicoService.excluir(ordemServicoId);

		return ResponseEntity.noContent().build();
	}

	private OrdemServicoModel toModel(OrdemServico ordemServico)
	{
		return this.modelMapper.map(ordemServico, OrdemServicoModel.class);
	}

	private List<OrdemServicoModel> toCollectionModel(List<OrdemServico> ordensServico)
	{
		return ordensServico.stream()
				.map(ordemServico -> this.toModel(ordemServico))
				.collect(Collectors.toList());
	}
```

```
        private OrdemServico toEntity(OrdemServicoInput ordemServicoInput)
        {
                return this.modelMapper.map(ordemServicoInput, OrdemServico.class);
        }
}
```

Remover as validações referentes a Ordem de Serviço das entidades:

```
package com.lduran.osworks.domain.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class Cliente
{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @EqualsAndHashCode.Include
        private Long id;

        @NotBlank
        @Size(max=60)
        private String nome;

        @NotBlank
        @Email
        @Size(max=255)
        private String email;

        @NotBlank
        @Size(max=20)
        private String telefone;
}
```

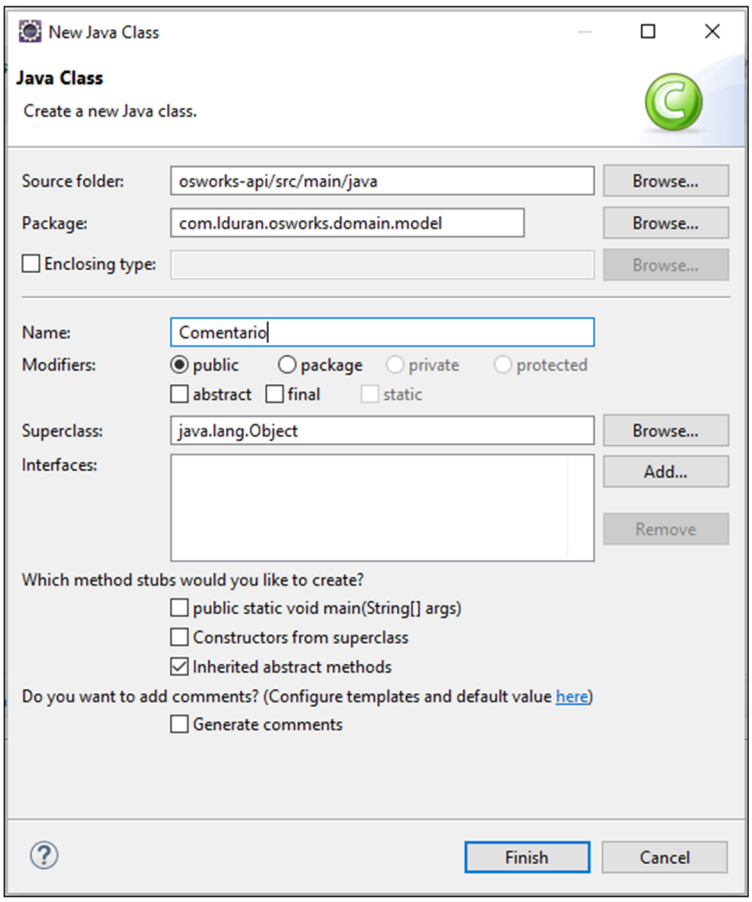
```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.OffsetDateTime;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @EqualsAndHashCode.Include
        private Long id;

        @ManyToOne
        private Cliente cliente;
```

```
        private String descricao;

        private BigDecimal preco;

        @Enumerated(EnumType.STRING)
        private StatusOrdemServico status;

        private OffsetDateTime dataAbertura;

        private OffsetDateTime dataFinalizacao;
}
```

Remover o ValidationGroups(), pois não é mais necessário

Implementar os End-Points de Comentario():

Criar a entidade Comentario():



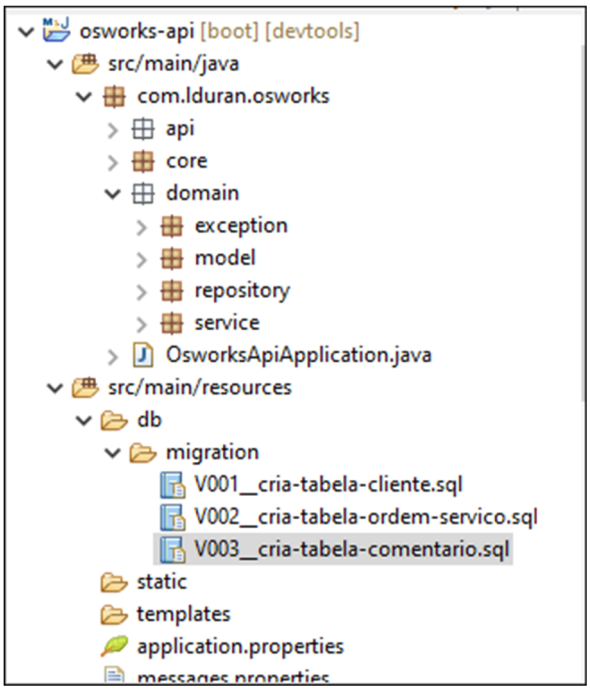

```
package com.lduran.osworks.domain.model;

import java.time.OffsetDateTime;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.validation.constraints.NotBlank;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class Comentario
{
```

```
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @EqualsAndHashCode.Include
        private Long id;

        @ManyToOne
        private OrdemServico ordemServico;

        @NotBlank
        private String descricao;

        private OffsetDateTime dataEnvio;
}
```

Criar a lista de Comentario() na OrdemServico():

```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.OffsetDateTime;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @EqualsAndHashCode.Include
        private Long id;

        @ManyToOne
        private Cliente cliente;

        private String descricao;

        private BigDecimal preco;

        @Enumerated(EnumType.STRING)
        private StatusOrdemServico status;

        private OffsetDateTime dataAbertura;

        private OffsetDateTime dataFinalizacao;

        @OneToMany(mappedBy = "ordemServico")
        private List<Comentario> comentarios = new ArrayList<>();
}
```

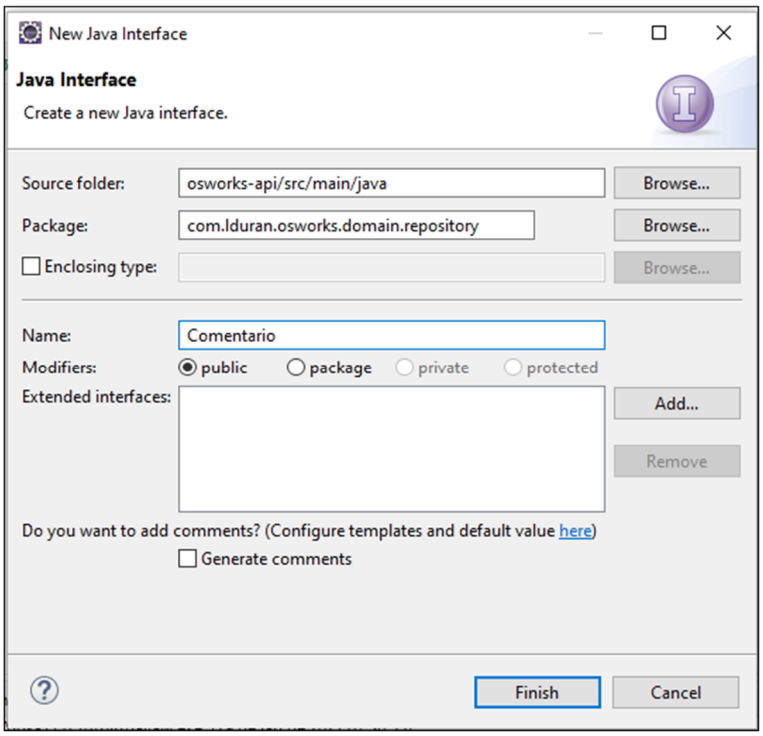
Criar o script sql inicial **V003__cria-tabela-comentario.sql**:



Alterar o conteúdo do script sql inicial **V003__cria-tabela-comentario.sql** com o script de criação da tabela Comentario:

```
create table comentario
(
  id bigint not null auto_increment,
  ordem_servico_id bigint not null,
  descricao text not null,
  data_envio datetime not null,
  primary key (id)
);

alter table comentario add constraint fk_comentario_ordem_servico foreign key (ordem_servico_id) references ordem_servico (id);
```

Criar a interface Repositório para o Comentario:

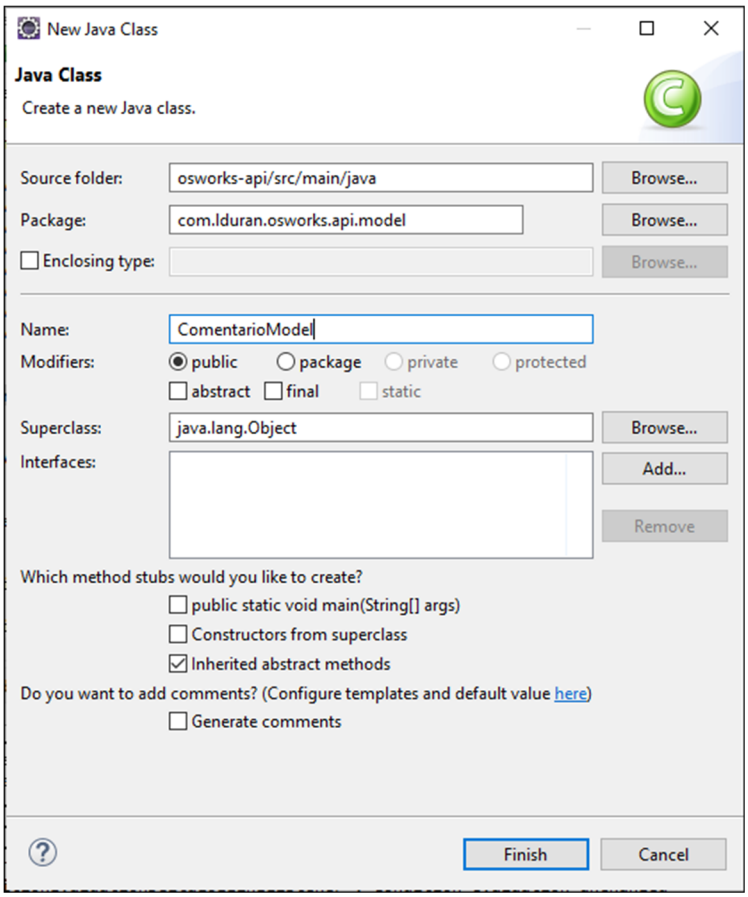


```java
package com.lduran.osworks.domain.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.lduran.osworks.domain.model.Comentario;

@Repository
public interface ComentarioRepository extends JpaRepository<Comentario, Long>
{

}
```

Criar um método adicionarComentario() em GestaoOrdemServicoService():

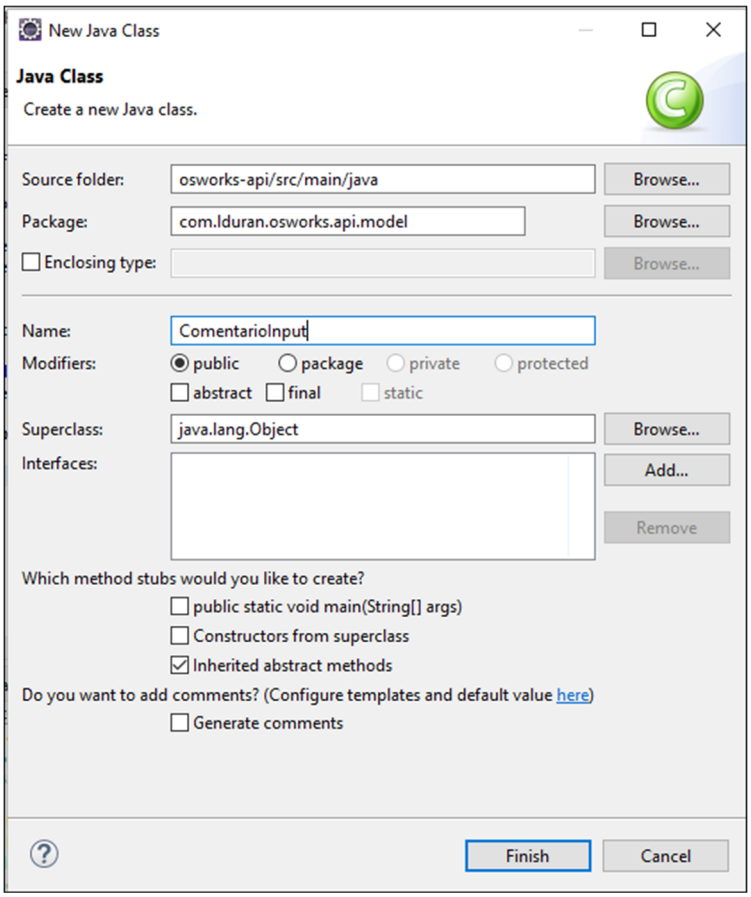
```java
package com.lduran.osworks.domain.service;

import java.time.OffsetDateTime;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.exception.NegocioException;
import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.model.Comentario;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.model.StatusOrdemServico;
import com.lduran.osworks.domain.repository.ClienteRepository;
import com.lduran.osworks.domain.repository.ComentarioRepository;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;

@Service
public class GestaoOrdemServicoService
{
	@Autowired
	private OrdemServicoRepository ordemServicoRepository;

	@Autowired
	private ClienteRepository clienteRepository;

	@Autowired
	private ComentarioRepository comentarioRepository;

	public OrdemServico criar(OrdemServico ordemServico)
	{
		Cliente cliente = this.clienteRepository.findById(ordemServico.getCliente().getId())
				.orElseThrow(() -> new NegocioException("Cliente não encontrado."));

		ordemServico.setCliente(cliente);
		ordemServico.setStatus(StatusOrdemServico.ABERTA);
		ordemServico.setDataAbertura(OffsetDateTime.now());

		return this.ordemServicoRepository.save(ordemServico);
	}

	public void excluir(Long ordemServicoId)
	{
		this.ordemServicoRepository.deleteById(ordemServicoId);
	}

	public Comentario adicionarComentario(Long ordemServicoId, String descricao)
	{
		OrdemServico ordemServico = this.ordemServicoRepository.findById(ordemServicoId)
				.orElseThrow(() -> new NegocioException("Ordem de Serviço não encontrada."));

		Comentario comentario = new Comentario();
		comentario.setOrdemServico(ordemServico);
		comentario.setDescricao(descricao);
		comentario.setDataEnvio(OffsetDateTime.now());

		return this.comentarioRepository.save(comentario);
	}
}
```

Criar a classe DTO para a entidade Comentario():



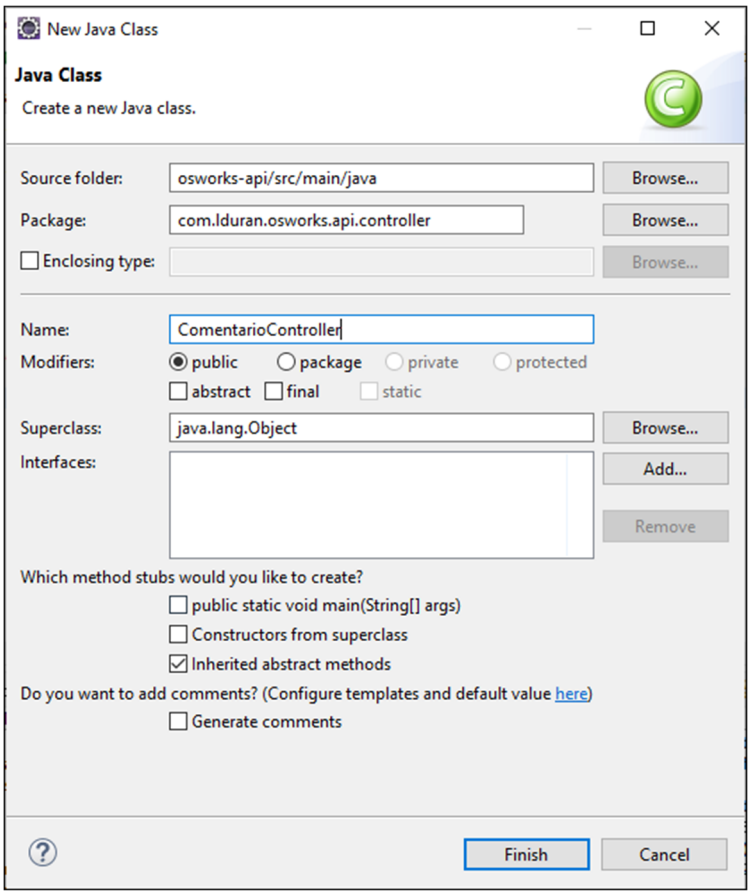

```java
package com.lduran.osworks.api.model;

import java.time.OffsetDateTime;

import com.lduran.osworks.domain.model.OrdemServico;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class ComentarioModel
{
	private Long id;
	private String descricao;
	private OffsetDateTime dataEnvio;
}
```

Criar a classe DTO para entrada de dados da entidade Comentario():




```java
package com.lduran.osworks.api.model;

import javax.validation.constraints.NotBlank;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class ComentarioInput
{
	@NotBlank
	private String descricao;
}
```

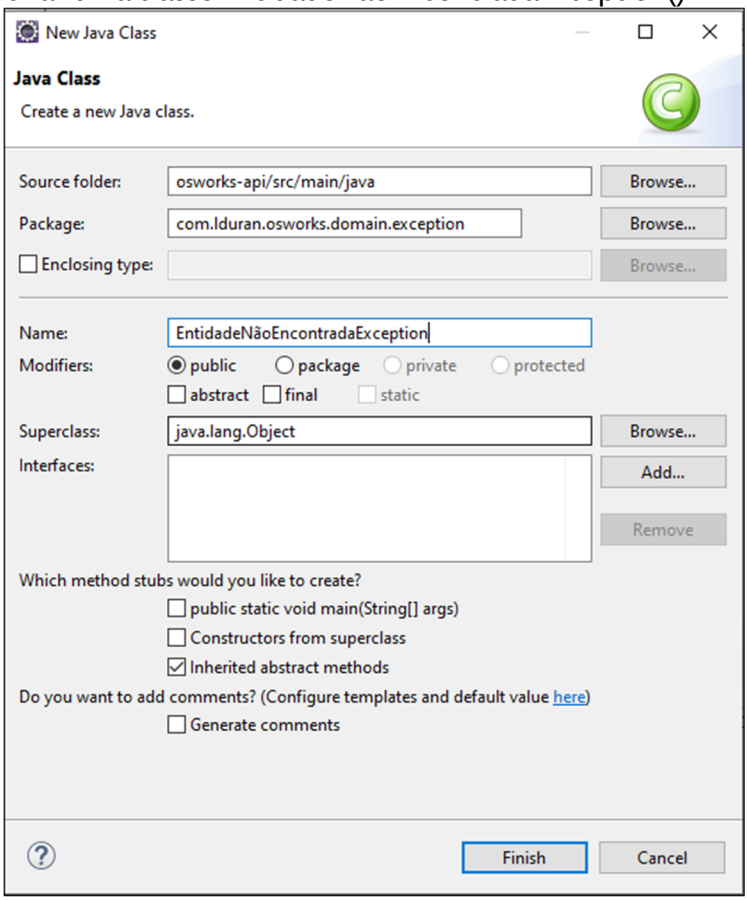
Criar o ComentarioController():




```java
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import javax.validation.Valid;

import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.api.model.ComentarioInput;
import com.lduran.osworks.api.model.ComentarioModel;
import com.lduran.osworks.domain.model.Comentario;
import com.lduran.osworks.domain.repository.ComentarioRepository;
import com.lduran.osworks.domain.service.GestaoOrdemServicoService;

@RestController
@RequestMapping("/ordens-servico/{ordemServicoId}/comentarios")
public class ComentarioController
{
	@Autowired
	private ComentarioRepository comentarioRepository;

	@Autowired
	private GestaoOrdemServicoService gestaoOrdemServicoService;

	@Autowired
	private ModelMapper modelMapper;
```

```java
	@GetMapping
	public List<ComentarioModel> listar()
	{
		return this.toCollectionModel(this.comentarioRepository.findAll());
	}

	@GetMapping("/{comentarioId}")
	public ResponseEntity<ComentarioModel> buscar(@PathVariable Long comentarioId)
	{
		Optional<Comentario> comentario = this.comentarioRepository.findById(comentarioId);
		if(comentario.isPresent())
		{
			return ResponseEntity.ok(this.toModel(comentario.get()));
		}

		return ResponseEntity.notFound().build();
	}

	@PostMapping
	@ResponseStatus(HttpStatus.CREATED)
	public ComentarioModel adicionar(@PathVariable Long ordemServicoId, @Valid @RequestBody ComentarioInput comentarioInput)
	{
		Comentario comentario = this.gestaoOrdemServicoService.adicionarComentario(ordemServicoId, comentarioInput.getDescricao());
		return this.toModel(comentario);
	}

	private ComentarioModel toModel(Comentario comentario)
	{
		return this.modelMapper.map(comentario, ComentarioModel.class);
	}

	private List<ComentarioModel> toCollectionModel(List<Comentario> comentarios)
	{
		return comentarios.stream()
				.map(comentario -> this.toModel(comentario))
				.collect(Collectors.toList());
	}
}
```

Criar uma classe EntidadeNãoEncontradaException():



```java
package com.lduran.osworks.domain.exception;

public class EntidadeNãoEncontradaException extends NegocioException
{
	private static final long serialVersionUID = 1L;

	public EntidadeNãoEncontradaException(String message)
	{
		super(message);
	}
}
```

Criar uma nova ExceptionHandler em ApiExceptionHandler() para tratar as exceções de EntidadeNãoEncontradaException():

```java
package com.lduran.osworks.api.exceptionhandler;

import java.time.OffsetDateTime;
import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import com.lduran.osworks.domain.exception.EntidadeNãoEncontradaException;
import com.lduran.osworks.domain.exception.NegocioException;

@ControllerAdvice
public class ApiExceptionHandler extends ResponseEntityExceptionHandler
{
	@Autowired
	private MessageSource messageSource;

	@ExceptionHandler(EntidadeNãoEncontradaException.class)
	public ResponseEntity<Object> handleNegocio(EntidadeNãoEncontradaException ex, WebRequest request)
	{
		var status = HttpStatus.NOT_FOUND;

		var problema = new Problema();
		problema.setStatus(status.value());
		problema.setTitulo(ex.getMessage());
		problema.setDataHora(OffsetDateTime.now());

		return super.handleExceptionInternal(ex, problema, new HttpHeaders(), status, request);
	}

	@ExceptionHandler(NegocioException.class)
	public ResponseEntity<Object> handleNegocio(NegocioException ex, WebRequest request)
	{
		var status = HttpStatus.BAD_REQUEST;

		var problema = new Problema();
		problema.setStatus(status.value());
		problema.setTitulo(ex.getMessage());
		problema.setDataHora(OffsetDateTime.now());

		return super.handleExceptionInternal(ex, problema, new HttpHeaders(), status, request);
	}

	@Override
	protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
			HttpHeaders headers, HttpStatus status, WebRequest request)
	{
		var campos = new ArrayList<Campo>();

		for(ObjectError error : ex.getBindingResult().getAllErrors())
		{
```

```
                        String nome = ((FieldError)error).getField();
                        String mensagem = this.messageSource.getMessage(error,LocaleContextHolder.getLocale());

                        campos.add(new Campo(nome,mensagem));
                }

                var problema = new Problema();

                problema.setStatus(status.value());

                problema.setTitulo("Um ou mais campos estão inválidos. " +
                                "Faça o preenchimento correto e tente novamente.");

                problema.setDataHora(OffsetDateTime.now());

                problema.setCampos(campos);

                return super.handleExceptionInternal(ex, problema, headers, status, request);
        }
}
```

Utilizar a ExceptionHandler de tratamento das exceções de EntidadeNãoEncontradaException() no método adicionarComentario() da GestaoOrdemServicoService():

```
package com.lduran.osworks.domain.service;

import java.time.OffsetDateTime;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.exception.EntidadeNãoEncontradaException;
import com.lduran.osworks.domain.exception.NegocioException;
import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.model.Comentario;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.model.StatusOrdemServico;
import com.lduran.osworks.domain.repository.ClienteRepository;
import com.lduran.osworks.domain.repository.ComentarioRepository;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;

@Service
public class GestaoOrdemServicoService
{
        @Autowired
        private OrdemServicoRepository ordemServicoRepository;

        @Autowired
        private ClienteRepository clienteRepository;

        @Autowired
        private ComentarioRepository comentarioRepository;

        public OrdemServico criar(OrdemServico ordemServico)
        {
                Cliente cliente = this.clienteRepository.findById(ordemServico.getCliente().getId())
                                .orElseThrow(() -> new NegocioException("Cliente não encontrado."));

                ordemServico.setCliente(cliente);
                ordemServico.setStatus(StatusOrdemServico.ABERTA);
                ordemServico.setDataAbertura(OffsetDateTime.now());

                return this.ordemServicoRepository.save(ordemServico);
        }

        public void excluir(Long ordemServicoId)
        {
                this.ordemServicoRepository.deleteById(ordemServicoId);
        }

        public Comentario adicionarComentario(Long ordemServicoId, String descricao)
        {
                OrdemServico ordemServico = this.ordemServicoRepository.findById(ordemServicoId)
                                .orElseThrow(() -> new EntidadeNãoEncontradaException("Ordem de Serviço não encontrada."));
```

```
                Comentario comentario = new Comentario();
                comentario.setOrdemServico(ordemServico);
                comentario.setDescricao(descricao);
                comentario.setDataEnvio(OffsetDateTime.now());

                return this.comentarioRepository.save(comentario);
        }
}
```

Criar o método finalizar() em OrdemServico():

```
package com.lduran.osworks.domain.model;

import java.math.BigDecimal;
import java.time.OffsetDateTime;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

import com.lduran.osworks.domain.exception.NegocioException;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter @Setter
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
public class OrdemServico
{
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @EqualsAndHashCode.Include
        private Long id;

        @ManyToOne
        private Cliente cliente;

        private String descricao;

        private BigDecimal preco;

        @Enumerated(EnumType.STRING)
        private StatusOrdemServico status;

        private OffsetDateTime dataAbertura;

        private OffsetDateTime dataFinalizacao;

        @OneToMany(mappedBy = "ordemServico")
        private List<Comentario> comentarios = new ArrayList<>();

        private boolean podeSerFinalizada()
        {
                return StatusOrdemServico.FINALIZADA.equals(this.getStatus());
        }

        private boolean naoPodeSerFinalizada()
        {
                return !StatusOrdemServico.FINALIZADA.equals(this.getStatus());
        }

        public void finalizar()
        {
                if(this.naoPodeSerFinalizada())
                {
                        throw new NegocioException("Ordem de serviço não pode ser finalizada");
                }
```

```
                this.setStatus(StatusOrdemServico.FINALIZADA);
                this.setDataFinalizacao(OffsetDateTime.now());
        }
}
```

Criar o método finalizar() em GestaoOrdemServicoService():

```
package com.lduran.osworks.domain.service;

import java.time.OffsetDateTime;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.lduran.osworks.domain.exception.EntidadeNãoEncontradaException;
import com.lduran.osworks.domain.exception.NegocioException;
import com.lduran.osworks.domain.model.Cliente;
import com.lduran.osworks.domain.model.Comentario;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.model.StatusOrdemServico;
import com.lduran.osworks.domain.repository.ClienteRepository;
import com.lduran.osworks.domain.repository.ComentarioRepository;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;

@Service
public class GestaoOrdemServicoService
{
        @Autowired
        private OrdemServicoRepository ordemServicoRepository;

        @Autowired
        private ClienteRepository clienteRepository;

        @Autowired
        private ComentarioRepository comentarioRepository;

        public OrdemServico criar(OrdemServico ordemServico)
        {
                Cliente cliente = this.clienteRepository.findById(ordemServico.getCliente().getId())
                                .orElseThrow(() -> new NegocioException("Cliente não encontrado."));

                ordemServico.setCliente(cliente);
                ordemServico.setStatus(StatusOrdemServico.ABERTA);
                ordemServico.setDataAbertura(OffsetDateTime.now());

                return this.ordemServicoRepository.save(ordemServico);
        }

        public void excluir(Long ordemServicoId)
        {
                this.ordemServicoRepository.deleteById(ordemServicoId);
        }

        public void finaliza(Long ordemServicoId)
        {
                OrdemServico ordemServico = buscar(ordemServicoId);

                ordemServico.finalizar();

                this.ordemServicoRepository.save(ordemServico);
        }

        public Comentario adicionarComentario(Long ordemServicoId, String descricao)
        {
                OrdemServico ordemServico = buscar(ordemServicoId);

                Comentario comentario = new Comentario();
                comentario.setOrdemServico(ordemServico);
                comentario.setDescricao(descricao);
                comentario.setDataEnvio(OffsetDateTime.now());

                return this.comentarioRepository.save(comentario);
        }

        private OrdemServico buscar(Long ordemServicoId)
```

```
        {
                return this.ordemServicoRepository.findById(ordemServicoId)
                                .orElseThrow(() -> new EntidadeNãoEncontradaException("Ordem de Serviço não encontrada."));
        }
}
```

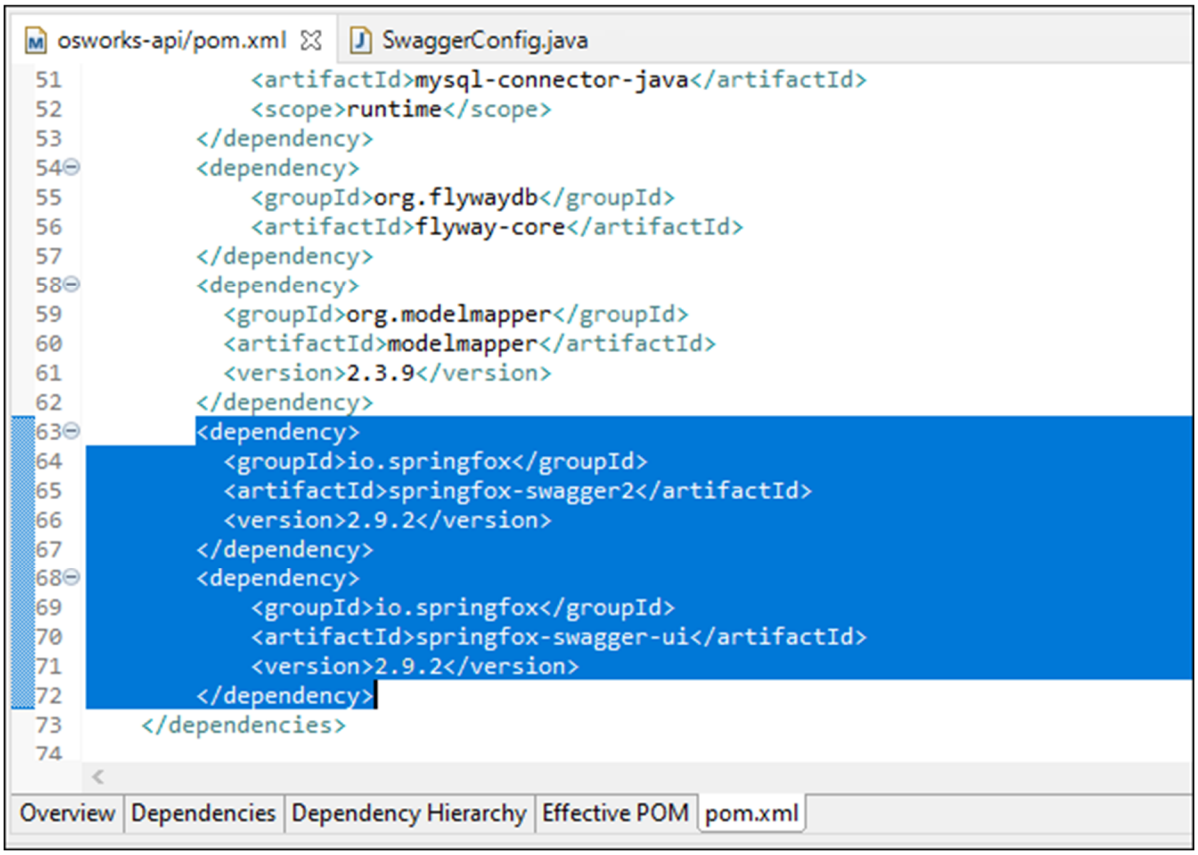
Criar o End-Point de Finalização de Ordem de Serviço no OrdemServicoController ():

```
package com.lduran.osworks.api.controller;

import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import javax.validation.Valid;

import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.lduran.osworks.api.model.OrdemServicoInput;
import com.lduran.osworks.api.model.OrdemServicoModel;
import com.lduran.osworks.domain.model.OrdemServico;
import com.lduran.osworks.domain.repository.OrdemServicoRepository;
import com.lduran.osworks.domain.service.GestaoOrdemServicoService;

@RestController
@RequestMapping("/ordens-servico")
public class OrdemServicoController
{
        @Autowired
        private GestaoOrdemServicoService gestaoOrdemServicoService;

        @Autowired
        private OrdemServicoRepository ordemServicoRepository;

        @Autowired
        private ModelMapper modelMapper;

        @GetMapping()
        public List<OrdemServicoModel> listar()
        {
                return this.toCollectionModel(this.ordemServicoRepository.findAll());
        }

        @GetMapping("/{ordemServicoId}")
        public ResponseEntity<OrdemServicoModel> buscar(@PathVariable Long ordemServicoId)
        {
                Optional<OrdemServico> ordemServico = this.ordemServicoRepository.findById(ordemServicoId);
                if(ordemServico.isPresent())
                {
                        OrdemServicoModel ordemServicoModel = this.toModel(ordemServico.get());
                        return ResponseEntity.ok(ordemServicoModel);
                }

                return ResponseEntity.notFound().build();
        }

        @PostMapping
        @ResponseStatus(HttpStatus.CREATED)
        public OrdemServicoModel criar(@Valid @RequestBody OrdemServicoInput ordemServicoInput)
        {
                OrdemServico ordemServico = this.toEntity(ordemServicoInput);
                return this.toModel(this.gestaoOrdemServicoService.criar(ordemServico));
        }
```

```
	@PutMapping("/{ordemServicoId}/finalizacao")
	@ResponseStatus(HttpStatus.NO_CONTENT)
	public void finalizar(@PathVariable Long ordemServicoId)
	{
		this.gestaoOrdemServicoService.finalizar(ordemServicoId);
	}

	@DeleteMapping("/{ordemServicoId}")
	public ResponseEntity<Void> remover(@PathVariable Long ordemServicoId)
	{
		if(!this.ordemServicoRepository.existsById(ordemServicoId))
		{
			return ResponseEntity.notFound().build();
		}

		this.gestaoOrdemServicoService.excluir(ordemServicoId);

		return ResponseEntity.noContent().build();
	}

	private OrdemServicoModel toModel(OrdemServico ordemServico)
	{
		return this.modelMapper.map(ordemServico, OrdemServicoModel.class);
	}

	private List<OrdemServicoModel> toCollectionModel(List<OrdemServico> ordensServico)
	{
		return ordensServico.stream()
				.map(ordemServico -> this.toModel(ordemServico))
				.collect(Collectors.toList());
	}

	private OrdemServico toEntity(OrdemServicoInput ordemServicoInput)
	{
		return this.modelMapper.map(ordemServicoInput, OrdemServico.class);
	}
}
```

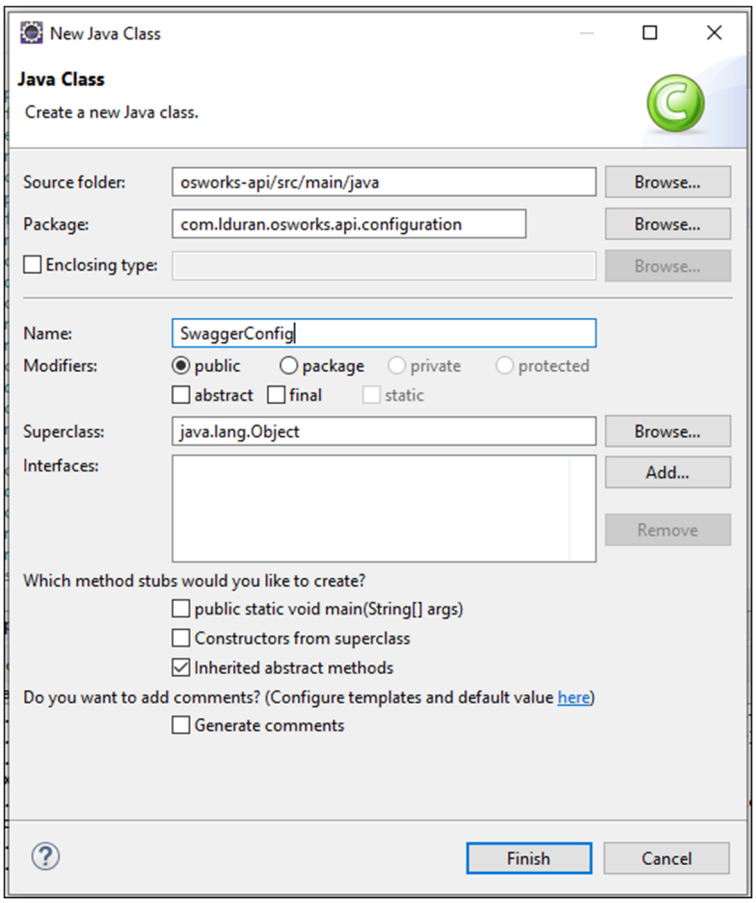
Documentando a API Spring Boot com o Swagger:

Inserir as dependênciad de SpringFox-Swagger2 e SpringFox-Swagger Ui nas dependências do pom.xml:



```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

Agora para que o arquivo de especificação da API seja criado, é necessário habilitar o Swagger na aplicação. Para isso, adicione nela uma classe chamada **SwaggerConfig**, com o conteúdo abaixo:




```
package com.lduran.osworks.api.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
	@Bean
	public Docket api() {
		return new Docket(DocumentationType.SWAGGER_2)
				.select()
				.apis(RequestHandlerSelectors.any())
				.paths(PathSelectors.any())
				.build();
	}
}
```

Ao executar a API, o Swagger UI estará disponível em **http://localhost:8080/swagger-ui.html**: