Disciplina: Nível 4: Vamos Integrar Sistemas
Semestre: 1º - 2024
Aluno: DAVIDSON PEREIRA DE OLIVEIRA
Matrícula: 2022.11.54585-5
Repositório no GIT: dev-davidson/RPG0017---Vamos-integrar-sistemas (github.com)

Relatório discente de acompanhamento

👉 1º Procedimento | Camadas de Persistência e Controle

Códigos:

Model/Movimento

```
package cadastroee.model;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "Movimento")
@NamedQueries({
@NamedQuery(name = "Movimento.findAll",
        query = "SELECT m FROM Movimento m"),
@NamedQuery(name = "Movimento.findByIdMovimento",
        query = "SELECT m FROM Movimento m WHERE m.idMovimento = :idMovimento"),
@NamedQuery(name = "Movimento.findByQuantidade",
        query = "SELECT m FROM Movimento m WHERE m.quantidade = :quantidade"),
@NamedQuery(name = "Movimento.findByTipo",
        query = "SELECT m FROM Movimento m WHERE m.tipo = :tipo"),
@NamedQuery(name = "Movimento.findByValorUnitario",
        query = "SELECT m FROM Movimento m WHERE m.valorUnitario = "
```

```
                + ":valorUnitario")})
public class Movimento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idMovimento")
    private Integer idMovimento;
    @Basic(optional = false)
    @Column(name = "quantidade")
    private int quantidade;
    @Basic(optional = false)
    @Column(name = "tipo")
    private Character tipo;
    @Basic(optional = false)
    @Column(name = "valorUnitario")
    private long valorUnitario;
    @JoinColumn(name = "Pessoa_idPessoa", referencedColumnName = "idPessoa")
    @ManyToOne(optional = false)
    private Pessoa pessoaidPessoa;
    @JoinColumn(name = "Produto_idProduto", referencedColumnName = "idProduto")
    @ManyToOne(optional = false)
    private Produto produtoidProduto;
    @JoinColumn(name = "Usuario_idUsuario", referencedColumnName = "idUsuario")
    @ManyToOne(optional = false)
    private Usuario usuarioidUsuario;
```

```
    public Movimento() {
    }

    public Movimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }

    public Movimento(Integer idMovimento, int quantidade, Character tipo,
            long valorUnitario) {
        this.idMovimento = idMovimento;
        this.quantidade = quantidade;
        this.tipo = tipo;
        this.valorUnitario = valorUnitario;
    }

    // Getters e Setters
}
```

Model/Pessoa

```
package cadastroee.model;

import java.io.Serializable;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "Pessoa")
@NamedQueries({
    @NamedQuery(name = "Pessoa.findAll", query = "SELECT p FROM Pessoa p"),
    @NamedQuery(name = "Pessoa.findByIdPessoa", query = "SELECT p FROM Pessoa p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "Pessoa.findByNome", query = "SELECT p FROM Pessoa p WHERE p.nome = :nome"),
    @NamedQuery(name = "Pessoa.findByLogradouro", query = "SELECT p FROM Pessoa p WHERE p.logradouro = :logradouro")
    @NamedQuery(name = "Pessoa.findByCidade", query = "SELECT p FROM Pessoa p WHERE p.cidade = :cidade"),
    @NamedQuery(name = "Pessoa.findByEstado", query = "SELECT p FROM Pessoa p WHERE p.estado = :estado"),
    @NamedQuery(name = "Pessoa.findByTelefone", query = "SELECT p FROM Pessoa p WHERE p.telefone = :telefone"),
    @NamedQuery(name = "Pessoa.findByEmail", query = "SELECT p FROM Pessoa p WHERE p.email = :email")})
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;

    @Basic(optional = false)
    @Column(name = "nome")
    private String nome;

    @Column(name = "logradouro")
    private String logradouro;

    @Column(name = "cidade")
    private String cidade;

    @Column(name = "estado")
    private String estado;

    @Column(name = "telefone")
    private String telefone;
```

```
    @Column(name = "email")
    private String email;

    @OneToOne(mappedBy = "pessoa", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    private PessoaJuridica pessoaJuridica;

    @OneToOne(mappedBy = "pessoa", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    private PessoaFisica pessoaFisica;

    @OneToMany(mappedBy = "pessoaidPessoa", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    private List<Movimento> movimentoList;

    // Construtores, getters e setters foram mantidos
}
```

Model/PessoaFisica

```
package cadastroee.model;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "PessoaFisica")
@NamedQueries({
    @NamedQuery(name = "PessoaFisica.findAll",
            query = "SELECT p FROM PessoaFisica p"),
    @NamedQuery(name = "PessoaFisica.findByIdPessoa",
            query = "SELECT p FROM PessoaFisica p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "PessoaFisica.findByCpf",
            query = "SELECT p FROM PessoaFisica p WHERE p.cpf = :cpf")})
public class PessoaFisica implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
```

```
    private Integer idPessoa;

    @Basic(optional = false)
    @Column(name = "CPF")
    private String cpf;

    @OneToOne(mappedBy = "pessoaFisica")
    private Pessoa pessoa;

    public PessoaFisica() {
    }

    public PessoaFisica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public PessoaFisica(Integer idPessoa, String cpf) {
        this.idPessoa = idPessoa;
        this.cpf = cpf;
    }

    // Getters e Setters
}
```

Model/PessoaJuridica

```
package cadastroee.model;

import java.io.Serializable;
import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;

@Entity
@Table(name = "PessoaJuridica")
@NamedQueries({
    @NamedQuery(name = "PessoaJuridica.findAll",
            query = "SELECT p FROM PessoaJuridica p"),
    @NamedQuery(name = "PessoaJuridica.findByIdPessoa",
            query = "SELECT p FROM PessoaJuridica p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "PessoaJuridica.findByCnpj",
            query = "SELECT p FROM PessoaJuridica p WHERE p.cnpj = :cnpj")})
public class PessoaJuridica implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
```

```
    private Integer idPessoa;

    @Basic(optional = false)
    @Column(name = "CNPJ")
    private String cnpj;

    @OneToOne(mappedBy = "pessoaJuridica")
    private Pessoa pessoa;

    public PessoaJuridica() {
    }

    public PessoaJuridica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public PessoaJuridica(Integer idPessoa, String cnpj) {
        this.idPessoa = idPessoa;
        this.cnpj = cnpj;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }
```

```
    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof PessoaJuridica)) {
            return false;
```

```
        }
        PessoaJuridica other = (PessoaJuridica) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
            (this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.PessoaJuridica[ idPessoa=" + idPessoa + " ]";
    }

}
```

Model/Produto

```
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;

@Entity
@Table(name = "Produto")
@NamedQueries({
    @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM Produto p"),
    @NamedQuery(name = "Produto.findByIdProduto",
            query = "SELECT p FROM Produto p WHERE p.idProduto = :idProduto"),
    @NamedQuery(name = "Produto.findByNome",
            query = "SELECT p FROM Produto p WHERE p.nome = :nome"),
    @NamedQuery(name = "Produto.findByQuantidade",
            query = "SELECT p FROM Produto p WHERE p.quantidade = :quantidade"),
    @NamedQuery(name = "Produto.findByPrecoVenda",
```

```
            query = "SELECT p FROM Produto p WHERE p.precoVenda = :precoVenda")})
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;

    @Basic(optional = false)
    @Column(name = "nome")
    private String nome;

    @Basic(optional = false)
    @Column(name = "quantidade")
    private int quantidade;

    @Basic(optional = false)
    @Column(name = "precoVenda")
    private float precoVenda;

    @OneToMany(mappedBy = "produtoidProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }
```

```
    public Produto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Produto(Integer idProduto, String nome, int quantidade, float precoVenda) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.quantidade = quantidade;
        this.precoVenda = precoVenda;
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
```

```
    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public float getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(float precoVenda) {
        this.precoVenda = precoVenda;
    }

    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProduto != null ? idProduto.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Produto)) {
            return false;
        }
        Produto other = (Produto) object;
        if ((this.idProduto == null && other.idProduto != null) ||
            (this.idProduto != null && !this.idProduto.equals(other.idProduto))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Produto[ idProduto=" + idProduto + " ]";
    }

}
```

Model/Usuario

```
package cadastroee.model;

import java.io.Serializable;
import java.util.Collection;
import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;

@Entity
@Table(name = "Usuario")
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll",
            query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findByIdUsuario",
            query = "SELECT u FROM Usuario u WHERE u.idUsuario = :idUsuario"),
    @NamedQuery(name = "Usuario.findByLogin",
            query = "SELECT u FROM Usuario u WHERE u.login = :login"),
    @NamedQuery(name = "Usuario.findBySenha",
            query = "SELECT u FROM Usuario u WHERE u.senha = :senha")})
public class Usuario implements Serializable {
```

```
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idUsuario")
    private Integer idUsuario;

    @Column(name = "login")
    private String login;

    @Column(name = "senha")
    private String senha;

    @OneToMany(mappedBy = "usuarioidUsuario")
    private Collection<Movimento> movimentoCollection;

    public Usuario() {
    }

    public Usuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public Integer getIdUsuario() {
        return idUsuario;
    }
```

```
    public void setIdUsuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }
```

```
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idUsuario != null ? idUsuario.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Usuario)) {
            return false;
        }
        Usuario other = (Usuario) object;
        if ((this.idUsuario == null && other.idUsuario != null) ||
         (this.idUsuario != null && !this.idUsuario.equals(other.idUsuario))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "cadastroee.model.Usuario[ idUsuario=" + idUsuario + " ]";
    }

}
```

Controller/AbstractFaca

```
package cadastroee.controller;

import java.util.List;
import javax.persistence.EntityManager;

public abstract class AbstractFacade<T> {

    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    }

    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    public T find(Object id) {
```

```
    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    public List<T> findAll() {
        javax.persistence.criteria.CriteriaQuery<T> cq = getEntityManager()
                .getCriteriaBuilder().createQuery(entityClass);
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    public List<T> findRange(int[] range) {
        javax.persistence.criteria.CriteriaQuery<T> cq = getEntityManager()
                .getCriteriaBuilder().createQuery(entityClass);
        cq.select(cq.from(entityClass));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    public int count() {
        javax.persistence.criteria.CriteriaQuery<Long> cq = getEntityManager()
                .getCriteriaBuilder().createQuery(Long.class);
```

```
        javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

Controller/MovimentoFacade

```
package cadastroee.controller;

import cadastroee.model.Movimento;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;


@Stateless
public class MovimentoFacade extends AbstractFacade<Movimento> implements MovimentoFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public MovimentoFacade() {
        super(Movimento.class);
    }
}
```

Controller/MovimentoFacadeLocal

```
package cadastroee.controller;

import cadastroee.model.Movimento;
import java.util.List;
import javax.ejb.Local;


@Local
public interface MovimentoFacadeLocal {

    void create(Movimento movimento);

    void edit(Movimento movimento);

    void remove(Movimento movimento);

    Movimento find(Object id);

    List<Movimento> findAll();

    List<Movimento> findRange(int[] range);

    int count();

}
```

Controller/PessoaFacade

```
package cadastroee.controller;

import cadastroee.model.Pessoa;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;


@javax.ejb.Stateless
public class PessoaFacade extends AbstractFacade<Pessoa> implements PessoaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaFacade() {
        super(Pessoa.class);
    }

}
```

Controller/PessoaFacadeLocal

```
package cadastroee.controller;

import cadastroee.model.Pessoa;
import java.util.List;

@javax.ejb.Local
public interface PessoaFacadeLocal {

    void create(Pessoa pessoa);

    void edit(Pessoa pessoa);

    void remove(Pessoa pessoa);

    Pessoa find(Object id);

    List<Pessoa> findAll();

    List<Pessoa> findRange(int[] range);

    int count();

}
```

Controller/PessoaFisicaFacadeLocal

```
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import java.util.List;

@javax.ejb.Local
public interface PessoaFisicaFacadeLocal {

    void create(PessoaFisica pessoaFisica);

    void edit(PessoaFisica pessoaFisica);

    void remove(PessoaFisica pessoaFisica);

    PessoaFisica find(Object id);

    List<PessoaFisica> findAll();

    List<PessoaFisica> findRange(int[] range);

    int count();

}
```

Controller/PessoaFisicaFacade

```
package cadastroee.controller;

import cadastroee.model.PessoaFisica;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@javax.ejb.Stateless
public class PessoaFisicaFacade extends AbstractFacade<PessoaFisica> implements PessoaFisicaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaFisicaFacade() {
        super(PessoaFisica.class);
    }

}
```

Controller/PessoaFisicaFacade

```
package cadastroee.controller;

import cadastroee.model.PessoaJuridica;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

@jakarta.ejb.Stateless
public class PessoaJuridicaFacade extends AbstractFacade<PessoaJuridica> implements PessoaJuridicaFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public PessoaJuridicaFacade() {
        super(PessoaJuridica.class);
    }

}
```

Controller/PessoaJuridicaFacadeLocal

```
package cadastroee.controller;

import cadastroee.model.PessoaJuridica;
import java.util.List;

@jakarta.ejb.Local
public interface PessoaJuridicaFacadeLocal {

    void create(PessoaJuridica pessoaJuridica);

    void edit(PessoaJuridica pessoaJuridica);

    void remove(PessoaJuridica pessoaJuridica);

    PessoaJuridica find(Object id);

    List<PessoaJuridica> findAll();

    List<PessoaJuridica> findRange(int[] range);

    int count();

}
```

Controller/ProdutoFacade

```
package cadastroee.controller;

import cadastroee.model.Produto;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

@jakarta.ejb.Stateless
public class ProdutoFacade extends AbstractFacade<Produto> implements ProdutoFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public ProdutoFacade() {
        super(Produto.class);
    }

}
```

Controller/ProdutoFacadeLocal

```
package cadastroee.controller;

import cadastroee.model.Produto;
import java.util.List;

@jakarta.ejb.Local
public interface ProdutoFacadeLocal {

    void create(Produto produto);

    void edit(Produto produto);

    void remove(Produto produto);

    Produto find(Object id);

    List<Produto> findAll();

    List<Produto> findRange(int[] range);

    int count();

}
```

Controller/UsuarioFacade

```
package cadastroee.controller;

import cadastroee.model.Usuario;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;

@jakarta.ejb.Stateless
public class UsuarioFacade extends AbstractFacade<Usuario> implements UsuarioFacadeLocal {

    @PersistenceContext(unitName = "CadastroEE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public UsuarioFacade() {
        super(Usuario.class);
    }

}
```

Controller/UsuarioFacadeLocal

```
package cadastroee.controller;

import cadastroee.model.Usuario;
import java.util.List;

@jakarta.ejb.Local
public interface UsuarioFacadeLocal {

    void create(Usuario usuario);

    void edit(Usuario usuario);

    void remove(Usuario usuario);

    Usuario find(Object id);

    List<Usuario> findAll();

    List<Usuario> findRange(int[] range);

    int count();

}
```

Análise e Conclusão:

A. Como é organizado um projeto corporativo no NetBeans?

No NetBeans, um projeto corporativo geralmente é organizado em módulos, cada um correspondendo a uma parte específica do aplicativo. Por exemplo, pode haver módulos para a camada de apresentação (como o front-end da aplicação web), para a lógica de negócios (camada de serviço) e para o acesso a dados (camada de persistência).
Dentro de cada módulo, você pode encontrar diferentes pacotes para diferentes partes da aplicação, como controllers, models, views, etc.
O NetBeans oferece suporte para a criação e gerenciamento de vários tipos de projetos corporativos, incluindo projetos Java EE, onde você pode integrar tecnologias como JPA, EJB, Servlets, entre outras.

B. Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?

JPA (Java Persistence API) é uma especificação do Java EE para mapeamento objeto-relacional em aplicações Java. Ela permite que você mapeie objetos Java para tabelas em um banco de dados relacional de forma transparente e eficiente.
EJB (Enterprise JavaBeans) é uma especificação do Java EE para componentes de negócio em aplicações Java. EJBs fornecem serviços como transações, segurança e gerenciamento de ciclo de vida para objetos de negócio distribuídos.
Juntas, JPA e EJBs fornecem uma abordagem eficaz para desenvolver aplicações corporativas robustas e escaláveis. JPA lida com a persistência de dados, enquanto os EJBs fornecem a lógica de negócios.

C. Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?

O NetBeans oferece suporte completo para JPA e EJB, fornecendo ferramentas e assistentes que ajudam a criar e gerenciar entidades, sessões e beans de mensagens.
Ele fornece uma interface gráfica para configurar as conexões de banco de dados, mapeamento de entidades e outras configurações relacionadas ao JPA.
No caso dos EJBs, o NetBeans oferece assistentes para criar beans de sessão e beans de entidade, além de simplificar a configuração de transações, segurança e outros aspectos relacionados aos EJBs.

D. O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?

Servlets são componentes Java que estendem a funcionalidade de servidores web para gerar conteúdo dinâmico. Eles são uma parte fundamental do desenvolvimento web Java.
No NetBeans, você pode criar Servlets facilmente usando assistentes e modelos disponíveis na IDE.
A IDE oferece suporte para escrever, depurar e implantar Servlets, facilitando o desenvolvimento de aplicativos web baseados em Java.

E. Como é feita a comunicação entre os Serlvets e os Session Beans do pool de EJBs?

Os Servlets podem interagir com os Session Beans do pool de EJBs por meio da injeção de dependência ou da pesquisa do JNDI (Java Naming and Directory Interface).
A injeção de dependência é uma técnica em que os EJBs são injetados nos Servlets usando anotações como @EJB.
A pesquisa JNDI permite que os Servlets localizem e obtenham referências para os EJBs no contêiner EJB. O NetBeans oferece suporte para configurar e realizar essa pesquisa de forma eficiente.