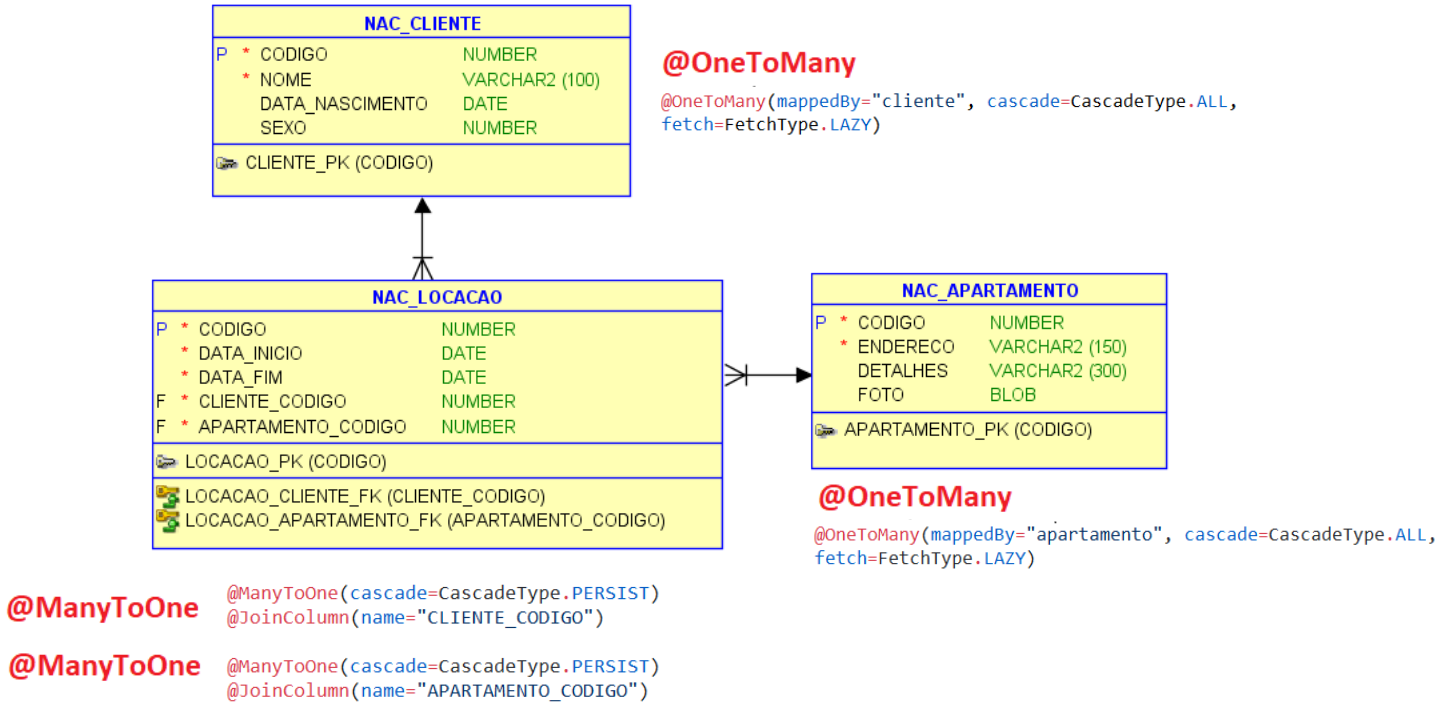
Criando POM

pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>06-JPA-JPQL-Apostila</groupId>
    <artifactId>06-JPA-JPQL-Apostila</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <build>
      <sourceDirectory>src</sourceDirectory>
      <plugins>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.7.0</version>
          <configuration>
            <source>1.8</source>
            <target>1.8</target>
          </configuration>
        </plugin>
      </plugins>
    </build>
      <dependencies>
                <dependency>
                        <groupId>org.hibernate</groupId>
                        <artifactId>hibernate-core</artifactId>
                        <version>5.2.12.Final</version>
                </dependency>
                <dependency>
                        <groupId>com.h2</groupId>
                        <artifactId>h2</artifactId>
                        <version>8</version>
                        <scope>system</scope>
                        <systemPath>D:/h2.jar</systemPath>
                </dependency>
         </dependencies>
     </project>
```

Exemplo de diagrama



Criando br.com.fiap.nac.entity

Apartamento

```
@Entity(name="Apartamento")
@Table(name="NAC_APARTAMENTO")
@SequenceGenerator(name="apartamento", sequenceName="SEQ_NAC_APARTAMENTO",
allocationSize=1)
public class Apartamento {

	@Id
	@Column(name="CODIGO")
	@GeneratedValue(generator="apartamento",
strategy=GenerationType.SEQUENCE)
	private int codigo;

	@Column(name="ENDERECO", length=150, nullable=false)
	private String endereco;

	@Column(name="DETALHES", length=300)
	private String detalhes;

	@Lob
	@Column(name="FOTO")
	private byte[] foto;

	@OneToMany(mappedBy="apartamento", cascade=CascadeType.ALL,
fetch=FetchType.LAZY)
	private List<Locacao> locacoes;

	public Apartamento() {

	}

	public Apartamento(String endereco, String detalhes, byte[] foto) {
		this.endereco = endereco;
		this.detalhes = detalhes;
		this.foto = foto;
	}

	public int getCodigo() {
		return codigo;
	}
```

```
	public void setCodigo(int codigo) {
		this.codigo = codigo;
	}

	public String getEndereco() {
		return endereco;
	}

	public void setEndereco(String endereco) {
		this.endereco = endereco;
	}

	public String getDetalhes() {
		return detalhes;
	}

	public void setDetalhes(String detalhes) {
		this.detalhes = detalhes;
	}

	public byte[] getFoto() {
		return foto;
	}

	public void setFoto(byte[] foto) {
		this.foto = foto;
	}

	public List<Locacao> getLocacoes() {
		return locacoes;
	}

	public void setLocacoes(List<Locacao> locacoes) {
		this.locacoes = locacoes;
	}

}
```

Cliente

```
@Entity(name="Cliente")
@Table(name="NAC_CLIENTE")
@SequenceGenerator(name="cliente", sequenceName="SEQ_NAC_CLIENTE", allocationSize=1)
public class Cliente {

	@Id
	@Column(name="CODIGO")
	@GeneratedValue(generator="cliente", strategy=GenerationType.SEQUENCE)
	private int codigo;

	@Column(name="NOME", length=100, nullable=false)
	private String nome;

	@Temporal(TemporalType.DATE)
	@Column(name="DATA_NASCIMENTO")
	private Calendar dataNascimento;

	@Enumerated(EnumType.ORDINAL)
	@Column(name="SEXO")
	private Sexo sexo;

	@OneToMany(mappedBy="cliente", cascade=CascadeType.ALL, fetch=FetchType.LAZY)
	private List<Locacao> locacoes;

	public Cliente() {
	}
	public Cliente(String nome, Calendar dataNascimento, Sexo sexo) {
		super();
		this.nome = nome;
		this.dataNascimento = dataNascimento;
		this.sexo = sexo;
	}
	public int getCodigo() {
		return codigo;
```

```
	}
	public void setCodigo(int codigo) {
		this.codigo = codigo;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public Calendar getDataNascimento() {
		return dataNascimento;
	}
	public void setDataNascimento(Calendar dataNascimento) {
		this.dataNascimento = dataNascimento;
	}
	public Sexo getSexo() {
		return sexo;
	}
	public void setSexo(Sexo sexo) {
		this.sexo = sexo;
	}
	public List<Locacao> getLocacoes() {
		return locacoes;
	}
	public void setLocacoes(List<Locacao> locacoes) {
		this.locacoes = locacoes;
	}
}
```

Locação

```
@Entity(name="Locacao")
@Table(name="NAC_LOCACAO")
@SequenceGenerator(name="locacao", sequenceName="SEQ_NAC_LOCACAO", allocationSize=1)
public class Locacao {

	@Id
	@Column(name="CODIGO")
	@GeneratedValue(generator="locacao", strategy=GenerationType.SEQUENCE)
	private int codigo;

	@Column(name="DATA_INICIO", nullable=false)
	@Temporal(TemporalType.DATE)
	private Calendar dataInicio;

	@Column(name="DATA_FIM", nullable=false)
	@Temporal(TemporalType.DATE)
	private Calendar dataFim;

	@ManyToOne(cascade=CascadeType.PERSIST)
	@JoinColumn(name="CLIENTE_CODIGO")
	private Cliente cliente;

	@ManyToOne(cascade=CascadeType.PERSIST)
	@JoinColumn(name="APARTAMENTO_CODIGO")
	private Apartamento apartamento;

	public Locacao() {

	}

	public Locacao(Calendar dataInicio, Calendar dataFim, Cliente cliente, Apartamento
apartamento) {
		super();
		this.dataInicio = dataInicio;
		this.dataFim = dataFim;
		this.cliente = cliente;
		this.apartamento = apartamento;
	}

	public int getCodigo() {
		return codigo;
```

```
	}

	public void setCodigo(int codigo) {
		this.codigo = codigo;
	}

	public Calendar getDataInicio() {
		return dataInicio;
	}

	public void setDataInicio(Calendar dataInicio) {
		this.dataInicio = dataInicio;
	}

	public Calendar getDataFim() {
		return dataFim;
	}

	public void setDataFim(Calendar dataFim) {
		this.dataFim = dataFim;
	}

	public Cliente getCliente() {
		return cliente;
	}

	public void setCliente(Cliente cliente) {
		this.cliente = cliente;
	}

	public Apartamento getApartamento() {
		return apartamento;
	}

	public void setApartamento(Apartamento apartamento) {
		this.apartamento = apartamento;
	}
}
```

Sexo

```
public enum Sexo {
	MASCULINO, FEMININO, OUTRO;
}
```

Criando br.com.fiap.nac.DAO

ApartamentoDAO

```
package
br.com.fiap.nac.dao;

                        import java.util.List;
                        import br.com.fiap.nac.entity.Apartamento;

                        public interface ApartamentoDAO extends GenericDAO<Apartamento, Integer>{

                               List<Apartamento> buscarPorDesc(String detalhes);
                        }
```

ClienteDAO

```
package
br.com.fiap.nac.dao;

                        import br.com.fiap.nac.entity.Cliente;

                        public interface ClienteDAO extends GenericDAO<Cliente, Integer>{
                        }
```

LocacaoDAO

```
   package
   br.com.fiap.nac.dao;

                         import java.util.Calendar;
                         import java.util.List;

                         import
                         br.com.fiap.nac.entity.Locacao;

                         public interface LocacaoDAO
                         extends GenericDAO<Locacao,
                         Integer>{

                                List<Locacao>
                         buscarDataInicio(Calendar
                         dataUm, Calendar dataDois);

                                int
                         buscarPorCliente(int
                         codigoCliente);
                         }
```

GenericDAO

```
package
br.com.fiap.nac.dao;

                        import java.util.List;

                        import br.com.fiap.nac.exception.DBException;
                        import br.com.fiap.nac.exception.IdNotFoundException;

                        public interface GenericDAO<T,K> {

                               void cadastrar(T entity);
                               void remover(K codigo) throws IdNotFoundException;
                               T pesquisar(K codigo);
                               void alterar(T entity);

                               List<T> listar();
                               void salvar() throws DBException;
                        }
```

Criando br.com.fiap.nac.DAOImpl

ApartamentoDAOImpl

```
public class ApartamentoDAOImpl extends GenericDAOImpl<Apartamento, Integer> implements
ApartamentoDAO{

	public ApartamentoDAOImpl(EntityManager em) {
		super(em);
	}

	@Override
	public List<Apartamento> buscarPorDesc(String detalhes) {
		Query q = em.createQuery("from Apartamento a where a.DETALHES like :detalhes");
		q.setParameter("detalhes", "%"+detalhes+"%");
		return null;
	}

}
```

ClienteDAOImpl

```
public class ClienteDAOImpl extends GenericDAOImpl<Cliente, Integer> implements ClienteDAO{

	public ClienteDAOImpl(EntityManager em) {
		super(em);
	}
}
```

LocacaoDAOImpl

```
public class LocacaoDAOImpl extends GenericDAOImpl<Locacao, Integer> implements LocacaoDAO{


	public LocacaoDAOImpl(EntityManager em) {
		super(em);
	}


	@Override
	@SuppressWarnings("unchecked")
	public List<Locacao> buscarDataInicio(Calendar dataUm, Calendar dataDois) {
		Query q = em.createQuery("from Locacao l where l.DATA_INICIO between :dataUm and
:dataDois");
		q.setParameter("dataUm", dataUm);
		q.setParameter("dataDois", dataDois);
		return q.getResultList();
	}


	@Override
	public int buscarPorCliente(int codigoCliente) {
		Query q = em.createQuery("select count(*) from Locacao l where l.CLIENTE_CODIGO
= :idCliente");
		q.setParameter("idCliente", codigoCliente);
		int total = (Integer)q.getSingleResult();
		return total;
	}
}
```

GenericDAOImpl

```
public class GenericDAOImpl<T,K> implements GenericDAO<T, K>{
```

```
	protected EntityManager em;
	private Class<T> classe;

	@SuppressWarnings("unchecked")
	public GenericDAOImpl(EntityManager em) {
		this.em = em;
		classe = (Class<T>) ((ParameterizedType)getClass()
			.getGenericSuperclass()).getActualTypeArguments()[0];
	}
	@Override
	public void cadastrar(T entity) {
		em.persist(entity);
	}
	@Override
	public void alterar(T entity) {
		em.merge(entity);
	}
	@Override
	public void remover(K codigo) throws IdNotFoundException {
		T entidade = pesquisar(codigo);
		if(entidade == null)
			throw new IdNotFoundException();
		em.remove(entidade);
	}
	@Override
	public T pesquisar(K codigo) {
		return em.find(classe, codigo);
	}
	@Override
	public void salvar() throws DBException{
		try {
			em.getTransaction().begin();
			em.getTransaction().commit();
		}catch(Exception e){
			if (em.getTransaction().isActive())
				em.getTransaction().rollback();
			throw new DBException("Erro no commit", e);
		}
	}
	@Override
	public List<T> listar() {
		return em.createQuery("from "
				+ classe.getName(), classe)
				.getResultList();
	}
}
```

Criando br.com.fiap.nac.exception

DBException

```
@SuppressWarnings("serial")
public class DBException extends Exception {
	public DBException() {
		super();
	}
	public DBException(String message, Throwable cause, boolean enableSuppression, boolean
writableStackTrace) {
		super(message, cause, enableSuppression, writableStackTrace);
	}
	public DBException(String message, Throwable cause) {
		super(message, cause);
	}
	public DBException(String message) {
		super(message);
	}
	public DBException(Throwable cause) {
		super(cause);
	}
}
```

IdNotFoundException

```
@SuppressWarnings("serial")
public class IdNotFoundException extends Exception {

	public IdNotFoundException() {
		super();
	}
	public IdNotFoundException(String message, Throwable cause, boolean enableSuppression,
boolean writableStackTrace) {
		super(message, cause, enableSuppression, writableStackTrace);
	}
	public IdNotFoundException(String message, Throwable cause) {
		super(message, cause);
	}
	public IdNotFoundException(String message) {
		super(message);
	}
	public IdNotFoundException(Throwable cause) {
		super(cause);
	}
}
```

Criando br.com.fiap.nac.singleton

EntityManagerFactorySingleton

```
public class EntityManagerFactorySingleton {

	private static EntityManagerFactory factory;

	private EntityManagerFactorySingleton(){
		super();
	}

	public static EntityManagerFactory getInstance(){
		if(factory == null)
			factory = Persistence.createEntityManagerFactory("CLIENTE_ORACLE");
		return factory;
	}
}
```

Criando br.com.fiap.nac.teste

Teste

```
public class Teste {
	public static void main(String[] args) {

		EntityManagerFactory fabrica = EntityManagerFactorySingleton.getInstance();
		EntityManager em = fabrica.createEntityManager();

		ClienteDAO clienteDao = new ClienteDAOImpl(em);
		ApartamentoDAO apartamentoDao = new ApartamentoDAOImpl(em);
		LocacaoDAO locacaoDao = new LocacaoDAOImpl(em);

		Cliente cliente = new Cliente("Gabriel", new GregorianCalendar(28, Calendar.MARCH,
1999), Sexo.MASCULINO);
		Apartamento apartamento = new Apartamento("Av. Paulista, 1124", "Consolacao", null);
		Locacao locacao = new Locacao(new GregorianCalendar(12, Calendar.APRIL, 2013), new
GregorianCalendar(12, Calendar.APRIL, 2013), cliente, apartamento);

		try {
			clienteDao.cadastrar(cliente);
			apartamentoDao.cadastrar(apartamento);
			locacaoDao.cadastrar(locacao);
		} catch(Exception e) {
			e.printStackTrace();
		}
		em.close();
		fabrica.close();
	}
}
```