Arquitetura dos Sistemas Institucionais

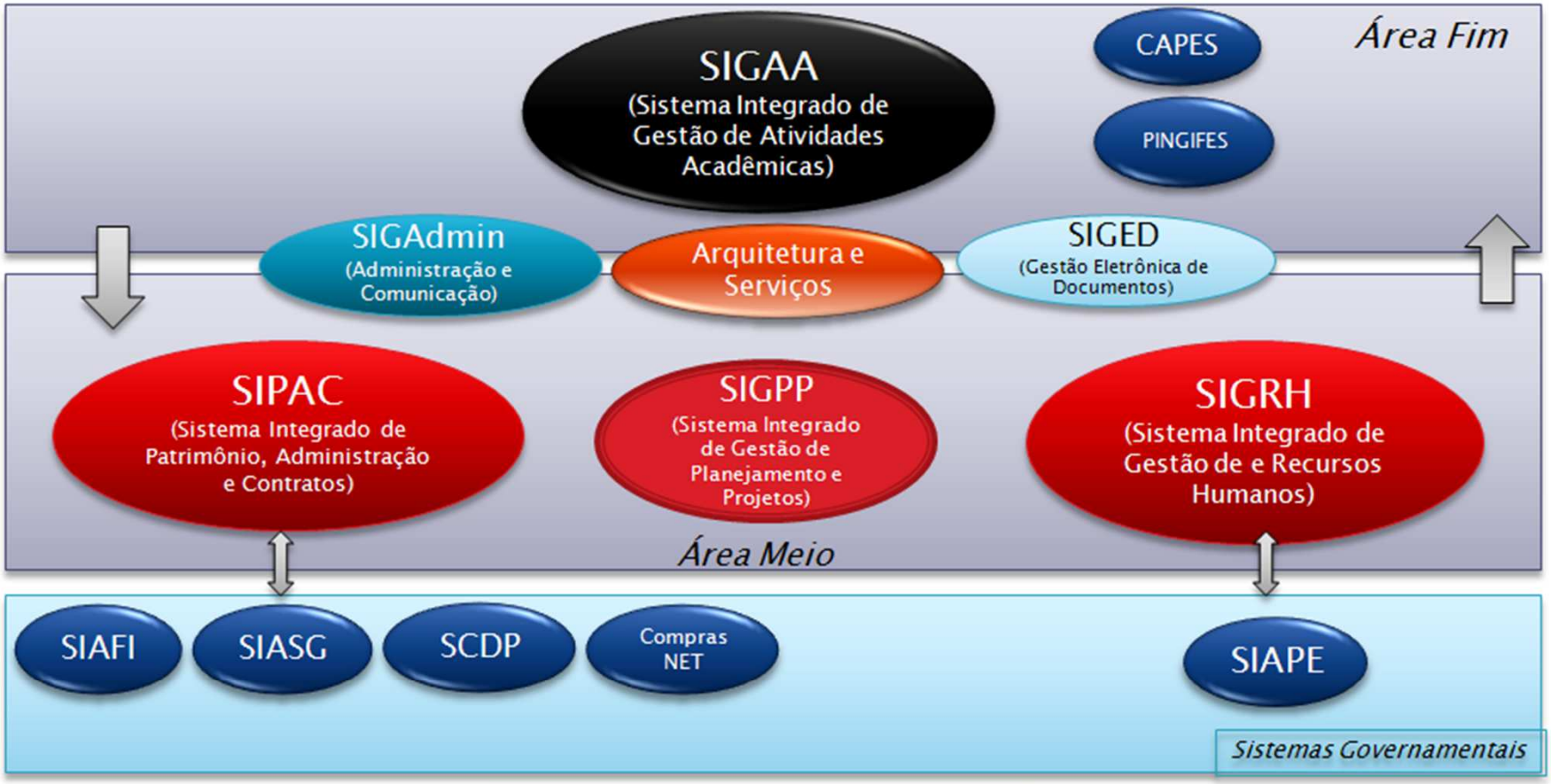
Diretoria de Sistemas
Superintendência de Informática – UFRN
dirsistemas@info.ufrn.br

Sumário

- Introdução
- Divisão em Camadas
  - Camada de Acesso a Dados
  - Camada de Apresentação
  - Camada de Negócio
- Classes Utilitárias
- Ambiente de Produção e Ferramentas

Introdução

Introdução



SIGAA: Ensino Infantil, Médio, Técnico, Graduação, Pós-Graduação Lato e Stricto, Pesquisa, Extensão, Monitoria, EAD, Produção Intelectual, Biblioteca, Ambientes Virtuais, Infra-Estrutura Física, Residência médica, etc.

SIPAC: Requisições, Almoxarifado, Orçamento, Compras, Registro de preços, Patrimônio, Licitação, Liquidação da despesa, Infra-Estrutura, Contratos, Convênios, Bolsas, Faturas, Transportes, Restaurante Universitário, Memorando Eletrônico, Protocolo, Boletim de Serviços, etc.

SIGPRH: Férias, Frequência, Financeiro, Dimensionamento de Força de Trabalho, Avaliação Funcional, Plano de saúde, Escalas, Serviços, Concursos, Capacitação, Plano Anual de Metas, Adkcional Noturno, Hora Extra, Comissões, Colegiado e resoluções, aposentadoria,

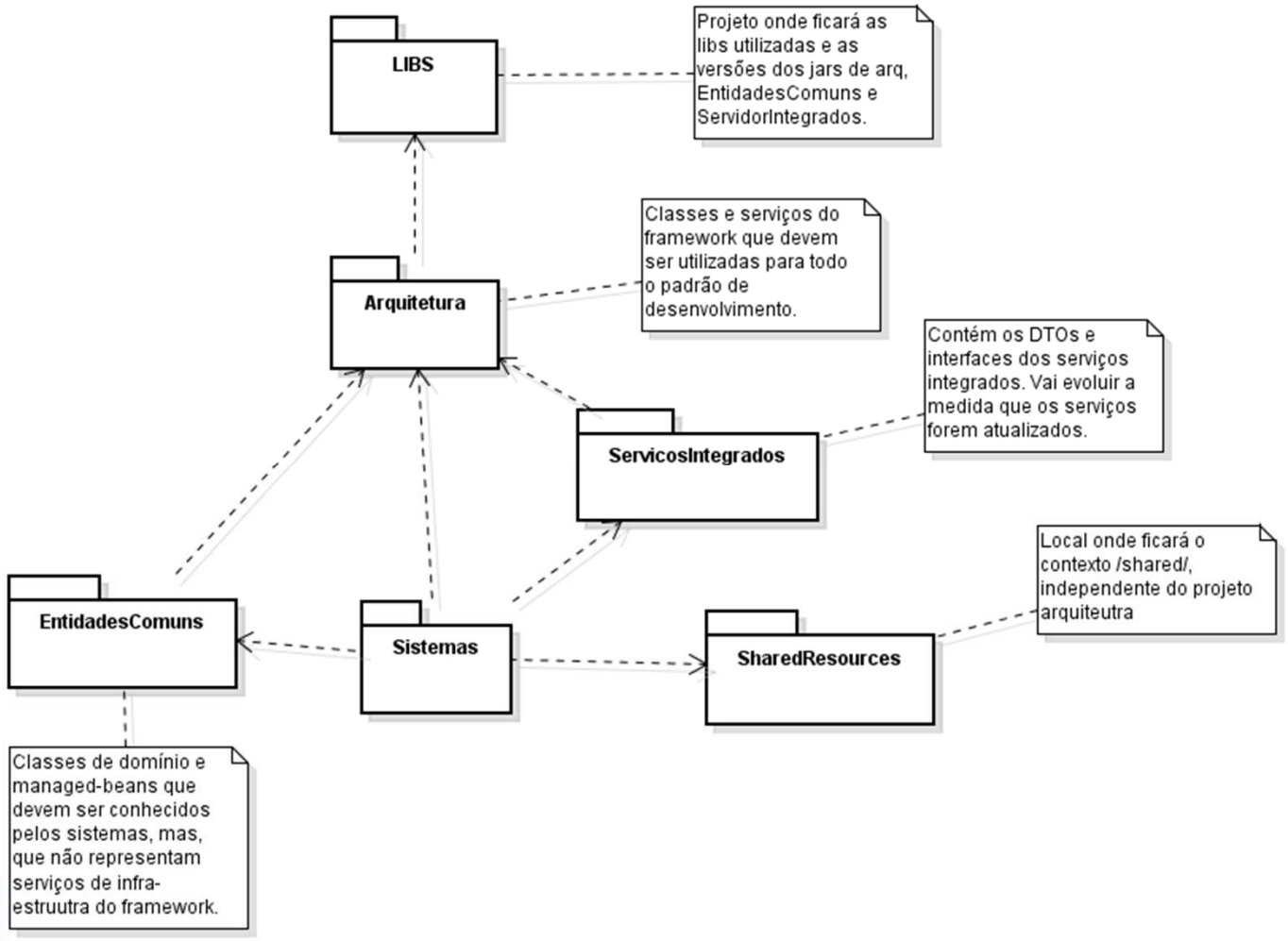
Introdução

- Metas
  - Interoperabilidade
  - Segurança
  - Escalabilidade
  - Alta Disponibilidade
  - Abstrair complexidade

Tecnologias Utilizadas

- Java 6
- Hibernate 3.2
- JavaServer Faces 1.2
- RichFaces 3.3
- Struts 1.2
- EJB 2.1
- Spring 2.5.6
- JBoss 4.2.2

Estrutura dos projetos



*

Métricas de código*

Sistema	Linhas	Classes	HTML/JSPs
Arquitetura	35334	410	0
EntidadesComuns	20932	202	0
ServicosIntegrados	6462	126	0
SharedResources	204978	0	32
SIPAC	796542	5116	4166
SIGRH	424440	2687	1673
SIGAdmin	26268	259	250
SIGPP	16397	122	115
SIGAA	646382	4750	4274
SIGED	8027	86	45
TOTAL	2185762	13758	10555

* Em 23/07/2012 *

Métricas de Banco de Dados*

Banco	Esquemas	Tabelas
Acadêmico	40	1135
Administrativo	55	1664
Comum	14	295
Arquivos	1	3
Log	1	12
SIGED	2	11
SIGConcurso	7	67
TOTAL	120	3187

* Em 23/07/2012

*

Divisão em Camadas

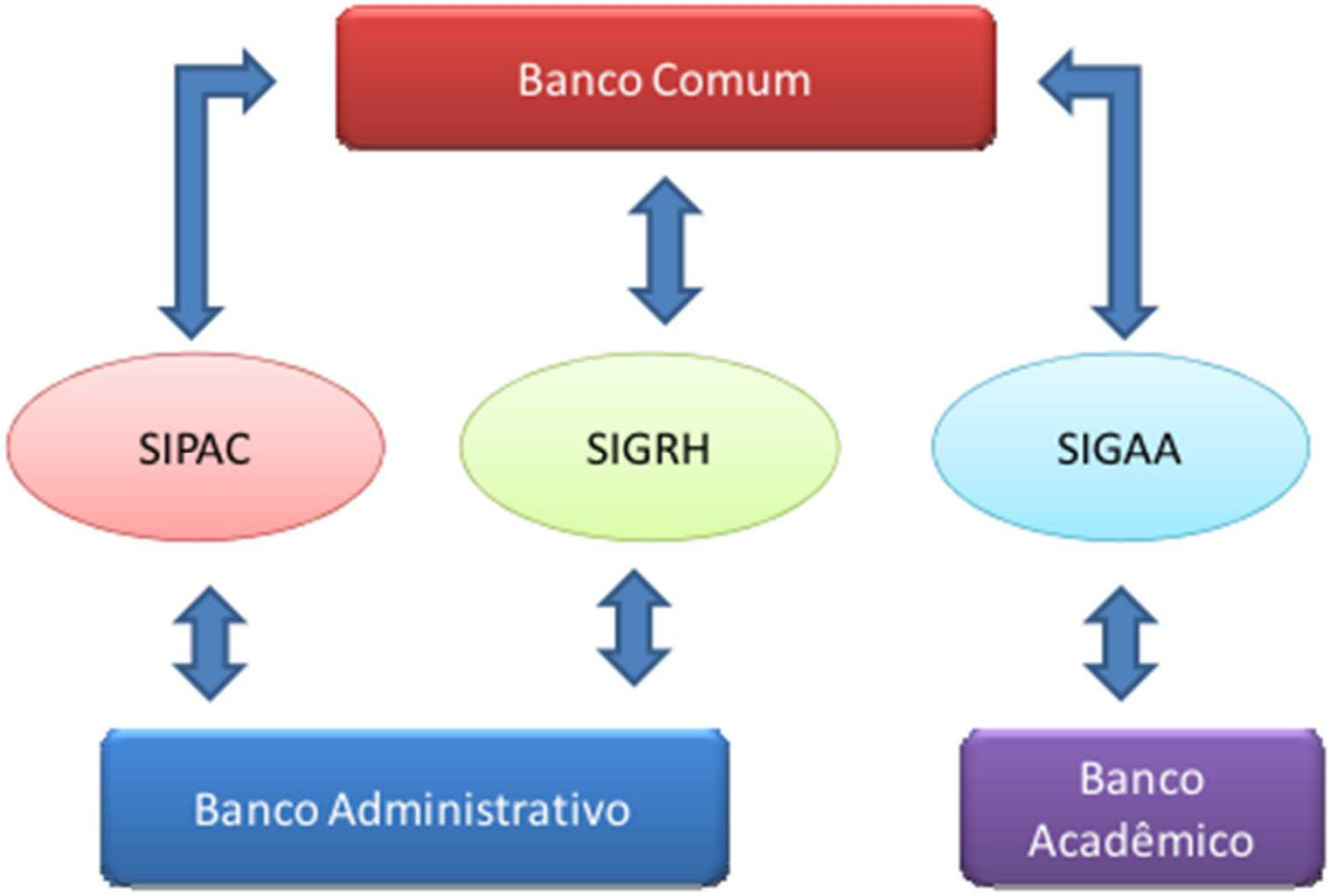
Camada de Acesso a Dados

Definição

- Acesso a dados: realiza a persistência e recuperação de dados. Normalmente trabalha com um banco de dados relacional.

- DAOs, Hibernate, JDBC

Bancos de Dados



*

Bancos de Dados

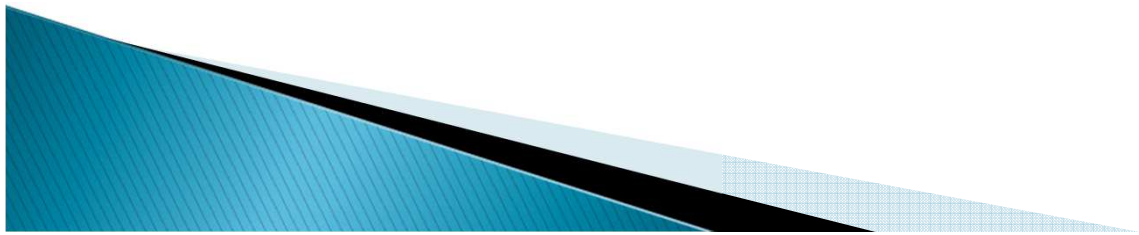
- ADMINISTRATIVO: base de dados para armazenar dados relacionados aos sistemas administrativos e de recursos humanos.

- ACADÊMICO: base de dados para armazenar dados relacionados ao sistema acadêmico.

- COMUM: base de dados com informações comuns a todos os sistemas, como por exemplo, a base de dados para autenticação de usuários, cadastro de permissões, entre outras.

Bancos de Dados

- PostgreSQL, versão 9.0
  - Usa Pool de Conexões:
  - Gerenciado pelo servidor de aplicações;
    - Usa o protocolo X/Open XA:
    - Permite que os vários banco de dados sejam acessados dentro de uma mesma transação;
- Utiliza o framework Hibernate para realização de mapeamento objeto relacional.
- Padrão de Projetos DAO (Data Access Object)
- Padrão Open Session In View;



Bancos de Dados

Configuração de DataSources

JBOSS_DIR/server/default/deploy/postgres-ds.xml

Bancos de Dados

```
<xa-datasource>

    <jndi-name>jdbc/SIPACDB</jndi-name>
    <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    <xa-datasource-property name="ServerName">desenvolvimento.info.ufrn.br</xa-datasource-property>
    <xa-datasource-property name="PortNumber">5432</xa-datasource-property>

    <xa-datasource-property name="DatabaseName">administrativo</xa-datasource-property>
    <xa-datasource-property name="User">sipac</xa-datasource-property>
    <xa-datasource-property name="Password">[redacted]</xa-datasource-property>

    <track-connection-by-tx/>

    <min-pool-size>1</min-pool-size>
    <max-pool-size>10</max-pool-size>

    <check-valid-connection-sql>select 1</check-valid-connection-sql>

</xa-datasource>
```

Bancos de Dados

- Datasources
  - jdbc/ComumDB
  - jdbc/SIPACDB
  - jdbc/SIGRHDB
  - jdbc/SIGAADB
  - jdbc/SIGEDDB
  - jdbc/SIGPPDB

Hibernate

Hibernate

- Framework para realização de mapeamento objeto-relacional;
- Versão 3.2;
- Mapeamentos em:
  - Arquivos XML (Apenas no SIPAC)
  - Utilizando anotações (SIPAC, SIGAA, SIGRH).

Mapeamento em XML

```
<hibernate-mapping>
    <class name="br.ufrn.sipac.compras.licitacao.dominio.CartaConvite"
        table="CARTA_CONVITE" dynamic-update="false" dynamic-insert="false" schema="compras">

        <id name="id" column="ID_CARTA_CONVITE" type="int" unsaved-value="0">
            <generator class="sequence">
                <param name="sequence">carta_seq</param>
            </generator>
        </id>

        <many-to-one fetch="join" name="fornecedor" class="br.ufrn.sipac.cadastro.dominio.Pessoa" cascade="none"
            outer-join="true" update="true" insert="true" access="property"
            column="ID_FORNECEDOR"/>

        <many-to-one fetch="join" name="procCompra" class="br.ufrn.sipac.compras.dominio.ProcessoCompra" cascade="none"
            outer-join="true" update="true" insert="true" access="property"
            column="ID_PROCESSO_COMPRA"/>

       <many-to-one fetch="join" name="usuario" class="br.ufrn.sipac.cadastro.dominio.Usuario" cascade="none"
            outer-join="true" update="true" insert="true" access="property"
            column="ID_USUARIO"/>

        <property name="dataCadastro" type="java.util.Date" update="true"
            insert="true" access="property" column="DATA" not-null="true"/>

    </class>

</hibernate-mapping>
```

Mapeamento com Annotations

```java
@Entity @Table(name = "tarefa", schema = "ava")
public class TarefaTurma implements PersistDB {

    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int id;

    @Column(name = "ativo")
    private Boolean ativo;

    private String titulo;

    @CriadoPor
    @ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name = "id_usuario_cadastro")
    private Usuario usuarioCadastro;

    @CriadoEm
    @Column(name = "data_cadastro")
    private Date dataCadastro;

    @OneToMany(mappedBy="tarefa", cascade=CascadeType.ALL)
    private List<RespostaTarefaTurma> respostas;
```

Configuração do Hibernate

- Arquivo .cfg.xml
  - Comum: comum.cfg.xml
  - SIPAC: sipac.cfg.xml
  - SIGRH e SIGAA: hibernate.cfg.xml

```
<property name="connection.datasource">java:/jdbc/SIGAADB</property>
<property name="transaction.factory_class">org.hibernate.transaction.JTATransactionFactory</property>
<property name="transaction.manager_lookup_class">org.hibernate.transaction.JBossTransactionManagerLookup</property>
<property name="jta.UserTransaction">UserTransaction</property>

<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="hibernate.show_sql">true</property>
<property name="connection.release_mode">on_close</property>
<property name="max_fetch_depth">5</property>
<property name="hibernate.cache.use_second_level_cache">true</property>
<property name="cache.provider_class">org.hibernate.cache.HashtableCacheProvider</property>

<mapping class="br.ufrn.sigaa.arq.dominio.OperacaoLote" />
<mapping class="br.ufrn.sigaa.arq.dominio.SeqAno" />
<mapping class="br.ufrn.rh.dominio.ClasseFuncional" />
<mapping class="br.ufrn.rh.dominio.TipoCategoria" />
```

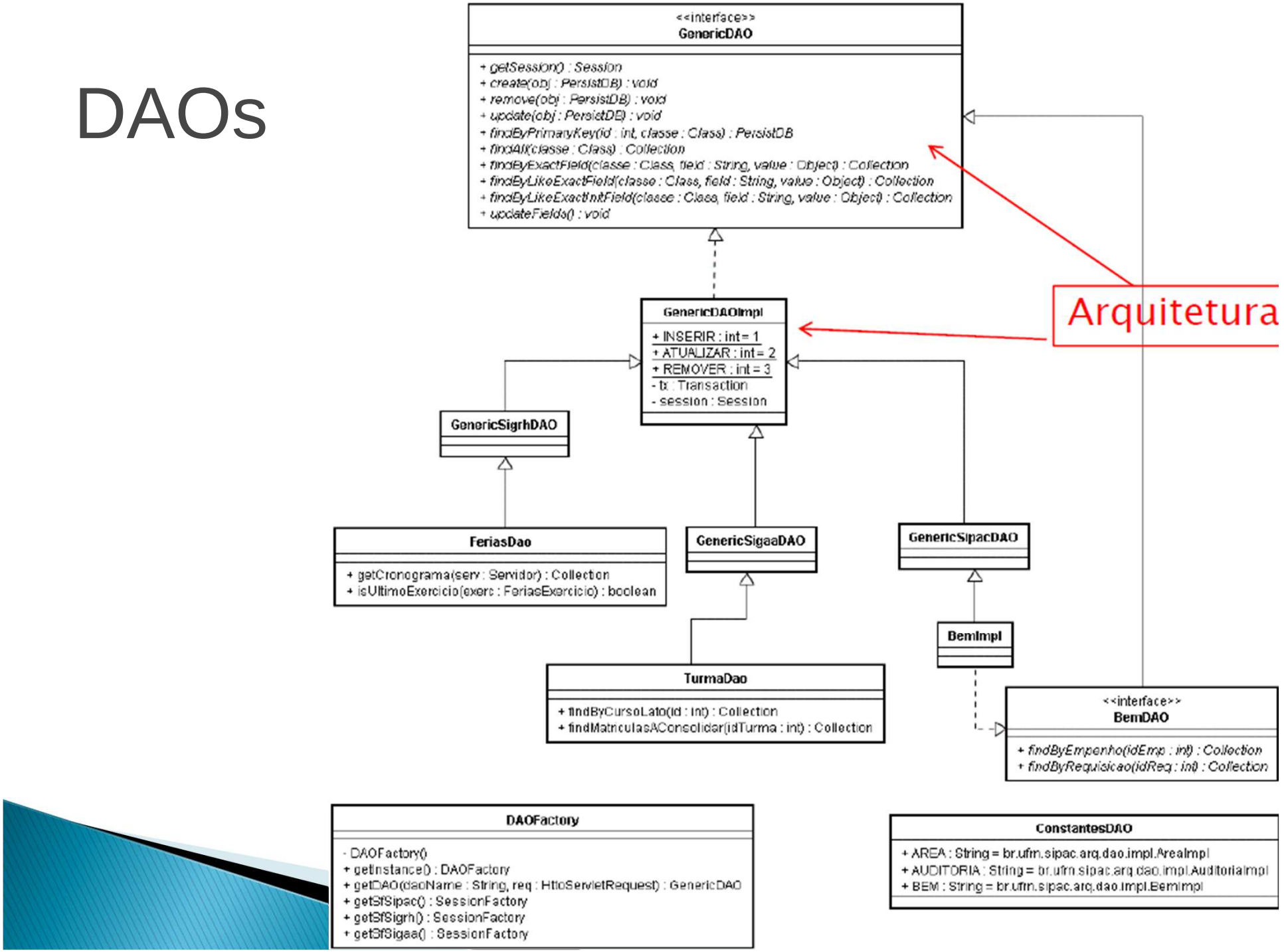
DAOs

DAOs

- Desvincula dos usuários da arquitetura a dependência do framework de mapeamento relacionado

- Objetos para abstrair e encapsular todo o acesso às fontes dados

DAOs


<<interface>>
GenericDAO
+ getSession() : Session
+ create(obj : PersistDB) : void
+ remove(obj : PersistDB) : void
+ update(obj : PersistDB) : void
+ findByPrimaryKey(id : int, classe : Class) : PersistDB
+ findAll(classe : Class) : Collection
+ findByExactField(classe : Class, field : String, value : Object) : Collection
+ findByLikeExactField(classe : Class, field : String, value : Object) : Collection
+ findByLikeExactIntField(classe : Class, field : String, value : Object) : Collection
+ updateFields() : void
GenericDAOImpl
+ INSERIR : int = 1
+ ATUALIZAR : int = 2
+ REMOVER : int = 3
- tx : Transaction
- session : Session
Arquitetura
GenericSigrhDAO
FeriasDao
+ getCronograma(serv : Servidor) : Collection
+ isUltimoExercicio(exerc : FeriasExercicio) : boolean
GenericSigaaDAO
TurmaDao
+ findByCursoLato(id : int) : Collection
+ findMatriculasAConsolidar(idTurma : int) : Collection
GenericSipacDAO
BemImpl
<<interface>>
BemDAO
+ findByEmpenho(idEmp : int) : Collection
+ findByRequisicao(idReq : int) : Collection
DAOFactory
- DAOFactory()
+ getInstance() : DAOFactory
+ getDAO(daoName : String, req : HttpServletRequest) : GenericDAO
+ getSfSipac() : SessionFactory
+ getSfSigrh() : SessionFactory
+ getSfSigaa() : SessionFactory
ConstantesDAO
+ AREA : String = br.ufrn.sipac.arq.dao.impl.AreaImpl
+ AUDITORIA : String = br.ufrn.sipac.arq.dao.impl.AuditoriaImpl
+ BEM : String = br.ufrn.sipac.arq.dao.impl.BemImpl

DAOs

```
public interface GenericDAO {

    public void create(PersistDB obj) throws DAOException;

    public void createNoFlush(PersistDB obj) throws DAOException;

    public void remove(PersistDB obj) throws DAOException;

    public void updateNoFlush(PersistDB obj) throws DAOException;

    public void update(PersistDB obj) throws DAOException;

    public void createOrUpdate(PersistDB obj) throws DAOException;

    public <T extends PersistDB> T findByPrimaryKey(int id, Class<T> classe) throws DAOException;

    public PersistDB findByPrimaryKeyLock(Serializable id, Class<?> classe)
            throws DAOException;

    public <T> Collection<T> findAll(Class<T> classe) throws DAOException;

    public <T> Collection<T> findAllAtivos(Class<T> classe, String orderColumn)
            throws DAOException;

    public <T> Collection<T> findAll(Class<T> classe, String orderBy[],
            String ascDesc[]) throws DAOException;
```

DAOs

- Cada sistema possui o seu DAO Genérico

```
public class GenericSigrhDAO extends GenericDAOImpl {

    public GenericSigrhDAO() {
        super(Sistema.SIGRH);
    }
    //...
```

```
public class GenericSipacDAO extends GenericDAOImpl {

    public GenericSipacDAO() {
        super(Sistema.SIPAC);
    }
    //...
```

```
public class GenericSigaaDAO extends GenericDAOImpl {

    public GenericSigaaDAO() {
        super(Sistema.SIGAA);
    }
```

Através do construtor, definimos qual o sistema associado ao DAO

Realizando consultas com Hibernate

- GenericDAO disponibiliza método getSession() para consultas com Hibernate

- O SessionFactory utilizado dependerá do sistema que foi setado no DAO

- Suporte ao padrão Open Session In View

Realizando consultas com Hibernate

```
public synchronized Session getSession() throws DAOException {

    if (session == null || !session.isOpen()) {
        try {
            session = getSF().openSession(interceptor);
        } catch (HibernateException e) {
            e.printStackTrace();
            throw new DAOException("Nao foi possivel abrir sessao: " + e.getMessage(), e);
        }
    }
    return session;
}
```

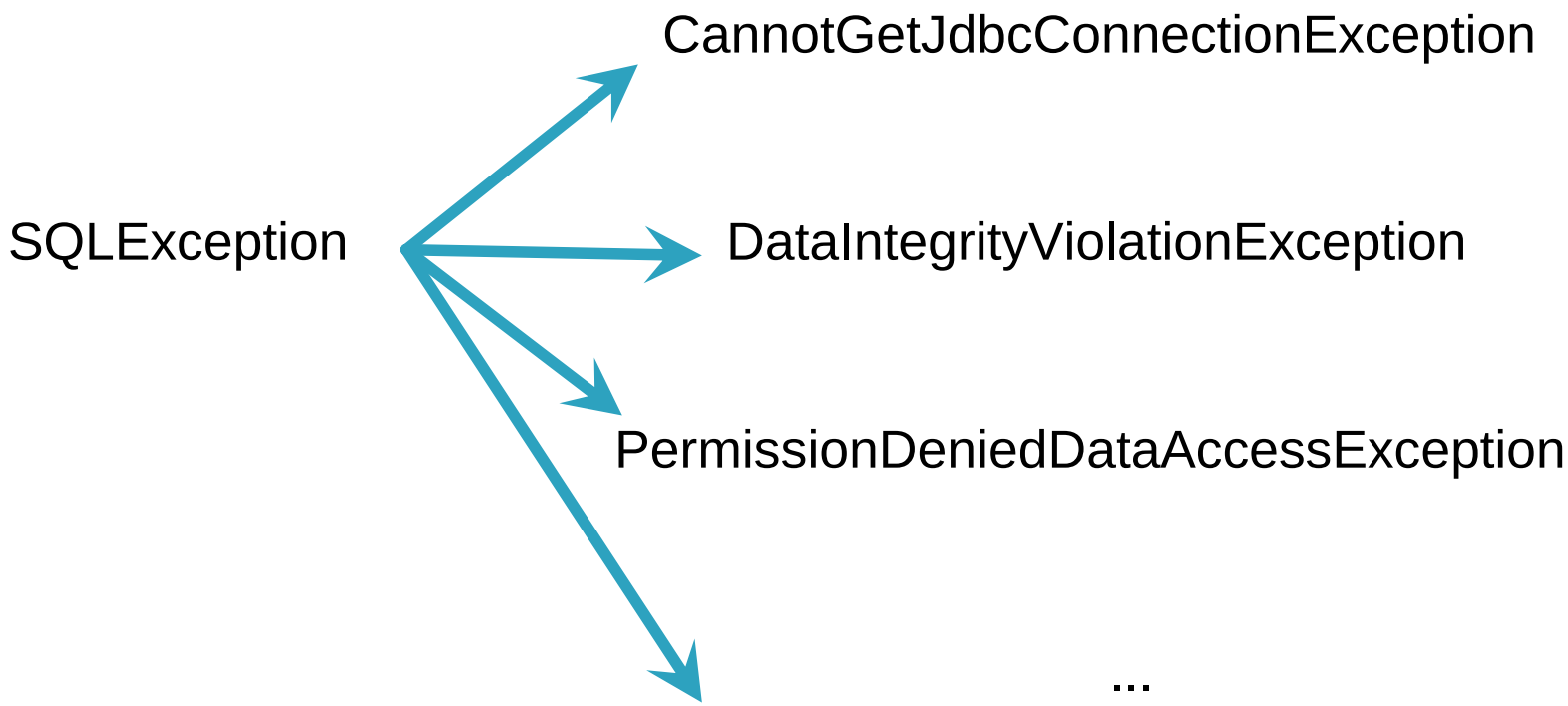
```
public SessionFactory getSF() {

    switch (sistema) {
    case Sistema.SIPAC:
        return DAOFactory.getInstance().getSfSipac();
    case Sistema.SIGAA:
        return DAOFactory.getInstance().getSfSigaa();
    case Sistema.PROTOCOLO:
        return DAOFactory.getInstance().getSfSipac();
    case Sistema.COMUM:
        return DAOFactory.getInstance().getSfComum();
    case Sistema.IPROJECT:
        return DAOFactory.getInstance().getSfIProject();
    case Sistema.SIGRH:
        return DAOFactory.getInstance().getSfSigrh();
    default:
        return null;
    }

}
```

JdbcTemplate

Task	Spring	You
Connection Management	√	
SQL		√
Statement Management	√	
ResultSet Management	√	
Row Data Retrieval		√
Parameter Declaration		√
Parameter Setting	√	
Transaction Management	√	

JdbcTemplate

- Tradução de Exceções



JdbcTemplate

```
int count = 0;
try {
        PreparedStatement st = getConnection().
                prepareStatement("select count(*) from cliente");
        ResultSet rs = st.executeQuery();
        if (rs.next()) count = rs.getInt(1);
} catch(SQLException e) {
        log.error(e);
} finally {
        try { if (stmt != null) stmt.close(); }
        catch(SQLException e) { log.warn(e); }
        try { if (conn != null) conn.close(); }
        catch(SQLException e) { log.warn(e); }
}
```

JdbcTemplate

```
int count = getJdbcTemplate().
            queryForInt( “select count(*) from
cliente”);
```

JdbcTemplate

- queryForInt
- queryForLong
- query
- queryForObject
- queryForList
- queryForMap
- update
- RowMapper
- ResultSetExtractor

Camada de Apresentação

Camada de Apresentação

- Camada de Apresentação: responsável por controlar a interação entre o usuário e o software ou entre dois softwares

- Páginas HTML, interfaces baseadas em janelas, webservices, etc.Exibem informação a um usuário (ou outro software) e interpretam ações do usuário (ou outro software)

Camada de Apresentação

- Frameworks de Apresentação
- SIPAC: Struts e JSF
- SIGAA: Struts e JSF
- SIGRH, SIGAdmin e SIGED: JSF
- JSF: Managed Beans
- Struts: Actions

Struts

Struts

- Framework MVC
- Utilizado no início dos projetos
- Substituído pelo JavaServer Faces
- Existem muitos casos de uso, principalmente no SIPAC, que utilizam Struts

Struts

- AbstractAction
  - Métodos auxiliares
  - getUsuarioLogado
  - checkRole
  - getDAO
  - Etc.

Struts


<<org.apache.struts.action>>
DispatchAction
<<br.ufrn.arq.web.struts>>
AbstractAction
+ getUsuarioLogado(req : HttpServletRequest) : UsuarioGeral
+ getSistema(req : HttpServletRequest) : Integer
+ execute(mov : Movimento, req : HttpServletRequest) : Object
+ execute(mov : Movimento, req : HttpServletRequest, mapping : ActionMapping) : Object
+ prepareMovimento(codMovimento : int, req : HttpServletRequest) : void
+ prepareMovimento(comando : Comando, req : HttpServletRequest) : void
+ getGenericDAO(req : HttpServletRequest) : GenericDAO
+ getDAO(constanteDAO : String, req : HttpServletRequest) : GenericDAO
+ getDAO(daoClass : Class, req : HttpServletRequest) : GenericDAO
+ checkRole(papel : int, req : HttpServletRequest) : void
+ checkRole(papeis : int[], req : HttpServletRequest) : void
+ checkRoleSipac(req : HttpServletRequest) : void
<<br.ufrn.sigaa.arq.struts>>
SigaaAbstractAction
<<br.ufrn.sigaa.arq.struts>>
AbstractCrudAction
<<br.ufrn.sigaa.pesquisa.struts>>
FinalizarPlanosTrabalhoAction
<<br.ufrn.sigaa.ensino.struts>>
AlteracaoMatriculaAction
<<br.ufrn.sipac.contratos.struts>>
BuscaContratoAction

Struts

```
public class CadNotaFiscalAction extends AbstractAction {

    public ActionForward execute(ActionMapping mapping,
            ActionForm form, HttpServletRequest req,
            HttpServletResponse res) {
            // ...
    }

}
```

JavaServer Faces

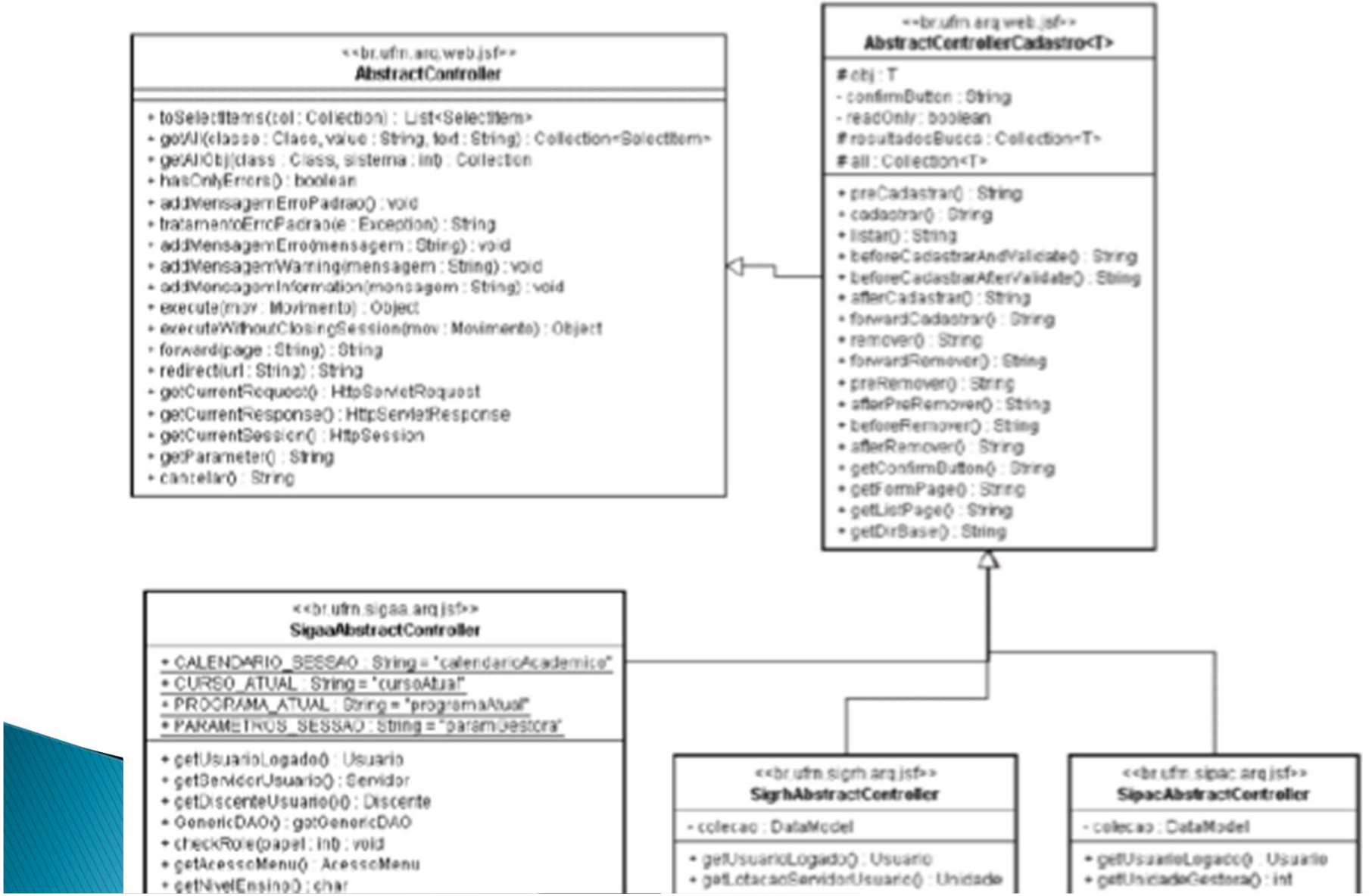
JavaServer Faces

- Substituto do Struts para a camada de apresentação
- Especificação JCP
- JavaServer Faces 1.2
- Implementação Sun Mojarra 1.2

JavaServer Faces

- Não necessita estender classes ou implementar interfaces
- Classes apenas auxiliares
  - AbstractController
  - AbstractControllerCadastro
  - <Sistema>AbstractController
- Provêem métodos auxiliares

JavaServer Faces



*

JavaServer Faces

- Integração com Spring
- @Component define que uma classe será um managed bean
- @Scope define o escopo do managed bean

JavaServer Faces

- RichFaces: biblioteca de componentes para dar suporte a AJAX
- Tags a4j
- Tags rich
- Demonstração online:
  - http://livedemo.exadel.com/richfaces-demo/index.jsp

ViewFilter

ViewFilter

- Filtro de Servlet com diversas funções
  - Segurança
  - Fechamento da sessão do Hibernate
  - Tratamento de Exceções
  - Realizar log das operações dos usuários

Camada de Domínio e Negócio

Camada de Negócio

- Negócio: lógica que o sistema precisa realizar com o domínio que se está trabalhando.

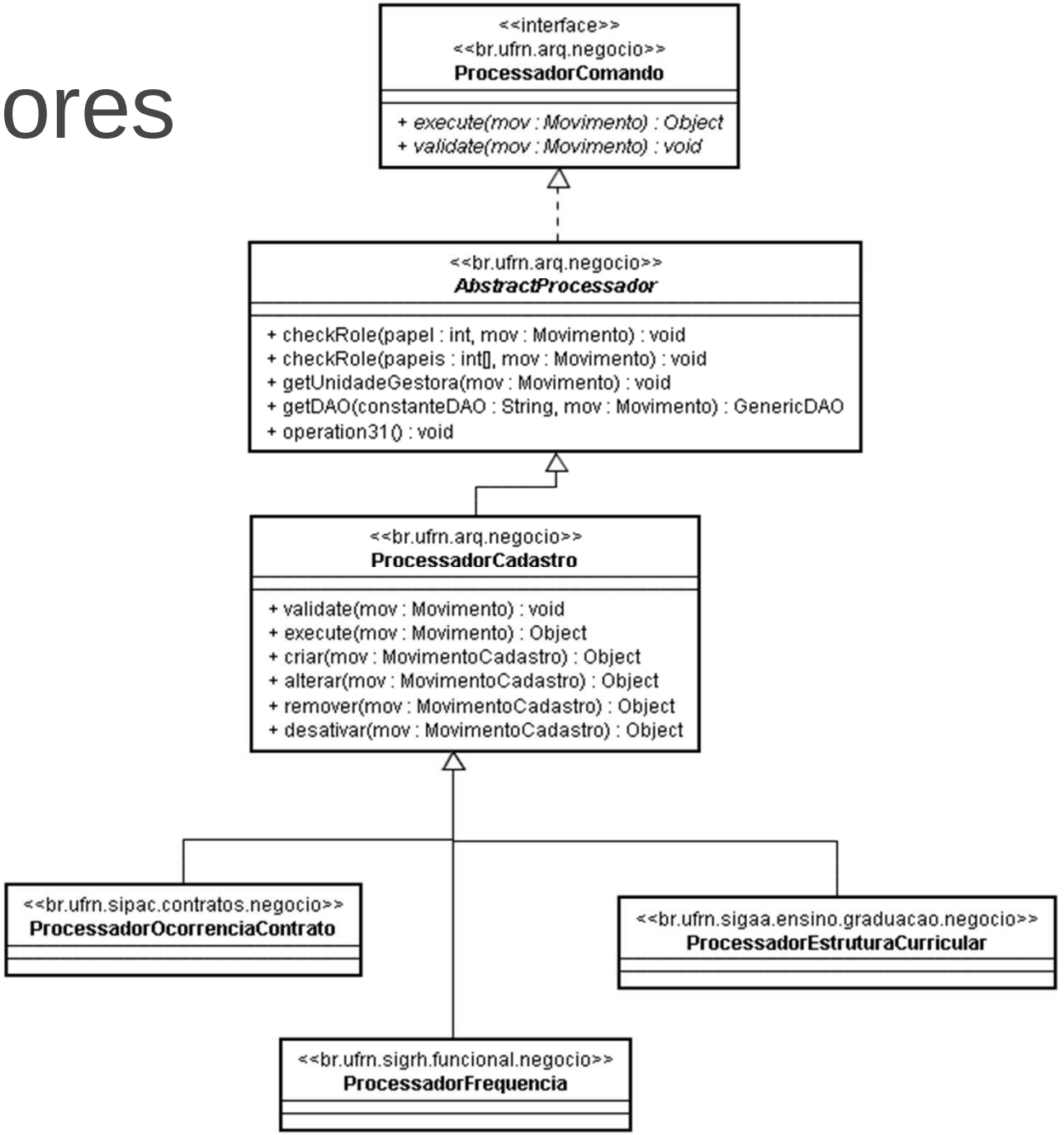
- Envolve cálculos baseados em entradas e dados armazenados, validações, etc.

Processadores

- Implementam lógica de negócio e gerenciam a persistência

```
public interface ProcessadorComando {

    public Object execute(Movimento mov)
     throws NegocioException,  ArqException,
     RemoteException;

    public void validate(Movimento mov)
     throws NegocioException, ArqException;

}
```

Processadores



Processadores

```
public class ProcessadorEnvioNotificacoes extends AbstractProcessador {

    public Object execute(Movimento mov) throws NegocioException, ArqException,
            RemoteException {

        // Validar dados
        validate(mov);

        Notificacao notificacao = ((MovimentoCadastro) mov).getObjMovimentado();

        // Popular destinatários definidos através de grupos
        if ( !ValidatorUtil.isEmpty(notificacao.getGruposDestinatarios()) ) {
            popularDestinatarios(notificacao, mov);
        }

        notificar(notificacao, mov.getUsuarioLogado());
        gravarNotificacao(notificacao, mov);

        return notificacao;
    }
```

Movimentos e Comandos

- Realiza a transição da camada de apresentação para a camada de negócio

  - Movimentos
    - Transferem objetos da apresentação para um processador

  - Comandos
    - Indicam que processador deve ser utilizado para realizar a lógica de negócio

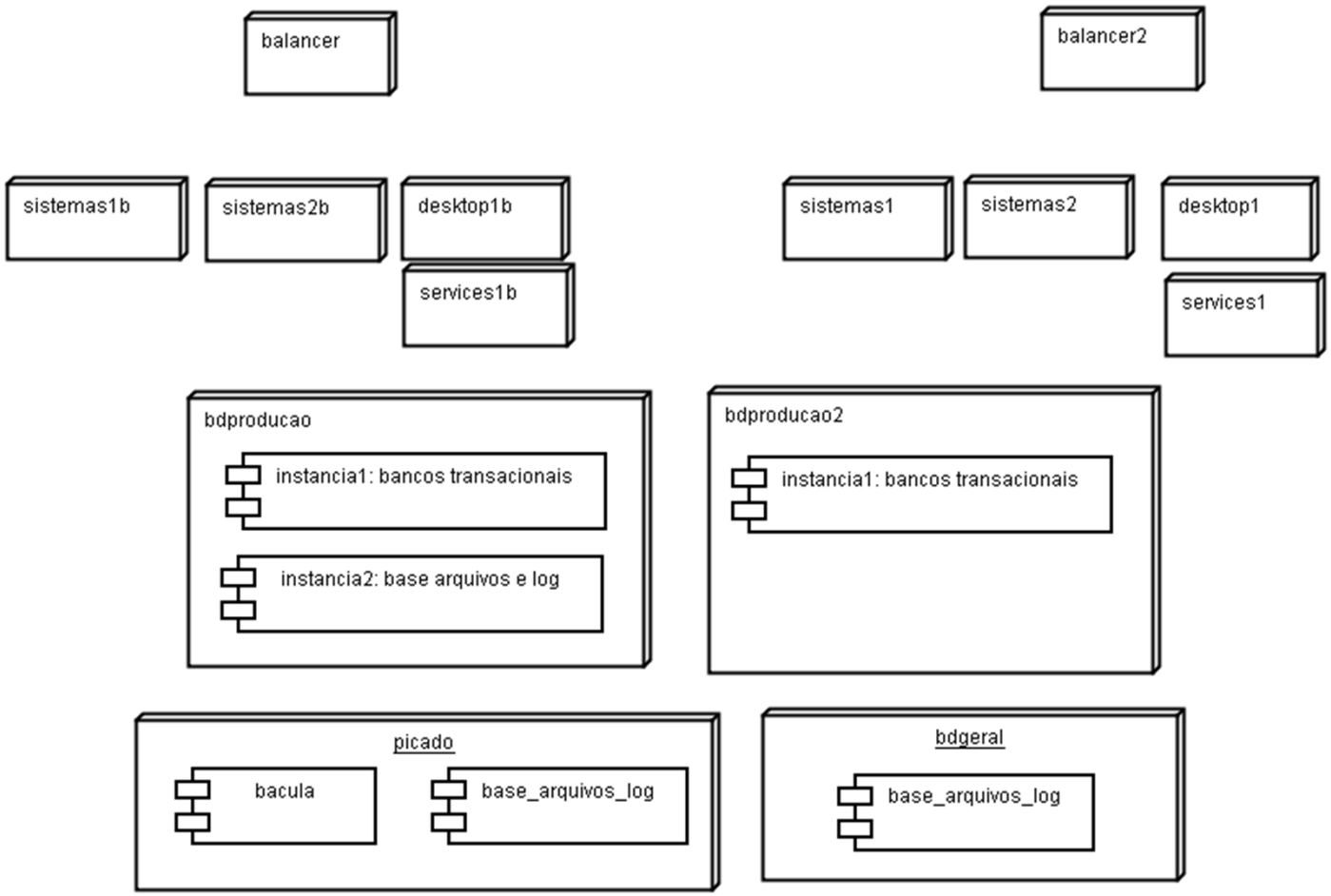
Comandos

- Para chamar um processador a partir da camada de apresentação:
  - Criar um movimento contendo os objetos que se deseja enviar para o domínio
  - Criar um novo comando informando que processador deve ser chamado
  - Chamar o método execute na action (Struts) ou managed bean (JSF) passando o movimento e o comando como parâmetros

Classes Utilitárias

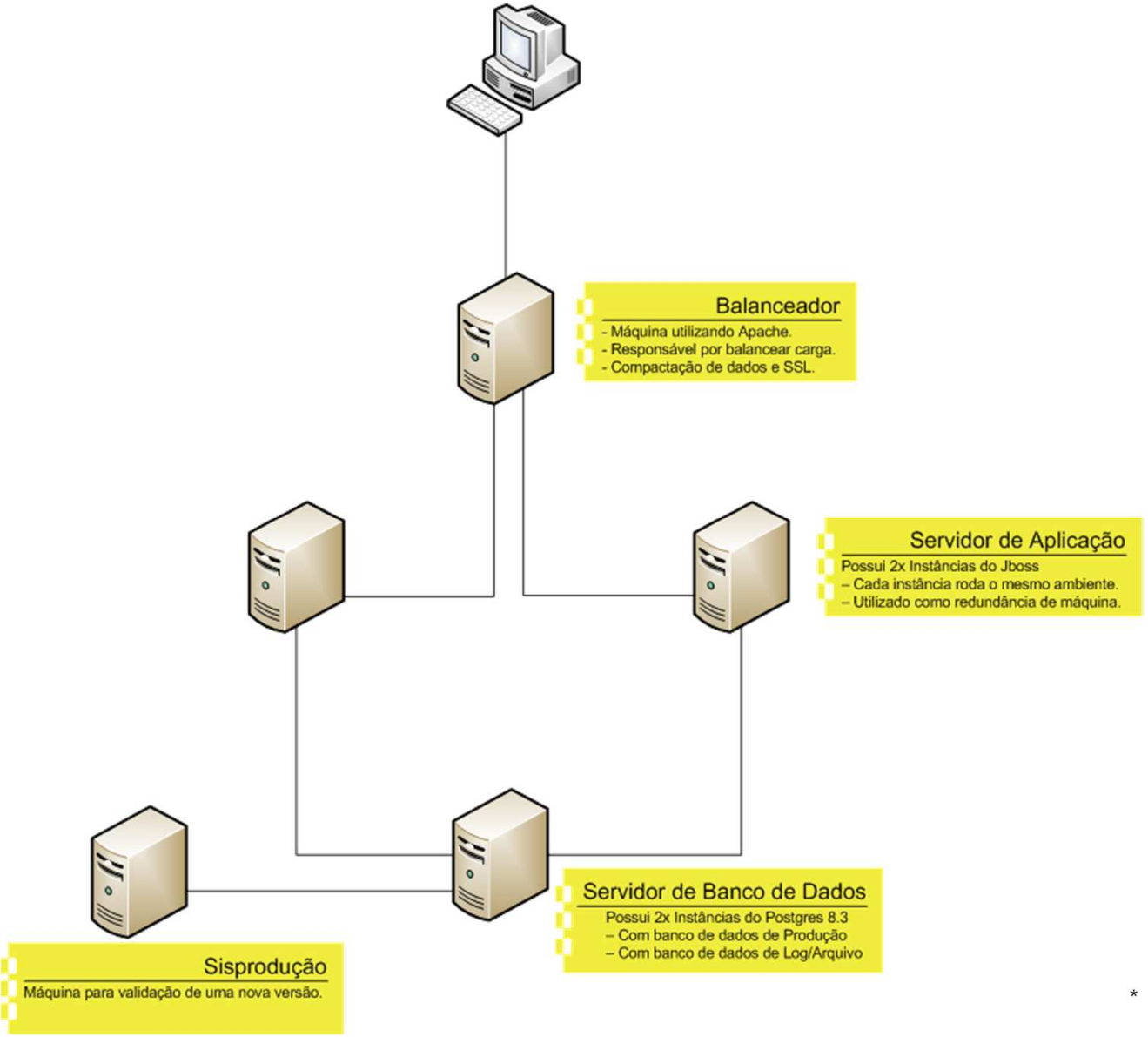
- AmbienteUtils
- CalendarUtils
- CodigoBarrasServlet
- EstadosHelper
- Formatador
- ImageUtils
- JasperReportsUtil
- ReflectionUtils
- StringUtils
- ValidatorUtils
- UFRNUtils

Ambiente de Produção



*

Ambiente de Produção



*

Ferramentas

- Apache + Mod JK
- O JK é utilizado para balanceamento de carga entre servidores jboss.

- Junto ao Apache, utilizamos a compactação de dados GZIP.
- Mod_deflate no apache

- Processo de criptografia dos dados(HTTPS), aplicado junto ao apache.
- Certificados gerados pela ipCA.

*

Ferramentas

- Eclipse
- IDE padrão para utilizar o código fonte do sistema.
- Necessário apenas o plugin para comunicação SVN

- JBoss
- Versão homologada 4.2.2
- Utilizando junto ao eclipse, para uso do código fonte do sistema.
- Necessário alterar o arquivo no diretório <ints>\conf\jboss-service.xml

*

Ferramentas

- Wiki
- Utilizado para documentação da SINFO.
- Manuais de utilização dos sistemas.
- Documentação dos casos de uso.
- Link para vídeos-aula dos módulos.

*

iProject

Ferramenta utilizada pela SINFO, para gerenciar demandas da instituição.
É possível a instituição utilizar a gerencia de demandas de forma interna, ou seja, entre os próprios usuário.
Importante para realizar acompanhamento das demandas e solicitações.
Existe um fórum, na qual é utilizado para centralizar dúvidas e pode ser visualizado por todos.

*

Contato: cooperacao@info.ufrn.br