Universidade Estácio de Sá

Polo Bangu - Rio de Janeiro

- **Curso:** Desenvolvimento Full-Stack
- **Disciplina:** Porque não paralelizar?
- **Semestre:** 3
- **Turma:** 2024.2
- **Aluna:** Clara Martins Azevedo

---

1. Missão Prática - Nível 5

2. Objetivos da Missão Prática:

• Criar servidores Java com base em Sockets.

• Criar clientes síncronos para servidores com base em Sockets.

• Criar clientes assíncronos para servidores com base em Sockets.

• Utilizar Threads para implementação de processos paralelos.

• No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para

implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

3. Códigos do Projeto:

- Movimento.java:

```
package model;

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

/**
 *
 * @author okidata
 */
@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
    @Basic(optional = false)
    @Column(name = "idMovimento")
    private Integer idMovimento;
    @Column(name = "quantidade")
    private Integer quantidade;
    @Column(name = "tipo")
    private String tipo;
    @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 Integer getIdMovimento() {
    return idMovimento;
}

public void setIdMovimento(Integer idMovimento) {
    this.idMovimento = idMovimento;
}

public Integer getQuantidade() {
    return quantidade;
}

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

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public Long getValorUnitario() {
    return valorUnitario;
}

public void setValorUnitario(Long valorUnitario) {
    this.valorUnitario = valorUnitario;
}

public Pessoa getPessoaidPessoa() {
    return pessoaidPessoa;
}

public void setPessoaidPessoa(Pessoa pessoaidPessoa) {
    this.pessoaidPessoa = pessoaidPessoa;
}

public Produto getProdutoidProduto() {
    return produtoidProduto;
}

public void setProdutoidProduto(Produto produtoidProduto) {
    this.produtoidProduto = produtoidProduto;
}

public Usuario getUsuarioidUsuario() {
    return usuarioidUsuario;
}

public void setUsuarioidUsuario(Usuario usuarioidUsuario) {
    this.usuarioidUsuario = usuarioidUsuario;
}
```

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

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Movimento)) {
            return false;
        }
        Movimento other = (Movimento) object;
        if ((this.idMovimento == null && other.idMovimento != null) || (this.idMovimento != null &&
!this.idMovimento.equals(other.idMovimento))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.Movimento[ idMovimento=" + idMovimento + " ]";
    }

}
```

- Pessoa.java:

```
package model;

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

/**
 *
 * @author okidata
 */
@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.findByLocadouro", query = "SELECT p FROM Pessoa p WHERE p.locadouro =
:locadouro"),
    @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
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Column(name = "nome")
    private String nome;
    @Column(name = "locadouro")
    private String locadouro;
    @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(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaJuridica pessoaJuridica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaFisica pessoaFisica;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "pessoaidPessoa")
    private Collection<Movimento> movimentoCollection;

    public Pessoa() {
    }

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

    public Integer getIdPessoa() {
        return idPessoa;
    }

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

    public String getNome() {
        return nome;
    }

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

    public String getLocadouro() {
        return locadouro;
    }

    public void setLocadouro(String locadouro) {
        this.locadouro = locadouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }
```

```
    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public PessoaJuridica getPessoaJuridica() {
        return pessoaJuridica;
    }

    public void setPessoaJuridica(PessoaJuridica pessoaJuridica) {
        this.pessoaJuridica = pessoaJuridica;
    }

    public PessoaFisica getPessoaFisica() {
        return pessoaFisica;
    }

    public void setPessoaFisica(PessoaFisica pessoaFisica) {
        this.pessoaFisica = pessoaFisica;
    }

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

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

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

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Pessoa)) {
            return false;
        }
        Pessoa other = (Pessoa) 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 "model.Pessoa[ idPessoa=" + idPessoa + " ]";
    }

}
```

• PessoaFisica.java:

```
package model;

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

/**
 *
 * @author okidata
 */
@Entity
@Table(name = "PessoaFisica")
@NamedQueries({
    @NamedQuery(name = "PessoaFisica.findAll", query = "SELECT p FROM PessoaFisica p"),
    @NamedQuery(name = "PessoaFisica.findByIdPessoaFisica", query = "SELECT p FROM PessoaFisica p
WHERE p.idPessoaFisica = :idPessoaFisica"),
    @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 = "idPessoaFisica")
    private Integer idPessoaFisica;
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "idPessoaFisica", referencedColumnName = "idPessoa", insertable = false, updatable =
false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaFisica() {
    }

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

    public Integer getIdPessoaFisica() {
        return idPessoaFisica;
    }

    public void setIdPessoaFisica(Integer idPessoaFisica) {
        this.idPessoaFisica = idPessoaFisica;
    }
```

```
    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

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

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

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof PessoaFisica)) {
            return false;
        }
        PessoaFisica other = (PessoaFisica) object;
        if ((this.idPessoaFisica == null && other.idPessoaFisica != null) || (this.idPessoaFisica != null &&
!this.idPessoaFisica.equals(other.idPessoaFisica))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.PessoaFisica[ idPessoaFisica=" + idPessoaFisica + " ]";
    }

}
```

• PessoaJuridica.java:

```
package model;

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

/**
 *
 * @author okidata
 */
```

```
@Entity
@Table(name = "PessoaJuridica")
@NamedQueries({
    @NamedQuery(name = "PessoaJuridica.findAll", query = "SELECT p FROM PessoaJuridica p"),
    @NamedQuery(name = "PessoaJuridica.findByIdPessoaJuridica", query = "SELECT p FROM PessoaJuridica p
WHERE p.idPessoaJuridica = :idPessoaJuridica"),
    @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 = "idPessoaJuridica")
    private Integer idPessoaJuridica;
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "idPessoaJuridica", referencedColumnName = "idPessoa", insertable = false, updatable =
false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaJuridica() {
    }

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

    public Integer getIdPessoaJuridica() {
        return idPessoaJuridica;
    }

    public void setIdPessoaJuridica(Integer idPessoaJuridica) {
        this.idPessoaJuridica = idPessoaJuridica;
    }

    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 += (idPessoaJuridica != null ? idPessoaJuridica.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof PessoaJuridica)) {
            return false;
        }
        PessoaJuridica other = (PessoaJuridica) object;
```

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

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

}
```

• Produto.java:

```
package model;

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

/**
 *
 * @author okidata
 */
@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
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;
    @Column(name = "nome")
    private String nome;
    @Column(name = "quantidade")
    private Integer quantidade;
    @Column(name = "precoVenda")
    private Long precoVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "produtoidProduto")
    private Collection<Movimento> movimentoCollection;
```

```
    public Produto() {
    }

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

    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 Integer getQuantidade() {
        return quantidade;
    }

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

    public Long getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(Long 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) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        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;
    }
```

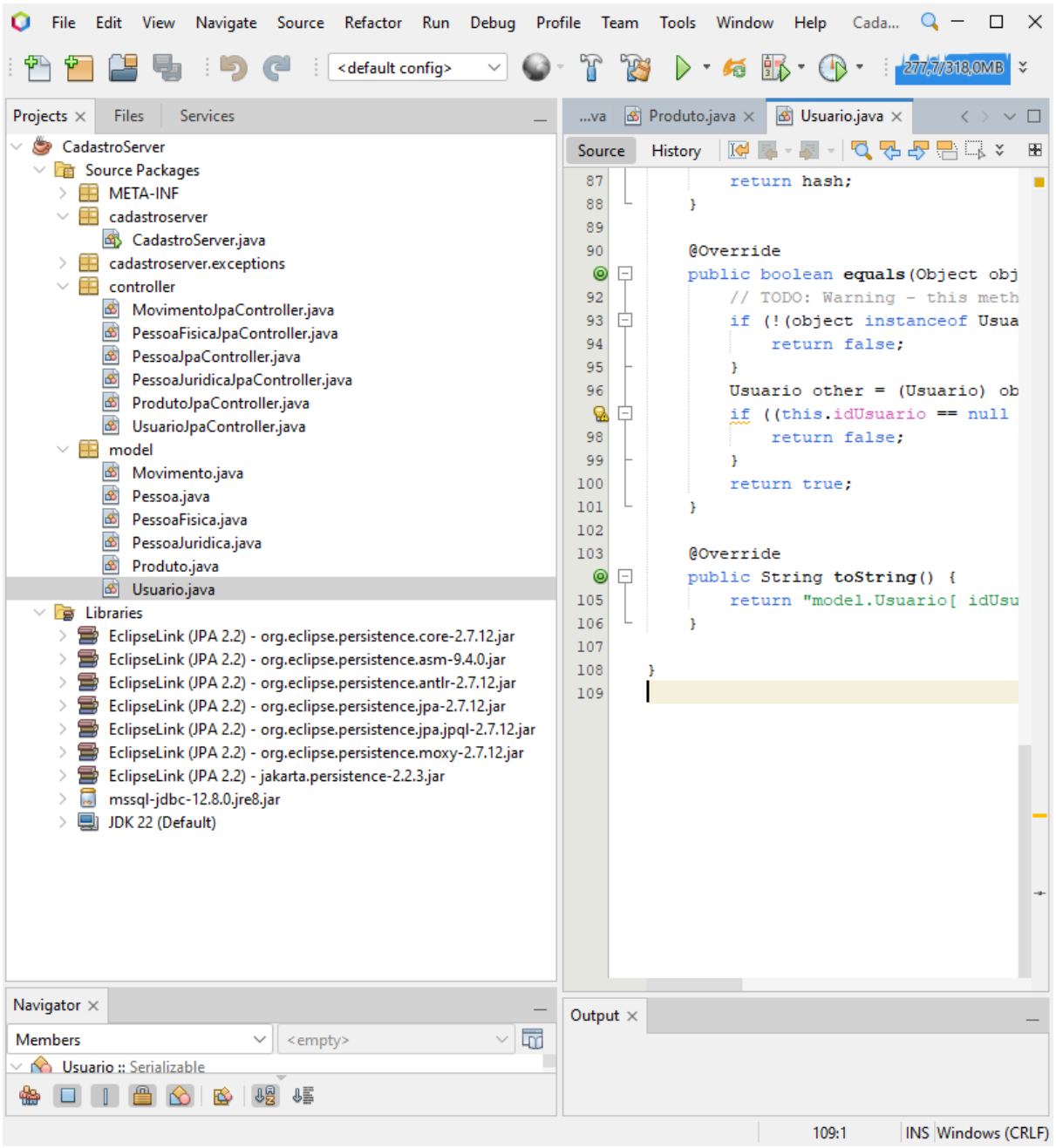
```
    @Override
    public String toString() {
        return "model.Produto[ idProduto=" + idProduto + " ]";
    }

}
```

• Usuario.java:

```
package model;

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

/**
 *
 * @author okidata
 */
@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.findByLoginUsuario", query = "SELECT u FROM Usuario u WHERE
u.loginUsuario = :loginUsuario"),
    @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
    @Basic(optional = false)
    @Column(name = "idUsuario")
    private Integer idUsuario;
    @Column(name = "loginUsuario")
    private String loginUsuario;
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, 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 getLoginUsuario() {
        return loginUsuario;
    }

    public void setLoginUsuario(String loginUsuario) {
        this.loginUsuario = loginUsuario;
    }

    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) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        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 "model.Usuario[ idUsuario=" + idUsuario + " ]";
    }

}
```

4. Resultado da execução do código:



5. Análise e Conclusão:

a. *Como funcionam as classes Socket e ServerSocket?*

São usados para estabelecer conexões cliente-servidor. Enquanto Socket cria o ponto de conexão para comunicação entre cliente e servidor, ServerSocket funciona como ponto de escuta para do cliente para o servidor.

b. *Qual a importância das portas para a conexão com servidores?*

Sem as portas viabiliza comunicações entre redes, sem elas isso não seria possível já que, como o próprio nome diz, elas são a porta de acesso para a comunicação servidor-cliente.

c. *Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?*

São usados para inserir e recuperar objetos serializados do stream. Eles devem ser serializados pois assim eles são convertidos em um formato para serem transportados pela rede, sendo convertidos novamente para o formato original quando recebidos pelo receptor.

d. *Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?*

Pois o cliente não interage diretamente com o banco, afinal devido a arquitetura do projeto ele apenas interagem com uma representação dos dados, garantindo a segurança do banco.

1. Missão Prática - Nível 5

2. Objetivos da Missão Prática:

• Criar servidores Java com base em Sockets.

• Criar clientes síncronos para servidores com base em Sockets.

• Criar clientes assíncronos para servidores com base em Sockets.

• Utilizar Threads para implementação de processos paralelos.

• No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

3. Códigos do Projeto:

4. Resultado da execução do código:

5. Análise e Conclusão:

a. *Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?*

b. *Para que serve o método invokeLater, da classe SwingUtilities?*

Serve para assegurar que as mudanças realizadas na interface sejam feitas de forma segura.

c. *Como os objetos são enviados e recebidos pelo Socket Java?*

Através da utilização das classes “ObjectOutputStream” e “ObjectInputStream”.

d. *Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.*