

True positives	
True negatives	
False positives	
False negatives	

```
JBook - Shadowing
Source folder: "src/main/java"
Main package: "br.com.infowaypi.jbook."
=====
Actual package: "autenticacao"
=====
-----
Java file: Autenticador.java
-----
package br.com.infowaypi.jbook.autenticacao;

public class Autenticador {

    private static final ThreadLocal<UsuarioInterface> threadUsuario = new ThreadLocal<UsuarioInterface>();

    public Autenticador() {
    }

    public UsuarioInterface getUsuario() {
        return threadUsuario.get();
    }

    public String[] getRoles(String login, String senha) throws Exception {
        if (Utils.isStringVazia(senha)) {
            return null;
        }
        UsuarioInterface usuario = buscaUsuario(login);
        if (usuario != null && usuario.autentica(senha)) {
            threadUsuario.set(usuario);
            return new String[] { usuario.getRole() };
        }
        return null;
    }

    public Usuario buscaUsuario(String login) {
        SearchAgent sa = new SearchAgent();
        sa.addParameter(new Equals("login", login));
        return sa.uniqueResult(Usuario.class);
    }
}
-----
Java file: UsuarioInterface.java
-----
package br.com.infowaypi.jbook.autenticacao;

public interface UsuarioInterface extends Serializable, Comparable<UsuarioInterface> {

    public static final String ATIVO = "A";
    public static final String CANCELADO = "C";

    public abstract Boolean validate() throws ValidateException;

    public abstract Long getIdUsuario();

    public abstract void setIdUsuario(Long idUsuario);

    public abstract String getNome();

    public abstract void setNome(String nome);

    public abstract String getEmail();

    public abstract void setEmail(String email);

    public abstract String getLogin();

    public abstract void setLogin(String login);

    public abstract String getSenha();

    public abstract void setSenha(String senha);

    public abstract String getStatus();
```

```
public abstract void setStatus(String status);

public String getNovaSenhaConfirmacao();

public abstract void setNovaSenhaConfirmacao(String novaSenhaConfirmacao);

public String getNovaSenhaDigitada();

public abstract void setNovaSenhaDigitada(String novaSenhaDigitada);

public abstract String getRole();

public abstract void setRole(String role);

public abstract boolean isPossuiRole(String... roles);

public abstract boolean autentica(String senha);

public void tocarObjetos();
}
=====
Actual package: "config"
=====
-----
Java file:  Config.java
-----
package br.com.infowaypi.jbook.config;

public class Config {

    public static String aplicacionPath = currentPath();

    public static final String sourcePath = aplicacionPath + File.separator + "src" + File.separator + "main" + File.separator;

    public static final String resoucesPath = sourcePath + "resources" + File.separator;

    public static final String webAppPath = sourcePath + "webapp" + File.separator;

    public static String getFileInResources(String file) {
        return resoucesPath + file;
    }

    public static String getFileInWebApp(String file) {
        return webAppPath + file;
    }

    private static String currentPath(){
        try{
            return new File("").getCanonicalPath();
        }catch (IOException e) {
            new IOException(e.getMessage());
        }
        return null;
    }
}
=====
Actual package: "core"
=====
-----
Java file:  Emprestimo.java
-----
package br.com.infowaypi.jbook.core;

public class Emprestimo implements Serializable, Comparable<Emprestimo> {

    private static final long serialVersionUID = -7427026885020906922L;

    private Long idEmprestimo;

    private Usuario bibliotecario;

    private Usuario leitor;

    private Exemplar exemplar;

    private Date dataSolicitacao;
```

```
private Date dataPrevisaoDeDevolucao;

private Date dataDevolucao;

private String situacao;

public Emprestimo() {
    this.dataSolicitacao = new Date();
}

public Long getIdEmprestimo() {
    return idEmprestimo;
}

public void setIdEmprestimo(Long idEmprestimo) {
    this.idEmprestimo = idEmprestimo;
}

public Usuario getBibliotecario() {
    return bibliotecario;
}

public void setBibliotecario(Usuario bibliotecario) {
    this.bibliotecario = bibliotecario;
}

public Usuario getLeitor() {
    return leitor;
}

public void setLeitor(Usuario leitor) {
    this.leitor = leitor;
}

public Exemplar getExemplar() {
    return exemplar;
}

public void setExemplar(Exemplar exemplar) {
    this.exemplar = exemplar;
}

public Date getDataSolicitacao() {
    return dataSolicitacao;
}

public void setDataSolicitacao(Date dataSolicitacao) {
    this.dataSolicitacao = dataSolicitacao;
}

public Date getDataPrevisaoDeDevolucao() {
    return dataPrevisaoDeDevolucao;
}

public void setDataPrevisaoDeDevolucao(Date dataPrevisaoDeDevolucao) {
    this.dataPrevisaoDeDevolucao = dataPrevisaoDeDevolucao;
}

public Date getDataDevolucao() {
    return dataDevolucao;
}

public void setDataDevolucao(Date dataDevolucao) {
    this.dataDevolucao = dataDevolucao;
}

public Boolean validate(UsuarioInterface bibliotecario) throws ValidateException {
    if ( getDataSolicitacao().compareTo(getDataPrevisaoDeDevolucao()) > 0)
        throw new ValidateException( "A data de previsão para devolução deve ser futura");
    if (EmprestimoManager.passouLimiteEmprestimos(this.getLeitor())) {
        throw new ValidateException( "O limite de 02 (duas) solicitações de empréstimo foi atingido. Não será possível realizar a solicitação de empréstimo.");
    }
    if (EmprestimoManager.solicitouMesmaPublicacao(this.getLeitor(),this.getExemplar())) {
        throw new ValidateException( "Não é possível solicitar empréstimo de uma mesma publicação já solicitada e em aberto.");
    }
    getExemplar().setSituacao(SituacaoExemplarEnum.EMPRESTADO.getValor());
    this.setBibliotecario((Usuario) bibliotecario);
    this.situacao = SituacaoEmprestimoEnum.CONFIRMADO.getValor();
    return true;
}
```

```

public int hashCode() {
    return new HashCodeBuilder().append(getIdEmprestimo()).hashCode();
}

public boolean equals(Object obj) {
    if (!(obj instanceof Emprestimo)) {
        return false;
    }
    Emprestimo emprestimo = (Emprestimo) obj;
    return new EqualsBuilder().append(this.getIdEmprestimo(), emprestimo.getIdEmprestimo()).isEquals();
}

public int compareTo(Emprestimo outro) {
    return this.getDataSolicitacao().compareTo(outro.getDataSolicitacao());
}

public String getSituacao() {
    return situacao;
}

public void setSituacao(String situacao) {
    this.situacao = situacao;
}

public void cancelarSolicitacao() {
    getExemplar().setSituacao(SituacaoExemplarEnum.DISPONIVEL.getValor());
    setSituacao(SituacaoEmprestimoEnum.CANCELADO.getValor());
    Transaction trans = HibernateUtil.currentSession().beginTransaction();
    HibernateUtil.currentSession().saveOrUpdate(this);
    trans.commit();
}

public void confirmarSolicitacao() {
    getExemplar().setSituacao(SituacaoExemplarEnum.EMPRESTADO.getValor());
    setSituacao(SituacaoEmprestimoEnum.CONFIRMADO.getValor());
    Transaction trans = HibernateUtil.currentSession().beginTransaction();
    HibernateUtil.currentSession().saveOrUpdate(this);
    trans.commit();
}

public void encerrar() throws ValidateException {
    getExemplar().setSituacao(SituacaoExemplarEnum.DISPONIVEL.getValor());
    setSituacao(SituacaoEmprestimoEnum.FINALIZADO.getValor());
    setDataDevolucao(new Date());
    Transaction trans = HibernateUtil.currentSession().beginTransaction();
    HibernateUtil.currentSession().saveOrUpdate(this);
    trans.commit();
    if (getDataDevolucao().compareTo(getDataPrevisaoDeDevolucao()) > 0) {
        throw new ValidateException("Devolução em atraso");
    }
}
}

```

Java file: EtiquetaTombo.java

```
package br.com.infowaypi.jbook.core;
```

```

public class EtiquetaTombo {

    private Long idEtiquetaTombo = 1L;

    private Long ultimoTombo;

    public Long getIdEtiquetaTombo() {
        return idEtiquetaTombo;
    }

    public void setIdEtiquetaTombo(Long idEtiquetaTombo) {
        this.idEtiquetaTombo = idEtiquetaTombo;
    }

    public Long getUltimoTombo() {
        return ultimoTombo;
    }

    public void setUltimoTombo(Long ultimoTombo) {
        this.ultimoTombo = ultimoTombo;
    }
}

```

```
public int hashCode() {
    return new HashCodeBuilder().append(getIdEtiquetaTombo()).hashCode();
}

public boolean equals(Object obj) {
    if (!(obj instanceof EtiquetaTombo)) {
        return false;
    }
    EtiquetaTombo etiqueta = (EtiquetaTombo) obj;
    return new EqualsBuilder().append(this.getIdEtiquetaTombo(), etiqueta.getIdEtiquetaTombo()).isEquals();
}
}
```

Java file: Exemplar.java

```
package br.com.infowaypi.jbook.core;
```

```
public class Exemplar implements Serializable, Comparable<Exemplar> {
```

```
    private static final long serialVersionUID = 4448872540560235275L;
```

```
    public Exemplar() {
        this.dataCatalogacao = new Date();
        this.situacao = SituacaoExemplarEnum.DISPONIVEL.getValor();
        this.emprestimos = new ArrayList<Emprestimo>();
    }
```

```
    private Long idExemplar;
```

```
    private Publicacao publicacao;
```

```
    private Long tombo;
```

```
    private Date dataCatalogacao;
```

```
    private List<Emprestimo> emprestimos;
```

```
    private String estadoDeConservacao;
```

```
    private String situacao;
```

```
    public Long getIdExemplar() {
        return idExemplar;
    }
```

```
    public void setIdExemplar(Long idExemplar) {
        this.idExemplar = idExemplar;
    }
```

```
    public Publicacao getPublicacao() {
        return publicacao;
    }
```

```
    public void setPublicacao(Publicacao publicacao) {
        this.publicacao = publicacao;
    }
```

```
    public Long getTombo() {
        return tombo;
    }
```

```
    public void setTombo(Long tombo) {
        this.tombo = tombo;
    }
```

```
    public Date getDataCatalogacao() {
        return dataCatalogacao;
    }
```

```
    public void setDataCatalogacao(Date dataCatalogacao) {
        this.dataCatalogacao = dataCatalogacao;
    }
```

```
    public List<Emprestimo> getEmprestimos() {
        return emprestimos;
    }
```

```
    public void setEmprestimos(List<Emprestimo> emprestimos) {
        this.emprestimos = emprestimos;
    }
```

```
public String getEstadoDeConservacao() {
    return estadoDeConservacao;
}

public void setEstadoDeConservacao(String estadoDeConservacao) {
    this.estadoDeConservacao = estadoDeConservacao;
}

public String getSituacao() {
    return situacao;
}

public void setSituacao(String situacao) {
    this.situacao = situacao;
}

public Usuario getUltimoLeitorAssociado() {
    return getEmprestimos().get(getEmprestimos().size() - 1).getLeitor();
}

public Emprestimo getUltimoEmprestimoAssociado() {
    return getEmprestimos().get(getEmprestimos().size() - 1);
}

public String getTituloTombo(){
    return publicacao.getTitulo() + " - " + getTombo();
}

public Boolean validate() throws Exception {
    SearchAgent sa = new SearchAgent();
    sa.addParameter(new Equals("tombo", getTombo()));
    Exemplar exemplar = sa.uniqueResult(Exemplar.class);
    if (exemplar != null && !this.equals(exemplar)) {
        throw new ValidateException("Tombo já cadastrado!");
    }
    return true;
}

public boolean equals(Object obj) {
    if (!(obj instanceof Exemplar)) {
        return false;
    }
    Exemplar exemplar = (Exemplar) obj;
    return new EqualsBuilder().append(this.getIdExemplar(), exemplar.getIdExemplar()).append(this.getTombo(), exemplar.getTombo()).isEquals();
}

public int hashCode() {
    return new HashCodeBuilder().append(this.getTombo()).toHashCode();
}

public int compareTo(Exemplar outro) {
    return this.getPublicacao().getTitulo().compareTo(outro.getPublicacao().getTitulo());
}
}
```

Java file: Publicacao.java

```
package br.com.infowaypi.jbook.core;
```

```
public class Publicacao implements Serializable, Comparable<Publicacao> {

    private static final long serialVersionUID = 7770198638083066524L;

    public Publicacao() {}

    private Long idPublicacao;

    private String titulo;

    private String assunto;

    private String autor;

    private String editora;

    private Long ISBN;

    private String tipoDePublicacao;
```

```
private Set<Exemplar> exemplares;

public Long getIdPublicacao() {
    return idPublicacao;
}

public void setIdPublicacao(Long idPublicacao) {
    this.idPublicacao = idPublicacao;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getAssunto() {
    return assunto;
}

public void setAssunto(String assunto) {
    this.assunto = assunto;
}

public String getAutor() {
    return autor;
}

public void setAutor(String autor) {
    this.autor = autor;
}

public String getEditora() {
    return editora;
}

public void setEditora(String editora) {
    this.editora = editora;
}

public Long getISBN() {
    return ISBN;
}

public void setISBN(Long isbn) {
    ISBN = isbn;
}

public String getTipoDePublicacao() {
    return tipoDePublicacao;
}

public void setTipoDePublicacao(String tipoDePublicacao) {
    this.tipoDePublicacao = tipoDePublicacao;
}

public Set<Exemplar> getExemplares() {
    return exemplares;
}

public void setExemplares(Set<Exemplar> exemplares) {
    this.exemplares = exemplares;
}

public Boolean validate() throws ValidateException {
    boolean retorno = false;
    if (this.getTipoDePublicacao().equals(TipoPublicacaoEnum.LIVRO.getValor()) && this.getISBN() == null) {
        throw new ValidateException("O preenchimento do ISBN é obrigatório para livros.");
    } else if (this.getTipoDePublicacao().equals(TipoPublicacaoEnum.REVISTA.getValor()) && this.getISBN() != null) {
        throw new ValidateException("O preenchimento do ISBN não é necessário para revistas.");
    }
    retorno = verificaPreExistenciaPublicacao();
    return retorno;
}

private Boolean verificaPreExistenciaPublicacao() throws ValidateException {
    SearchAgent sa = new SearchAgent();
```

```
sa.addParameter(new Equals("titulo", this.getTitulo()));
sa.addParameter(new Equals("assunto", this.getAssunto()));
sa.addParameter(new Equals("autor", this.getAutor()));
sa.addParameter(new Equals("editora", getEditora()));
sa.addParameter(new Equals("tipoDePublicacao", this.tipoDePublicacao));
if (sa.uniqueResultAny(Publicacao.class) != null) {
    throw new ValidateException("Publicação já cadastrada!");
}
if (Utils.isCampoDuplicado(this, "ISBN", this.getISBN())) {
    throw new ValidateException("Não é permitido cadastrar duas publicações com o mesmo ISBN!");
}
return true;
}

public int hashCode() {
    return new HashCodeBuilder().append(this.idPublicacao).append(this.ISBN).toHashCode();
}

public boolean equals(Object obj) {
    if (!(obj instanceof Publicacao)) {
        return false;
    }
    Publicacao publicacao = (Publicacao) obj;
    return new EqualsBuilder().append(this.getIdPublicacao(), publicacao.getIdPublicacao()).append(this.getISBN(), publicacao.getISBN()).isEquals();
}

public int compareTo(Publicacao outro) {
    return this.getTitulo().compareTo(outro.getTitulo());
}
}
```

Java file: Usuario.java

```
package br.com.infowaypi.jbook.core;
```

```
public class Usuario implements UsuarioInterface {
```

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

```
    protected Long idUsuario;
```

```
    private String login;
```

```
    private String senha;
```

```
    private String novaSenhaDigitada;
```

```
    private String novaSenhaConfirmacao;
```

```
    private String role;
```

```
    private String nome;
```

```
    private String email;
```

```
    private String status;
```

```
    private Set<Emprestimo> emprestimos;
```

```
    public Usuario() {
        this.status = ATIVO;
    }
```

```
    public Boolean validate() throws ValidateException {
        if (Utils.isStringVazia(this.getLogin()))
            throw new ValidateException("O Login deve ser informado.");
        if (Utils.isStringVazia(this.getNome()))
            throw new ValidateException("O Nome do usuário deve ser informado.");
        if (Utils.isStringVazia(this.getEmail())){
            throw new ValidateException("O Email deve ser informado.");
        }
        if (Utils.isStringVazia(this.getRole()))
            throw new ValidateException("O role do usuário deve ser informado.");
        if (Utils.isCampoDuplicado(this, "login", this.getLogin()))
            throw new ValidateException("O login informado já existe. Escolha outro nome para o login e tente novamente.");
        if (Utils.isStringVazia(this.getSenha())) {
            verificarRestricoes();
        } else {
            if (Utils.isStringVazia(this.getNovaSenhaDigitada()) && Utils.isStringVazia(this.getNovaSenhaConfirmacao())){
                return true;
            }
        }
    }
```

```
    }
    verificarRestricoes();
}
this.setSenha(String.valueOf(this.getNovaSenhaDigitada().hashCode()));
return true;
}

private void verificarRestricoes() throws ValidateException {
    if (Utils.isStringVazia(this.getNovaSenhaDigitada()))
        throw new ValidateException("A senha deve ser informada.");
    if (Utils.isStringVazia(this.getNovaSenhaConfirmacao()))
        throw new ValidateException("A confirmação da senha deve ser informada.");
    if (!this.getNovaSenhaDigitada().equals(this.getNovaSenhaConfirmacao()))
        throw new ValidateException("Senhas não conferem.");
}

public boolean isPossuiRole(String... roles) {
    for (String role : roles) {
        if (this.getRole().equals(role))
            return true;
    }
    return false;
}

public boolean autentica(String senhaDigitada) {
    if (!StringUtils.isEmpty(senhaDigitada) && this.status.equals(ATIVO) && this.getSenha().equals(String.valueOf(senhaDigitada.hashCode())))
        return true;
    return false;
}

public Long getIdUsuario() {
    return idUsuario;
}

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

public String getRole() {
    return role;
}

public void setRole(String role) {
    this.role = role;
}

public String getSenha() {
    return senha;
}

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

public String getLogin() {
    return login;
}

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

public String getNome() {
    return nome;
}

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

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

public Set<Emprestimo> getEmprestimos() {
    return emprestimos;
}
```

```
}

public void setEmprestimos(Set<Emprestimo> emprestimos) {
    this.emprestimos = emprestimos;
}

public String getEmail() {
    return email;
}

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

public String getNovaSenhaConfirmacao() {
    return novaSenhaConfirmacao;
}

public void setNovaSenhaConfirmacao(String novaSenhaConfirmacao) {
    this.novaSenhaConfirmacao = novaSenhaConfirmacao;
}

public String getNovaSenhaDigitada() {
    return novaSenhaDigitada;
}

public void setNovaSenhaDigitada(String novaSenhaDigitada) {
    this.novaSenhaDigitada = novaSenhaDigitada;
}

public void tocarObjetos() {
    this.getIdUsuario();
    this.getNome();
    this.getRole();
}

public boolean equals(Object object) {
    if (!(object instanceof UsuarioInterface)) {
        return false;
    }
    Usuario usuario = (Usuario) object;
    UsuarioInterface user = (UsuarioInterface) object;
    return new EqualsBuilder()
        .append(this.getIdUsuario(), user.getIdUsuario())
        .append(this.getLogin(), usuario.getLogin()).isEquals();
}

public int hashCode() {
    return new HashCodeBuilder()
        .append(this.getIdUsuario())
        .append(this.getLogin()).toHashCode();
}

public String toString() {
    return new ToStringBuilder(this, ToStringStyle.DEFAULT_STYLE)
        .append("Login", this.login).append("nome", this.nome)
        .append("role", this.role).toString();
}

public int compareTo(UsuarioInterface outro) {
    Integer compareRole = this.getRole().compareTo(outro.getRole());
    Integer compareNome = this.getNome().compareTo(outro.getNome());
    if (!compareRole.equals(0))
        return compareRole;
    return compareNome;
}
}

=====
=====
Actual package: "datasource"
=====
-----
Java file:  DataSourceEtiquetaTombo.java
-----

package br.com.infowaypi.jbook.datasources;

public class DataSourceEtiquetaTombo {

    public DataSourceEtiquetaTombo(Long[] tombos) {
        this.tombos = tombos;
    }

    private Long[] tombos;

    public Long[] getTombos() {
        return tombos;
    }
}
```

```
}

public String getLogoInfoway() {
    return "/home/jbook/files/logoInfoway.png";
}
}
```

=====
=====
Actual package: "enumeration"
=====

Java file: EstadoConservacaoEnum.java

```
package br.com.infowaypi.jbook.enumeration;
```

```
public enum EstadoConservacaoEnum {

    NOVO      ("NOVO", "Novo"),
    EXCELENTE  ("EXCELENTE", "Excelente"),
    BOM        ("BOM", "Bom"),
    DEPRECIADO ("DEPRECIADO", "Depreciado"),
    INUTILIZAVEL ("INUTILIZAVEL", "Inutilizável");

    private String valor;

    private String descricao;

    public String getValor() {
        return valor;
    }

    public String getDescricao() {
        return descricao;
    }

    private EstadoConservacaoEnum(String valor, String descricao) {
        this.valor = valor;
        this.descricao = descricao;
    }

}
```

Java file: PeriodoSolicitacaoEnum.java

```
package br.com.infowaypi.jbook.enumeration;
```

```
public enum PeriodoSolicitacaoEmprestimoEnum {

    UMA      (1, "Uma Semana"),
    DUAS      (2, "Duas Semanas"),
    TRES      (3, "Três Semanas"),
    QUATRO    (4, "Quatro Semanas");

    private int valor;
    private String descricao;

    public int getValor() {
        return valor;
    }

    public String getDescricao() {
        return descricao;
    }

    private PeriodoSolicitacaoEmprestimoEnum(int valor, String descricao) {
        this.valor = valor;
        this.descricao = descricao;
    }

}
```

Java file: RolesEnum.java

```
package br.com.infowaypi.jbook.enumeration;
```

```
public enum RolesEnum {

    LEITOR      ("LEITOR", "Leitor"),
    BIBLIOTECARIO ("BIBLIOTECARIO", "Bibliotecário"),
    ROOT         ("ROOT", "Root");

}
```

```
private String valor;
private String descricao;

RolesEnum(String valor, String descricao) {
    this.valor = valor;
    this.descricao = descricao;
}

public String getValor() {
    return valor;
}

public String getDescricao() {
    return descricao;
}

}
```

Java file: SituacaoEmprestimoEnum.java

```
package br.com.infowaypi.jbook.enumeration;
```

```
public enum SituacaoEmprestimoEnum {

    SOLICITADO      ("SOLICITADO", "Solicitado"),
    CONFIRMADO      ("CONFIRMADO", "Confirmado"),
    CANCELADO       ("CANCELADO", "Cancelado"),
    EXPIRADO        ("EXPIRADO", "Expirado"),
    FINALIZADO      ("FINALIZADO", "Finalizado");
}
```

```
private String valor;

private String descricao;

public String getValor() {
    return valor;
}

public String getDescricao() {
    return descricao;
}
```

```
private SituacaoEmprestimoEnum(String valor, String descricao) {
    this.valor = valor;
    this.descricao = descricao;
}
```

Java file: SituacaoExemplarEnum.java

```
package br.com.infowaypi.jbook.enumeration;
```

```
public enum SituacaoExemplarEnum {

    DISPONIVEL("DISPONIVEL", "Disponível"),
    SOLICITADO("SOLICITADO", "Solicitado"),
    EMPRESTADO("EMPRESTADO", "Emprestado");

    private String valor;

    private String descricao;

    public String getValor() {
        return valor;
    }

    public String getDescricao() {
        return descricao;
    }

    private SituacaoExemplarEnum(String valor, String descricao) {
        this.valor = valor;
        this.descricao = descricao;
    }
}
```

Java file: TipoPublicacaoEnum.java

```
package br.com.infowaypi.jbook.enumeration;
```

```
public enum TipoPublicacaoEnum implements Serializable {

    LIVRO      ("LIVRO", "Livro"),
    REVISTA     ("REVISTA", "Revista");

    private TipoPublicacaoEnum(String valor, String descricao) {
        this.valor = valor;
        this.descricao = descricao;
    }

    private String valor;

    private String descricao;

    public String getValor() {
        return valor;
    }

    public String getDescricao() {
        return descricao;
    }
}
```

```
=====
=====
Actual package: "flow"
=====
-----
Java file:  AlterarSenhaFlow.java
-----
```

```
package br.com.infowaypi.jbook.flow;

public class AlterarSenhaFlow {

    public UsuarioInterface alteraSenha(UsuarioInterface usuario, String senhaAntiga, String senhaNova, String senhaConfirmacao) throws Exception {
        HibernateUtil.currentSession().evict(usuario);
        Usuario user = (Usuario) ImplDAO.findById(usuario.getIdUsuario(), Usuario.class);
        if(user == null){
            throw new ValidateException("Usuário nulo.");
        }
        boolean isSenhaNovaVazia = Utils.isStringVazia(senhaNova);
        boolean isSenhaConfirmacaoVazia = Utils.isStringVazia(senhaConfirmacao);
        boolean isSenhaAntigaVazia = Utils.isStringVazia(senhaAntiga);
        boolean isCamposSenhasVazios = isSenhaAntigaVazia && isSenhaConfirmacaoVazia && isSenhaNovaVazia;
        if (!isCamposSenhasVazios) {
            if(Utils.isStringVazia(senhaAntiga)){
                throw new ValidateException("A senha atual deve ser informada.");
            }
            if(isSenhaNovaVazia){
                throw new ValidateException("A nova senha deve ser informada.");
            }
            if(isSenhaConfirmacaoVazia){
                throw new ValidateException("A confirmação da nova senha deve ser informada.");
            }
        }
        boolean isSenhasNaoConferem = !usuario.getSenha().equals(String.valueOf(senhaAntiga.hashCode()));
        if(isSenhasNaoConferem){
            throw new ValidateException("A senha atual não confere.");
        }
        user.setNovaSenhaDigitada(senhaNova);
        user.setNovaSenhaConfirmacao(senhaConfirmacao);
    }
    user.validate();
    HibernateUtil.currentSession().save(user);
    return user;
}
}
```

```
-----
Java file:  CancelarSolicitacaoEmprestimoLeitorFlow.java
-----
```

```
package br.com.infowaypi.jbook.flow;

public class CancelarSolicitacaoEmprestimoLeitorFlow {

    public List<Emprestimo> getBuscarEmprestimosSolicitados(UsuarioInterface leitor) {
        return EmprestimoManager.getBuscarEmprestimosSolicitados(leitor);
    }

    public void cancelarEmprestimo(Emprestimo emprestimo){
        EmprestimoManager.cancelarEmprestimo(emprestimo);
    }
}
```

Java file: EtiquetaTomboFlow.java

```
package br.com.infowaypi.jbook.flow;

public class EtiquetaTomboFlow {

    public ResumoImpressaoEtiquetaTombo imprimirEtiquetasTombo(int qtdPaginas) throws Exception {
        List<DataSourceEtiquetaTombo> dataSource= EtiquetaTomboManager.getRelatorio(qtdPaginas);
        byte[] arquivo = EtiquetaTomboManager.getBytesRelatorio(dataSource);
        ResumoImpressaoEtiquetaTombo resumoEtiquetaTombo = new ResumoImpressaoEtiquetaTombo(arquivo);
        return resumoEtiquetaTombo;
    }

}
```

Java file: SolicitacaoEmprestimoFlow.java

```
package br.com.infowaypi.jbook.flow;

public class SolicitacaoEmprestimoFlow {

    public ResumoExemplares buscarPublicacao(String titulo, String assunto, String autor, TipoPublicacaoEnum tipoDePublicacao)throws ValidateException {
        return EmprestimoManager.buscarPublicacao(titulo, assunto, autor, tipoDePublicacao);
    }

    public Exemplar selecionarExemplar(UsuarioInterface leitor, ResumoExemplares resumoExemplares, Exemplar exemplar) throws ValidateException {
        return EmprestimoManager.selecionarExemplar(leitor, resumoExemplares, exemplar);
    }

    public void confirmarSolicitacao(UsuarioInterface leitor, Exemplar exemplar, PeriodoSolicitacaoEmprestimoEnum periodoEmprestimo) throws Exception {
        EmprestimoManager.confirmarSolicitacao(leitor, exemplar, periodoEmprestimo);
    }

}
```

=====
=====
Actual package: "manager"
=====
=====

Java file: EmprestimoManager.java

```
package br.com.infowaypi.jbook.manager;

public class EmprestimoManager {

    public static ResumoExemplares buscarPublicacao(String titulo, String assunto, String autor, TipoPublicacaoEnum tipoDePublicacao) throws ValidateException {
        boolean semParametrosDePesquisa = true;
        SearchAgent saExemplaresDisponiveis = new SearchAgent();
        SearchAgent saExemplaresIndisponiveis= new SearchAgent();
        Stack<ParameterInterface> parametros = new Stack<ParameterInterface>();
        if (!Utils.isStringVazia(titulo)) {
            parametros.add(new LikeFull("publicacao.titulo", titulo));
            semParametrosDePesquisa = false;
        }
        if (!Utils.isStringVazia(assunto)) {
            parametros.add(new LikeFull("publicacao.assunto", assunto));
            semParametrosDePesquisa = false;
        }
        if (!Utils.isStringVazia(autor)) {
            parametros.add(new LikeFull("publicacao.autor", autor));
            semParametrosDePesquisa = false;
        }
        if (tipoDePublicacao != null) {
            parametros.add(new Equals("publicacao.tipoDePublicacao",tipoDePublicacao.getValor()));
            semParametrosDePesquisa = false;
        }
        if(semParametrosDePesquisa){
            throw new ValidateException("Ã% necessário inserir pelo menos um parÃ¢metro de pesquisa!");
        }
        ResumoExemplares resumo = new ResumoExemplares(
            buscarExemplaresDisponiveis(saExemplaresDisponiveis, parametros),
            buscarExemplaresIndisponiveis(saExemplaresIndisponiveis, parametros)
        );
        if (resumo.isExemplaresNaoLocalizados()) {
            throw new ValidateException("Nenhum item encontrado.");
        }
        return resumo;
    }

    private static Collection<Exemplar> buscarExemplaresDisponiveis(SearchAgent saExemplaresDisponiveis, Stack<ParameterInterface> parametros) {
```

```

Stack<ParameterInterface> parametrosExemplaresDisponiveis = new Stack<ParameterInterface>();
parametrosExemplaresDisponiveis.addAll(parametros);
parametrosExemplaresDisponiveis.add(new Equals("situacao", SituacaoExemplarEnum.DISPONIVEL.getValor()));
return saExemplaresDisponiveis.listByParam(parametrosExemplaresDisponiveis, Exemplar.class);
}

private static Collection<Exemplar> buscarExemplaresIndisponiveis(SearchAgent saExemplaresIndisponiveis, Stack<ParameterInterface> parametros) {
Stack<ParameterInterface> parametrosExemplaresIndisponiveis = new Stack<ParameterInterface>();
parametrosExemplaresIndisponiveis.addAll(parametros);
parametrosExemplaresIndisponiveis.add(new OR(new Equals("situacao", SituacaoExemplarEnum.SOLICITADO.getValor()), new Equals("situacao", SituacaoExemplarEnum.EMPRESTADO.getValor()) ));
return saExemplaresIndisponiveis.listByParam(parametrosExemplaresIndisponiveis, Exemplar.class);
}

public static Exemplar selecionarExemplar(UsuarioInterface leitor, ResumoExemplares resumoExemplares, Exemplar exemplar) throws ValidateException {
if (exemplar == null) {
throw new ValidateException("Não há exemplares disponíveis para solicitação de empréstimo.");
}
if (passouLimiteEmprestimos((Usuario) leitor)) {
throw new ValidateException(
"O limite de 02 (duas) solicitações de empréstimo foi atingido. Não será possível realizar a solicitação de empréstimo. ");
}
if (solicitouMesmaPublicacao((Usuario) leitor, exemplar)) {
throw new ValidateException("Não é possível solicitar empréstimo de uma mesma publicação já solicitada e em aberto.");
}
return exemplar;
}

public static boolean passouLimiteEmprestimos(Usuario leitor) {
SearchAgent sa = new SearchAgent();
sa.addParameter(new Equals("leitor", leitor));
sa.addParameter(new OR(
new Equals("situacao", SituacaoEmprestimoEnum.SOLICITADO.getValor()),
new Equals("situacao", SituacaoEmprestimoEnum.CONFIRMADO.getValor())
));
if (sa.resultCount(Emprestimo.class) > 1) {
return true;
}
return false;
}

public static boolean solicitouMesmaPublicacao(Usuario leitor, Exemplar exemplar) {
SearchAgent sa = new SearchAgent();
sa.addParameter(new Equals("leitor", leitor));
sa.addParameter(new OR(
new Equals("situacao", SituacaoEmprestimoEnum.SOLICITADO.getValor()),
new Equals("situacao", SituacaoEmprestimoEnum.CONFIRMADO.getValor())
));
Criteria criteria0 = sa.createCriteriaFor(Emprestimo.class);
Criteria criterial = criteria0.createCriteria("exemplar");
criterial.add(Restrictions.eq("publicacao", exemplar.getPublicacao()));
if (criterial.list().size() != 0) {
return true;
}
return false;
}

public static void confirmarSolicitacao(UsuarioInterface leitor, Exemplar exemplar, PeriodoSolicitacaoEmprestimoEnum periodoEmprestimo) {
Emprestimo emp = new Emprestimo();
exemplar.setSituacao(SituacaoExemplarEnum.SOLICITADO.getValor());
emp.setExemplar(exemplar);
emp.setLeitor((Usuario) leitor);
emp.setSituacao(SituacaoEmprestimoEnum.SOLICITADO.getValor());
Calendar c = Calendar.getInstance();
c.add(Calendar.WEEK_OF_MONTH, periodoEmprestimo.getValor());
emp.setDataPrevisaoDeDevolucao(c.getTime());
HibernateUtil.currentSession().save(emp);
SchedulerManager.agendarExpiracaoDeSolicitacaoDeEmprestimo(emp.getIdEmprestimo());
}

public static List<Emprestimo> getBuscarEmprestimosSolicitados(UsuarioInterface leitor) {
SearchAgent sa = new SearchAgent();
sa.addParameter(new Equals("situacao", SituacaoEmprestimoEnum.SOLICITADO.getValor()));
sa.addParameter(new Equals("leitor", leitor));
return sa.list(Emprestimo.class);
}

public static void cancelarEmprestimo(Emprestimo emprestimo){
emprestimo.setSituacao(SituacaoEmprestimoEnum.CANCELADO.getValor());
}

```

```
    emprestimo.getExemplar().setSituacao(SituacaoExemplarEnum.DISPONIVEL.getValor());
    SchedulerManager.cancelarExpiracaoDeSolicitacaoDeEmprestimo(emprestimo.getIdEmprestimo());
}
}
```

Java file: EtiquetaTomboManager.java

```
package br.com.infowaypi.jbook.manager;
```

```
public class EtiquetaTomboManager {

    private static final int NUMERO_DE_ETIQUETAS_POR_PAGINA = 14;

    public static Long getUltimoTombo() {
        SearchAgent sa = new SearchAgent();
        return ((EtiquetaTombo) sa.findById(1L, EtiquetaTombo.class)).getUltimoTombo();
    }

    private static void setUltimoTombo(Long ultimoTombo) {
        SearchAgent sa = new SearchAgent();
        EtiquetaTombo etiquetaTombo = (EtiquetaTombo) sa.findById(1L, EtiquetaTombo.class);
        etiquetaTombo.setUltimoTombo(ultimoTombo);
    }

    public static byte[] getBytesRelatorio(List<DataSourceEtiquetaTombo> dataSource) throws Exception {
        JHeatReport report = new JHeatReport("../file\\etiquetas-tombo.xml", dataSource);
        ByteArrayOutputStream output = new ByteArrayOutputStream();
        report.createPDF(output);
        return output.toByteArray();
    }

    public static List<DataSourceEtiquetaTombo> getRelatorio(int qtdPaginas) throws Exception {
        List<DataSourceEtiquetaTombo> dataSource = new ArrayList<DataSourceEtiquetaTombo>();
        Long ultimoTomboDispionivel = getUltimoTombo() + 1;
        for(int x = 0; x < qtdPaginas; x++){
            Long[] tombosDaPagina = new Long[NUMERO_DE_ETIQUETAS_POR_PAGINA];
            for (int i =0; i < NUMERO_DE_ETIQUETAS_POR_PAGINA; i++) {
                tombosDaPagina[i] = ++ultimoTomboDispionivel;
            }
            dataSource.add(new DataSourceEtiquetaTombo(tombosDaPagina));
        }
        setUltimoTombo(ultimoTomboDispionivel);
        return dataSource;
    }
}
```

Java file: NotificadorManager.java

```
package br.com.infowaypi.jbook.manager;
```

```
public class NotificadorManager {

    public static boolean notificarNovasAquisicoes(Exemplar exemplar, Usuario usuario){
        if(exemplar.getPublicacao().getExemplares().size() == 1){
            enviarNotificacaoDeNovasAquisicoes(exemplar.getPublicacao(), usuario);
        }
        return true;
    }

    private static void enviarNotificacaoDeNovasAquisicoes(Publicacao publicacao, Usuario usuario) {
        String assunto = "Nova aquisição para nossa biblioteca!";
        StringBuilder corpo = new StringBuilder();
        corpo.append("A biblioteca acaba de disponibilizar a partir deste momento mais um exemplar. \n");
        corpo.append("Seguem abaixo os dados da publicação adquirida. \n");
        corpo.append("\n Título: " + publicacao.getTitulo());
        corpo.append("\n Assunto: " + publicacao.getAssunto());
        corpo.append("\n Autor: " + publicacao.getAutor());
        corpo.append("\n Editora: " + publicacao.getEditora());
        corpo.append("\n Tipo de publicação: " + publicacao.getTipoDePublicacao().toLowerCase());
        corpo.append("\n\n--\n");
        corpo.append("\n JBook - Sistema de controle de empréstimos de livros Infoway");
        new EmailThread(usuario, "JBook", assunto, corpo.toString()).starta();
    }
}
```

Java file: SchedulerManager.java

```
package br.com.infowaypi.jbook.manager;
```

```
public class SchedulerManager {
```

```
private static final String triggerName = "TRIGGER_EXPIRA_SOLICITACAO_DE_EMPRESTIMO_N:";

private static final String jobName = "JOB_EXPIRA_SOLICITACAO_DE_EMPRESTIMO_N:";

public static void agendarExpiracaoDeSolicitacaoDeEmprestimo(Long idSolicitacao){
    SimpleTrigger trigger = new SimpleTrigger(triggerName + idSolicitacao, Scheduler.DEFAULT_GROUP, new Date(System.currentTimeMillis() + (24 * (60 * 60000))));
    JobDetail tarefa = new JobDetail(jobName + idSolicitacao, Scheduler.DEFAULT_GROUP, ExpiraSolicitacaoEmprestimoTask.class);
    tarefa.getJobDataMap().put("idSolicitacao", idSolicitacao);
    try {
        QuartzConfigurator.getScheduler().scheduleJob(tarefa, trigger);
    } catch (SchedulerException e) {
        e.printStackTrace();
    }
}

public static void cancelarExpiracaoDeSolicitacaoDeEmprestimo(Long idSolicitacao){
    try {
        QuartzConfigurator.getScheduler().deleteJob(jobName + idSolicitacao, Scheduler.DEFAULT_GROUP);
        System.out.println("Cancelando Job");
    } catch (SchedulerException e) {
        e.printStackTrace();
    }
}

public static String getJobname() {
    return jobName;
}

public static String getTriggername() {
    return triggerName;
}
}
```

```
=====
=====
Actual package: "msg"
=====
=====
```

```
-----
Java file: EmailThread.java
-----
```

```
package br.com.infowaypi.jbook.msg;
```

```
public class EmailThread extends Thread {

    private Set<Usuario> usuarios = new HashSet<Usuario>();

    private String nome;

    private String assunto;

    private String corpo;

    public EmailThread(Usuario usuario, String nome, String assunto, String corpo) {
        this.usuarios.add(usuario);
        this.nome = nome;
        this.assunto = assunto;
        this.corpo = corpo;
    }

    public EmailThread(Set<? extends Usuario> usuarios, String nome, String assunto, String corpo) {
        this.usuarios.addAll(usuarios);
        this.nome = nome;
        this.assunto = assunto;
        this.corpo = corpo;
    }

    public void run() {
        String destino = "contato-no-reply@infoway-pi.com.br";
        for (Usuario usuario : usuarios) {
            MailSender.mandarEmail(usuario, this.nome, this.assunto, this.corpo, destino);
        }
    }

    public void start(){
        this.start();
    }
}
```

```
-----
Java file: MailSender.java
-----
```

```
package br.com.infowaypi.jbook.msg;

public class MailSender {

    public static void mandarEmail(Usuario usuario, String nome, String assunto, String corpo, String destino) {
        Usuario usuarioDestino = new Usuario();
        Usuario usuarioRemetente = new Usuario();
        usuarioRemetente.setEmail(destino);
        usuarioRemetente.setNome(nome);
        usuarioDestino.setNome(usuario.getNome());
        usuarioDestino.setEmail(usuario.getEmail());
        Mensagem mensagem = new Mensagem();
        mensagem.setAssunto(assunto);
        mensagem.setAvisarRemetente(true);
        mensagem.setCorpo(corpo);
        mensagem.setDataMensagem(new Date());
        mensagem.setDestinatario(usuarioDestino);
        mensagem.setRemetente(usuarioRemetente);
        mensagem.setEnviarEmail(true);
        mensagem.enviarEmail();
    }

    public static void mandarEmailHTML(Usuario usuario, String nome, String assunto, String corpo, String destino) {
        Usuario usuarioDestino = new Usuario();
        Usuario usuarioRemetente = new Usuario();
        usuarioRemetente.setEmail(destino);
        usuarioRemetente.setNome(nome);
        usuarioDestino.setNome(usuario.getNome());
        usuarioDestino.setEmail(usuario.getEmail());
        Mensagem mensagem = new Mensagem();
        mensagem.setAssunto(assunto);
        mensagem.setAvisarRemetente(true);
        mensagem.setCorpo(corpo);
        mensagem.setDataMensagem(new Date());
        mensagem.setDestinatario(usuarioDestino);
        mensagem.setRemetente(usuarioRemetente);
        mensagem.setEnviarEmail(true);
        mensagem.enviarEmail();
    }

    public static void mandarEmailHTML(Set<Usuario> usuarios, String nome, String assunto, String corpo, String destino) {
        for (Usuario usuario : usuarios) {
            mandarEmailHTML(usuario, nome, assunto, corpo, destino);
        }
    }

    public static void mandarEmail(Set<Usuario> usuarios, String nome, String assunto, String corpo, String destino) {
        for (Usuario usuario : usuarios) {
            mandarEmail(usuario, nome, assunto, corpo, destino);
        }
    }
}

=====
=====
Actual package: "report"
=====
=====

-----
Java file:  EmprestimoReport.java
-----

package br.com.infowaypi.jbook.report;
```

```
public class EmprestimoReport {

    public ReportEmprestimoResumo gerarRelatorio( TipoPublicacaoEnum tipoPublicacao, Exemplar exemplares, Usuario leitores, SituacaoEmprestimoEnum situacaoEmprestimo, Date dataEmprestimoInicio, Date dataEmprestimoFinal,
        Date dataDevolucaoInicial,  Date dataDevolucaoFinal  ) {
        List<Emprestimo> emprestimos = buscaPorEmprestimos(tipoPublicacao, exemplares, leitores, situacaoEmprestimo, dataEmprestimoInicio, dataEmprestimoFinal, dataDevolucaoInicial, dataDevolucaoFinal);
        ReportEmprestimoResumo rre = new ReportEmprestimoResumo();
        rre.setEmprestimos(emprestimos);
        return rre;
    }

    private List<Emprestimo> buscaPorEmprestimos(TipoPublicacaoEnum tipoPublicacao, Exemplar exemplares, Usuario leitores, SituacaoEmprestimoEnum situacaoEmprestimo, Date dataEmprestimoInicio, Date dataEmprestimoFinal,
        Date dataDevolucaoInicial, Date dataDevolucaoFinal) {
        Criteria c = HibernateUtil.currentSession(Emprestimo.class).createCriteria(Emprestimo.class);
        if (tipoPublicacao !=null ){
            c.createAlias("exemplar","e");
            c.createAlias("e.publicacao","p");
            c.add(Restrictions.eq("p.tipoDePublicacao", tipoPublicacao.getValor()));
        }
        if (exemplares!=null)
```

```
        c.add(Restrictions.eq("exemplar", exemplos));
    if (leitores!=null)
        c.add(Restrictions.eq("leitor", leitores));
    if (situacaoEmprestimo !=null ){
        c.add(Restrictions.eq("situacao", situacaoEmprestimo.getValor()));
    }
    if (dataEmprestimoInicio!=null && dataEmprestimoFinal!=null )
        c.add(Restrictions.between("dataSolicitacao", dataEmprestimoInicio, dataEmprestimoFinal));
    if (dataDevolucaoInicial!=null && dataDevolucaoFinal!=null )
        c.add(Restrictions.between("dataDevolucao", dataDevolucaoInicial, dataDevolucaoFinal));
    return c.list();
}
}
```

Actual package: "resumo"

Java file: ReportEmprestimoResumo.java

```
package br.com.infowaypi.jbook.resumo;
```

```
public class ReportEmprestimoResumo {

    private List<Emprestimo> emprestimos;

    public List<Emprestimo> getEmprestimos() {
        return emprestimos;
    }

    public void setEmprestimos(List<Emprestimo> emprestimos) {
        this.emprestimos = emprestimos;
    }
}
```

Java file: ResumoExemplares.java

```
package br.com.infowaypi.jbook.resumo;
```

```
public class ResumoExemplares {

    private Collection<Exemplar> exemplaresDisponiveis;

    private Collection<Exemplar> exemplaresIndisponiveis;

    public ResumoExemplares(Collection<Exemplar> exemplaresDisponiveis, Collection<Exemplar> exemplaresIndisponiveis) {
        this.exemplaresDisponiveis = exemplaresDisponiveis;
        this.exemplaresIndisponiveis = exemplaresIndisponiveis;
    }

    public boolean isExemplaresNaoLocalizados(){
        return ((exemplaresDisponiveis == null || exemplaresDisponiveis.size() == 0)
            && (exemplaresIndisponiveis == null || exemplaresIndisponiveis.size() == 0));
    }

    public Collection<Exemplar> getExemplaresDisponiveis() {
        return exemplaresDisponiveis;
    }

    public Collection<Exemplar> getExemplaresIndisponiveis() {
        return exemplaresIndisponiveis;
    }
}
```

Java file: ResumoImpressaoEtiquetaTombo.java

```
package br.com.infowaypi.jbook.resumo;
```

```
public class ResumoImpressaoEtiquetaTombo implements Serializable {

    private static final long serialVersionUID = 6380683336794912126L;

    public ResumoImpressaoEtiquetaTombo(byte[] conteudoArquivo) {
        this.conteudoArquivo = conteudoArquivo;
    }

    private byte[] conteudoArquivo;

    public byte[] getConteudoArquivo() {
        return conteudoArquivo;
    }
}
```

```
}

public String getFileName() {
    return "Etiquetas_de_Tombo.pdf";
}
}

=====
=====
Actual package: "scheduler"
=====
-----

Java file: ExpiraSolicitacaoEmprestimoTask.java
-----

package br.com.infowaypi.jbook.scheduler;

public class ExpiraSolicitacaoEmprestimoTask implements Job {

    public void execute(JobExecutionContext arg0) throws JobExecutionException {
        Long idSolicitacaoDeEmprestimo = (Long) arg0.getJobDetail().getJobDataMap().get("idSolicitacao");
        expirarSolicitacaoDeEmprestimo(idSolicitacaoDeEmprestimo);
    }

    public boolean expirarSolicitacaoDeEmprestimo(Long idSolicitacaoDeEmprestimo) {
        Session sessao = HibernateUtil.currentSession();
        Transaction tx = sessao.beginTransaction();
        Emprestimo solicitacaoDeEmprestimo = (Emprestimo) sessao.load(Emprestimo.class, idSolicitacaoDeEmprestimo);
        if (solicitacaoDeEmprestimo.getSituacao().equals(SituacaoEmprestimoEnum.SOLICITADO.getValor())) {
            solicitacaoDeEmprestimo.setSituacao(SituacaoEmprestimoEnum.EXPIRADO.getValor());
            solicitacaoDeEmprestimo.getExemplar().setSituacao(SituacaoExemplarEnum.DISPONIVEL.getValor());
        }
        sessao.update(solicitacaoDeEmprestimo);
        tx.commit();
        return true;
    }
}

-----

Java file: QuartzConfigurator.java
-----

package br.com.infowaypi.jbook.scheduler;

public class QuartzConfigurator implements PlugIn{

    private static Scheduler scheduler = null;

    public void destroy() {
        try {
            scheduler.shutdown();
        } catch (SchedulerException e) {
            e.printStackTrace();
        }
    }

    public void init(ActionServlet servlet, ModuleConfig config) throws ServletException {
        try {
            scheduler = StdSchedulerFactory.getDefaultScheduler();
            scheduler.start();
        } catch (SchedulerException e) {
            e.printStackTrace();
        }
    }

    public static Scheduler getScheduler() {
        return scheduler;
    }

    public static void setScheduler(Scheduler scheduler) {
        QuartzConfigurator.scheduler = scheduler;
    }
}

=====
=====
Actual package: "util"
=====
-----

Java file: ConfiguraBaseDeDados.java
-----

=====
FIM
=====
```

