 i6

|  |  |
| --- | --- |
| **Programa de Iniciação Científica**  **Relatório Final** | **PROTOCOLO**  Nº \_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_  Entrada: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Saída: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Recebido por** |

Título do projeto: Uma Proposta para Automatização de Formulários no Centro Universitário Unichristus

|  |  |
| --- | --- |
| Curso: Sistemas de Informação | |
| Bolsista: Giancarlo Lucciani Lopes Vasconcelos | |
| Professor(a) Orientador(a): Glaydson Vasconcelos de Sousa | |
| Data de início: 01/09/2013 | Data da conclusão: 31/07/2014 |

|  |
| --- |
| **RESUMO**  (Máximo de 25 linhas) |

O presente trabalho tem como finalidade propor uma solução tecnológica para automatizar o preenchimento dos formulários acadêmicos utilizados pelo Centro Universitário Unichristus (Unichristus), situado na cidade de Fortaleza-CE. A principal motivação para o desenvolvimento da proposta foi o fato de que os diversos atores que fazem parte da Unichristus utilizam, no seu dia-a-dia, diversos formulários para execução de suas atividades acadêmicas, sendo esses formulários atualmente preenchidos a mão, o que resulta em redundância de informações, ineficiência e grande possibilidade de erros. O sistema proposto, denominado FormChristus, foi projetado para execução em ambiente Web, utilizando a plataforma Java e tecnologias correlatas, tais como os Frameworks Java Server Faces (JSF), Hibernate e Spring. Foram inicialmente selecionados quatro formulários para servirem de base para a avaliação da solução proposta. O presente relatório apresenta também um breve relato da modelagem adotada, utilizando a análise orientada a objetos com o auxílio dos diagramas recomendados pela Linguagem de Modelagem Unificada (UML). São apresentados os resultados iniciais dos testes efetuados com o sistema, bem como propostas diversas outras funcionalidades que poderão ser incorporadas ao projeto no futuro.

**Palavras-chave:** Unichristus, Automatização de formulários, Sistemas Java para Web, Análise Orientada a Objetos.

|  |
| --- |
| **INTRODUÇÃO** |

Uma instituição de ensino superior, em suas atividades acadêmicas, necessita executar vários processos de trabalho para que possa desempenhar suas funções a contento, atingindo não somente os resultados esperados pelos seus alunos, mas também pela sociedade a qual atende e pelas instituições reguladoras e fiscalizadoras como, por exemplo, o Ministério da Educação.

O Centro Universitário Unichristus não foge dessa regra básica, tendo desenvolvido ao longo do tempo diversos processos com o objetivo de melhor executar suas atividades. Entre esses processos, destaca-se a criação de diversos formulários, necessários para as mais diferentes finalidades, como por exemplo:

* Registrar a inscrição de alunos em eventos, tais como Atividades Complementares;
* Registrar a participação de alunos e/ou professores em eventos externos, tais como Congressos, Seminários, etc.
* Permitir o registro da frequência de alunos em atividades de orientação de pesquisa;
* Permitir a inscrição de pessoas externas à Unichristus em eventos promovidos pela instituição
* Registrar relatórios de projetos de pesquisa desenvolvidos pela comunidade acadêmica.

Esses formulários são utilizados pelos diversos atores que fazem parte da Unichristus (alunos, professores, coordenadores, funcionários, público externo), para que possam desempenhar suas atividades acadêmicas.

Ocorre, entretanto, que a maioria dos formulários hoje utilizados pela Unichristus são impressos e preenchidos à mão, ou então baixados como arquivos de texto e editados eletronicamente em programas de processamento de texto, não existindo nenhum armazenamento ou reaproveitamento de informação de forma sistemática.

A utilização de formulários na Unichristus, na forma que é feita atualmente, apresenta diversas ineficiências, que poderiam ser reduzidas através de um projeto de automatização. Entre as principais deficiências desse processamento manual, pode-se apontar:

* Preenchimento (à mão ou digitando-se em processadores de texto) da mesma informação várias vezes, quando essas informações já poderiam vir preenchidas em um formulário eletrônico, considerando-se que existem informações comuns a vários formulários;
* Potenciais erros no preenchimento de informações de forma manual;
* Armazenamento físico dos formulários, estando sujeito a perdas. No caso dos alunos, em alguns casos eles não têm como comprovar que preencheram e entregaram um dado formulário, caso perca o seu recibo ou a instituição perca o seu formulário;

Essa automatização poderia ocorrer em diversas fases do processo, desde o desenvolvimento do *layout* do formulário, passando por seu preenchimento de forma automatizada com o uso de informações armazenadas em bancos de dados, permitindo o armazenamento em formato digital e a geração de arquivo para download pelo usuário. A automatização poderia atingir até mesmo o processamento do formulário, envolvendo de forma colaborativa os atores participantes do processo.

Como atores desse processo, os autores vivenciam no seu cotidiano acadêmico os problemas apontados. Seria natural então que se pensasse em formular uma proposta para automatizar o preenchimento de alguns dos formulários mais utilizados na Unichristus. Aproveitando a oportunidade oferecida pela instituição na forma de Projeto de Iniciação Científica, com duração de 11 (onze) meses, foi proposto então o projeto que resultou neste relatório.

O objetivo do projeto foi elaborar e implementar uma proposta para automatização do preenchimento dos formulários mais utilizados na Unichristus, modelando e implementando uma solução de software que permita sua utilização de forma sistemática. Com isso, pretendeu-se facilitar o manuseio dos formulários, por alunos, professores, coordenadores, funcionários e demais atores envolvidos, reduzindo-se o tempo gasto no preenchimento e os potenciais erros decorrentes da falta de validação automatizada desses documentos. Como benefício adicional, pode-se ainda citar a redução das impressões em papel, restringindo-se essas impressões ao mínimo necessário, conforme as exigências legais.

Como objetivos específicos do projeto, pode-se citar:

1. Identificar os formulários mais utilizados no Centro Universitário Unichristus, a partir de pesquisa com os principais atores envolvidos;
2. Projetar um modelo de dados para armazenar todas as informações necessárias ao preenchimento dos formulários;
3. Modelar uma solução tecnológica para o problema proposto, utilizando as tecnologias disponíveis para construção de aplicações Web, tendo como base os conhecimentos adquiridos nas disciplinas associadas à área de Engenharia de Software dentro da Unichristus, e
4. Implementar solução de software para automatização do preenchimento dos formulários, em plataforma web e/ou móvel.

|  |
| --- |
| **MATERIAL E MÉTODOS** |

**Plataformas de Hardware e Software**

O sistema FormChristus foi desenvolvido na plataforma Java, executando em ambiente Web. Conforme a especificação *Java EE* (*Java Enterprise Edition*), que disciplina a construção desse tipo de sistema, a aplicação necessita ser publicada em um servidor de aplicações compatível com essa plataforma. Para tal fim, foi utilizado servidor de aplicações *GlassFish*, utilizando a infraestrutura já existente no ambiente do Centro Universitário Unichristus.

O sistema é autocontido, não possuindo quaisquer interações com outros sistemas já existentes na Unichristus, em particular o sistema “aluno online” que já é utilizado pela instituição para registros acadêmicos. Poderão ser carregados inicialmente, para fins de teste, dados disponíveis em outros sistemas, mas sem comunicação entre eles. O FormChristus possui sua própria base de dados, não estando também previstas quaisquer exportações de dados para sistemas externos.

Para possibilitar o armazenamento dos dados produzidos pela aplicação, foi escolhido o *PostGresSQL*, que é um SGBD (Sistema de Gerenciamento de Bancos de Dados) robusto e bastante aceito pelo mercado.

Finalmente, o desenvolvimento do sistema foi feito utilizando-se o *Netbeans*, que é um ambiente integrado de desenvolvimento que fornece funcionalidades para facilitar o desenvolvimento de aplicações web. Os artefatos do sistema foram armazenados em repositório na nuvem, utilizando para controle de versão o *GitHub* (www.github.com), com o nome de projeto FormChristus. O projeto é aberto para leitura por qualquer usuário registrado no site que armazena os artefatos.

**Modelagem UML**

Para modelagem do sistema, utilizou-se a Linguagem de Modelagem Unificada (UML) que fornece um padrão para construção de modelos representativos do sistema. Isso permitiu a construção de vários diagramas representando as diversas partes do modelo, em particular os Diagramas de Casos de Uso, Diagramas de Classes e Diagramas de Sequencia. A modelagem foi feita com o auxílio da ferramenta *Astah*, disponível em www.astah.net.

**Descrição da Arquitetura do Sistema**

A arquitetura do sistema utiliza um modelo em camadas, composto de:

* Uma camada de **Apresentação**, que utiliza o *Primefaces*, uma implementação do framework *Java Server Faces* (JSF).
* Uma camada de **Negócio**, que utiliza o framework *EJB* e *CDI* (*Context and Dependency Injection*) para injeção de dependências.
* Uma camada de **Persistência**, que utiliza o *Hibernate* como framework de Mapeamento Objeto-Relacional, a biblioteca *Ehcache* para prover cache de segundo nível e injeção de dependência através do *CDI* para prover a conexão com o banco de dados juntamente com aplicação de um Pool de conexões implementado pelo servidor *GlassFish*.
* Uma camada utilitária adicional para visualização e impressão dos formulários em seu formato real, em papel ou em arquivo de formato PDF. Para tal, utilizou-se os frameworks *Jasper/IReports*.
* Uma camada de segurança para fornecimento de autenticação e autorização do acesso às funcionalidades do sistema. Para esse fim, utilizou-se o framework *Spring Security*.

**Biblioteca Primefaces**

O *Primefaces* é uma biblioteca de componentes baseada no framework *JSF* (*Java Server Faces*). Tem como principal objetivo adicionar componentes à interface da aplicação, tais como janelas de diálogo, campos de texto configuráveis, painéis com abas, menus, etc. Entre suas principais características, destacam-se:

* Biblioteca extensa de componentes visuais;
* Compatibilidade completa com a versão 2 do framework *JSF*;
* Disponível de forma gratuita e com código-fonte aberto;
* Baixo “*overhead*” para adição de componentes a uma aplicação *JSF*;

*Java Server Faces*, por sua vez, é um framework para aplicações web baseado na linguagem Java, projetado para simplificar o desenvolvimento de interfaces de usuário baseadas na web. É um framework web orientado por requisições, que implementa o padrão *MVC* (*Model-View-Controller*)[[1]](#footnote-1) para construção de interfaces utilizando componentes. JSF é uma tecnologia padronizada que foi formalizada em uma especificação que faz parte da Plataforma Java, em sua versão *Enterprise* (Java EE).

**Framework EJB e CDI**

*Enterprise Java Beans (EJB)* é um modelo de componentes especificado pela plataforma Java EE, projetado para resolver problemas complexos encontrados em aplicações corporativas. EJBs atuam no servidor e normalmente são classificados como componentes de negócio. Podem ser utilizados em diferentes situações, tais como: desenvolvimento distribuído, integração/conectividade com legado, processamento assíncrono, controle transacional, entre outros.

*Contexts and Dependency Injection* (*CDI*) define um conjunto de serviços para o ambiente *Java EE*, com o objetivo de tornar as aplicações mais simples de serem desenvolvidas. *CDI* fornece uma arquitetura que permite a componentes *Java EE* tais como *servlets*, *enterprise beans* e *JavaBeans* serem integrados ao ciclo de vida de uma aplicação com escopos bem definidos.

*CDI* também ajuda a reduzir o “*gap*” entre a camada enterprise e a camada web, permitindo, por exemplo, que uma página *JSF* acesse diretamente componentes *EJB*, ao invés de passar por *Managed Beans*. Isto se tornou possível após a especificação *Java EE* 6, que definiu o conceito de *Managed Beans* para unificar os vários tipos de *beans* existentes em uma aplicação *Java EE*, tais como *Managed Beans JSF*, *enterprise beans* e *CDI Beans*.

Dessa forma, é possível utilizar *CDI* para definir o componente *Managed Bean* que é utilizado pelas páginas *JSF* para fornecer e resgatar dados da aplicação, bem como para resolver as dependências desse *Managed Bean* com os componentes *EJB*, responsáveis pelas regras de negócio da aplicação (controle transacional e persistência).

**Framework Hibernate**

*Hibernate* é uma solução de mapeamento objeto-relacional (ORM) para Java, surgindo em 2001 como um framework de persistência *open source*. Fornece um serviço de consultas e persistência de alta performance para qualquer aplicação Java.

O framework mapeia classes Java para tabelas de bancos de dados e tipos de dados Java para tipos de dados SQL, retirando do programador grande parte das tarefas de programação relacionadas a persistência de dados.

*Hibernate* se posiciona entre os objetos Java tradicionais e o servidor de banco de dados para gerenciar todo o trabalho de persistir esses objetos, baseado em mecanismos e padrões apropriados de mapeamento. A Figura 1 mostra o funcionamento genérico do framework.

Figura 1 – Funcionamento do framework Hibernate



Fonte: http://www.tutorialspoint.com/hibernate/hibernate\_overview.htm

Entre as principais vantagens do uso do framework, pode-se destacar:

* Mapeamento de classes Java para tabelas de bancos de dados usando anotações ou arquivos XML, sem necessidade de escrever linhas de código;
* Fornece APIs para armazenar e recuperar objetos Java diretamente do banco de dados;
* A qualquer mudança no banco de dados ou em qualquer tabela, é necessário mudar somente a anotação ou arquivo XML;
* Abstrai tipos SQL não familiares, aproximando-se da nomenclatura Java mais utilizada;
* Não requer um servidor de aplicações para funcionar;
* Manipula associações complexas entre objetos;
* Minimiza o acesso ao banco de dados com estratégias inteligentes de “*fetching*”;
* Fornece mecanismos mais simples para consultas de dados.

**Frameworks Jasper e IReports**

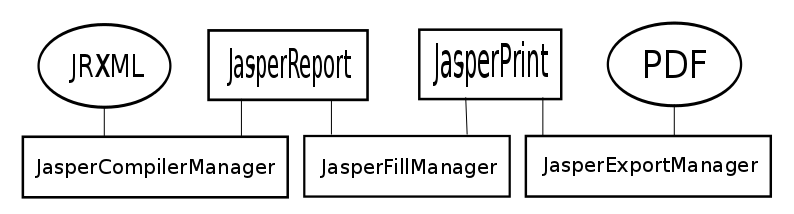
O *JasperReports* é um framework para a geração de relatórios. É uma ferramenta totalmente *open source* e gratuita, sendo uma das mais utilizadas atualmente com esse propósito. Entre as funcionalidades do *JasperReports* pode-se destacar:

* Capacidade de exportar relatórios para diversos formatos diferentes, tais como PDF, HTML, XML, XLS, etc.
* Aceita diversas formas de entrada de dados, tais como um arquivo XML ou CSV, conexão com um banco de dados, uma sessão do *Hibernate*, etc.
* Permite o uso de diagramas, gráficos e até códigos de barras.

Um aspecto importante do *JasperReports* é que o *layout* do relatório é definido em um arquivo XML, geralmente com a extensão .*jrxml*. Este arquivo possui as informações de formatação do relatório e, além disso, possui os campos que serão preenchidos posteriormente, de acordo com a fonte de dados (*data source*) utilizada.

Para gerar um relatório, o arquivo XML é compilado, resultando em um objeto do tipo *JasperReport*. Em seguida, o relatório é preenchido com os dados desejados, ficando então armazenado em um objeto do tipo *JasperPrint*. Esse objeto já representa o relatório finalizado, que pode ser enviado diretamente para impressão ou exportado para outro formato, como o PDF por exemplo. A Figura 2 mostra o funcionamento desse processo.

Figura 2 – Funcionamento do JasperReports



Fonte: http://community.jaspersoft.com

O *iReport* é um aplicativo gráfico, desenvolvido pela *JasperForge* (mesma desenvolvedora do *JasperReports*), que permite a criação do *layout* de um relatório, utilizando componentes visuais. Ao ser salvo, o *layout* gera automaticamente um arquivo com extensão .*jrxml* que poderá ser utilizada na aplicação em desenvolvimento. Para ser utilizada, a aplicação não requer conhecimento de XML, economizando tempo de desenvolvimento. O *iReport* disponibiliza ainda um conjunto de *templates* para serem utilizados diretamente, embora também seja possível criar os próprios *templates*.

O *iReport* geralmente é utilizado em conjunto com o *JasperReports*, uma vez que se torna complicado definir um *layout* de relatório escrevendo-se código em XML.

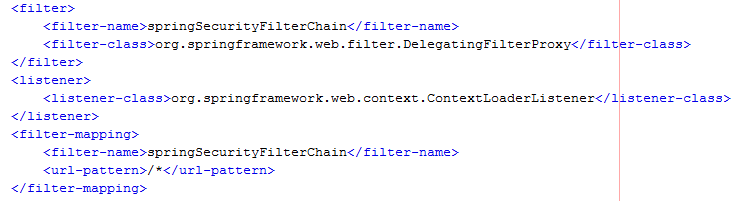
**Segurança**

Um dos requisitos básicos da aplicação é o controle do acesso de cada um dos possíveis usuários do sistema. Inicialmente foram definidos quatro possíveis tipos de usuário para o sistema: Aluno, Professor, Coordenador e Administrador. Cada um desses usuários deve ter à sua disposição somente as funcionalidades permitidas para a sua função. Além disso, o sistema deve também garantir a autenticação de cada usuário com seu próprio login e senha.

Para prover essas funcionalidades, foi utilizado o *Spring Security*, que é um framework *Java EE*, cujo objetivo é fornecer autenticação, autorização e outros serviços de segurança para aplicações corporativas. “Autenticação” é o processo de estabelecer se um “principal” é quem realmente diz ser (um “principal” geralmente significa um usuário, dispositivo ou algum outro sistema que pode executar alguma ação dentro da aplicação). “Autorização” se refere ao processo de decidir se um “principal” tem permissão para executar uma ação dentro da aplicação. Para se chegar ao ponto onde uma decisão de autorização é necessária, a identidade do “principal” já deve ter sido estabelecida pelo processo de autenticação.

A configuração do framework é feita através de um arquivo XML (*applicattionContext*), ao lado de algumas entradas no descritor da aplicação web (arquivo *web.xml*). Uma configuração típica para que o framework intercepte todas as requisições vindas do cliente é vista na figura 3.

Figura 3 – Configuração do spring framework no descritor da aplicação



Fonte: Descritor do Sistema FormChristus

Observa-se que o framework é configurado na forma de um filtro, presente na especificação *Java EE*, que neste caso está programado para interceptar todas as requisições oriundas do browser (“/\*).

Feita essa configuração, basta configurar o arquivo “*applicationContext*” com os parâmetros de segurança da aplicação. Nesse arquivo são definidos papéis (“*roles*”), qual a página de login da aplicação, onde estão armazenadas as senhas de cada usuário, quais recursos são permitidos para acesso por cada papel, etc.

|  |
| --- |
| **RESULTADOS E DISCUSSÃO** |

Para este projeto, foram selecionados pela coordenação do curso de Sistemas de Informação os quatro formulários abaixo:

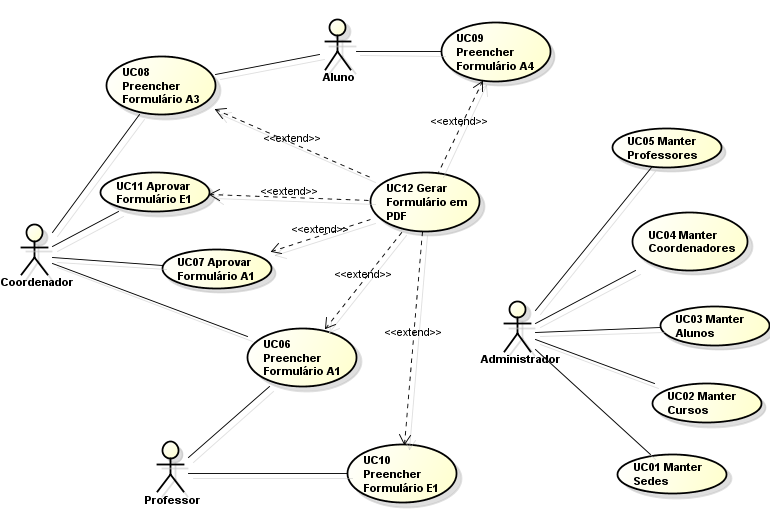
* A1 – Programa de Atividade Complementar
* A3 – Relatório de Atividade Complementar
* A4 – Inscrição em Atividade Complementar
* E1 – Programa de Extensão

A seguir, são demonstrados detalhes da implementação do sistema, utilizando-se o auxílio de diagramas fornecidos pela linguagem UML.

**Visão de Casos de Uso**

A pretensão inicial da equipe do projeto, conforme solicitação da Unichristus foi a automatização do preenchimento dos formulários, definindo-se assim as seguintes funcionalidades, apresentadas no Diagrama de Casos de Uso mostrado na Figura 4.

Figura 4 – Diagrama de Casos de Uso



Fonte: Modelagem do Sistema FormChristus

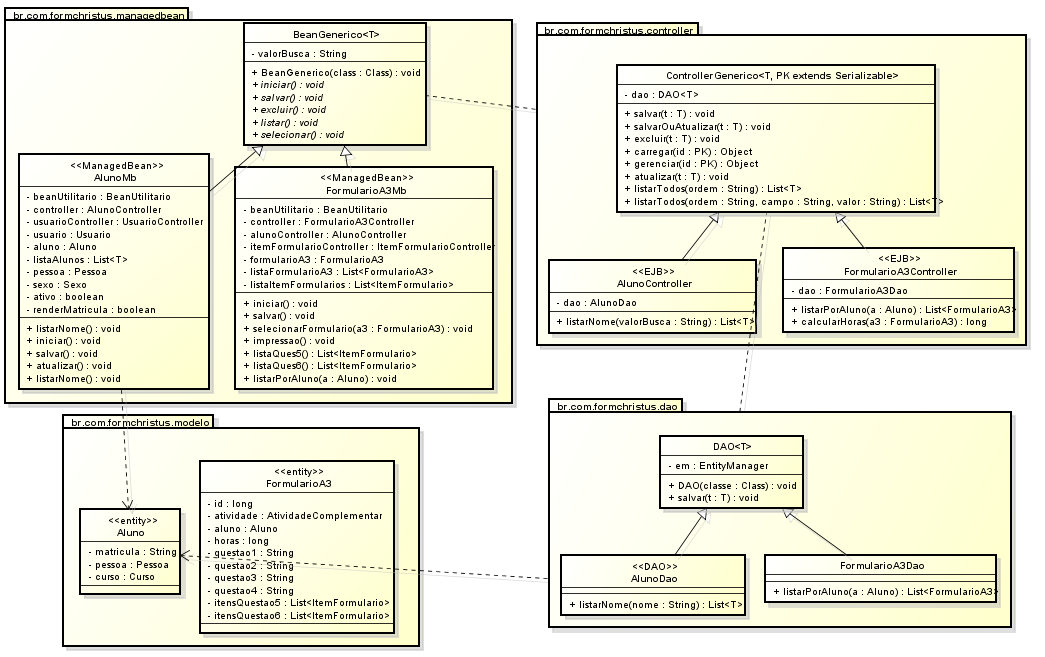
Todos os casos de uso mostrados na figura são considerados significativos e foram implementados para este projeto. Os formulários escolhidos foram modelados usando a ferramenta *IReports*, sendo para cada um deles gerado um arquivo com a extensão *jrxml*, conforme comentado na seção anterior deste relatório.

**Visão Lógica**

**Visão Geral**

O projeto e a divisão da aplicação em pacotes seguem a definição arquitetural exposta anteriormente. A divisão em pacotes objetiva facilitar o entendimento, o desenvolvimento e manutenção da aplicação. A Figura 5 mostra uma visão parcial dos pacotes do sistema, demonstrando o uso da arquitetura em camadas para as entidades Aluno e FormularioA3.

Figura 5 – Diagrama de Pacotes Parcial



Fonte: Modelagem do Sistema FormChristus

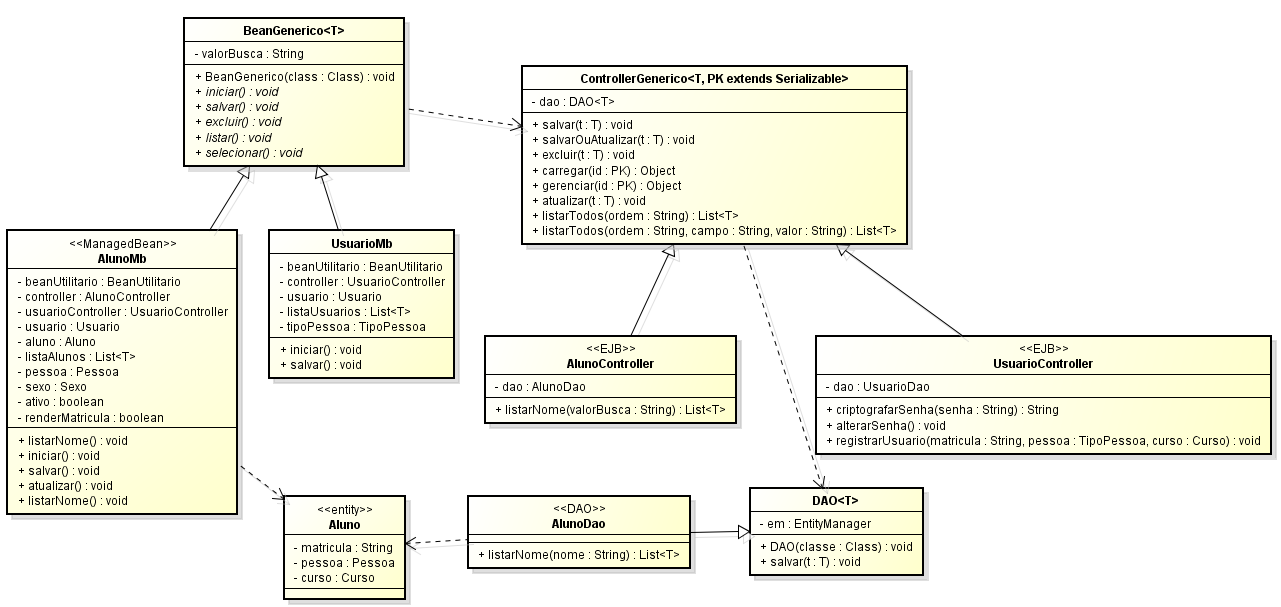
**Realização dos Casos de Uso**

Nesta seção demonstra-se a realização de dois casos de uso da aplicação, através de Diagramas de Classe e Sequencia UML.

**UC03 – Manter Alunos**

Este caso de uso é responsável pelo cadastramento, consulta, alteração e exclusão de alunos do sistema. Ao ser cadastrado, o aluno é também registrado como um usuário do sistema, sendo criado seu *login* e senha. A Figura 6 apresenta as principais classes envolvidas na realização desse caso de uso.

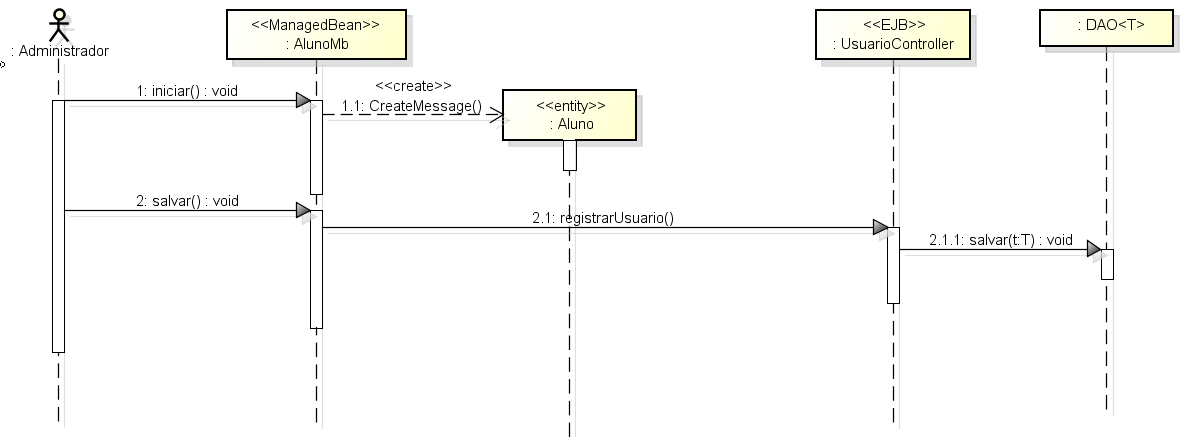
Figura 6 – UC03 – Manter Alunos (Cadastro) – Principais Classes Envolvidas



Fonte: Modelagem do Sistema FormChristus

Na realização desse UC, a partir da página de cadastramento, é acionado o *Managed* *Bean* AlunoMb, que cria uma entidade Aluno e a devolve para a apresentação. Ao serem preenchidos os campos do aluno e escolhida a opção salvar, o *Managed Bean* é novamente acionado, passando a requisição para o Controlador UsuarioController, que por sua vez aciona o DAO para gravação do aluno. Esse processo é mostrado na Figura 7.

Figura 7 – Diagrama de Sequência para o Cadastro de Aluno (UC01)

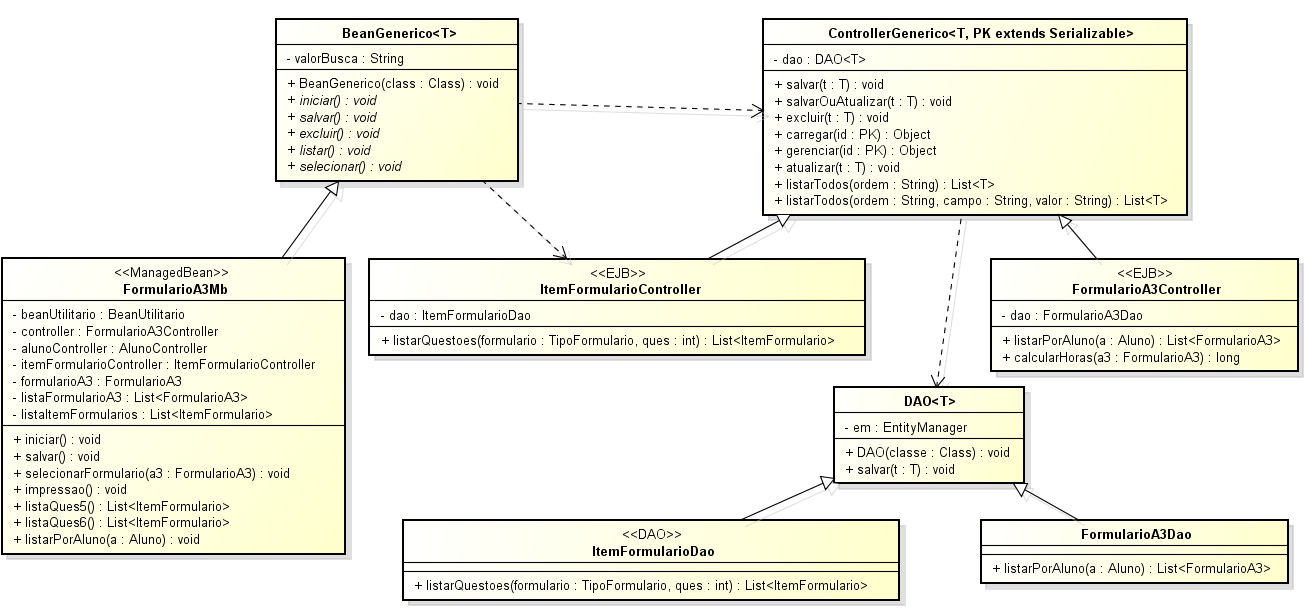


Fonte: Modelagem do Sistema FormChristus

**UC08 – Preencher Formulário A3**

Este UC é responsável pelo preenchimento do formulário A3, que se constitui no relatório de Atividade Complementar executada pelo aluno. As principais classes envolvidas nesse caso de uso são mostradas na Figura 8.

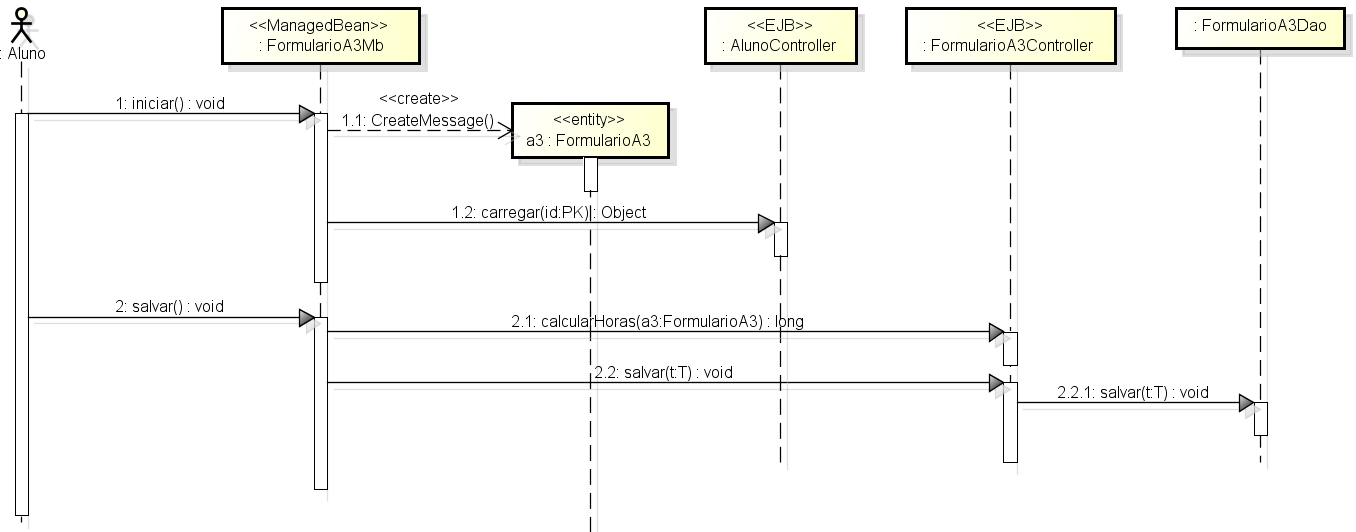
Figura 8 – UC08 – Preencher Formulário A3 – Principais Classes Envolvidas



Fonte: Modelagem do Sistema FormChristus

Na realização desse caso de uso, a partir da página de preenchimento do formulário na camada de apresentação, é acionado o *Managed Bean* FormularioA3Mb, que inicialmente obtém os dados do aluno a partir de seu controlador (AlunoController). Em seguida, uma instância da classe que representa o formulário (FormularioA3) é criada e devolvida à apresentação para preenchimento. Após o preenchimento do formulário, o *Managed Bean* é novamente chamado, repassando os dados ao controlador (FormularioA3Controller) e daí ao DAO para gravação, conforme mostrado na Figura 9.

Figura 9 – Diagrama de Sequência para o UC08 – Preencher Formulário A3



Fonte: Modelagem do Sistema FormChristus

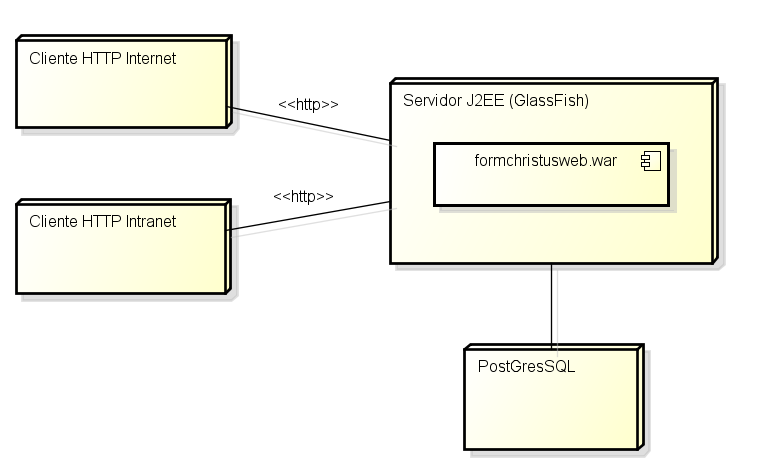
**Visão de Implantação**

A aplicação foi desenvolvida em dois módulos distintos, a saber:

* FormChristusModelo – contém os pacotes com as classes do modelo e as enumerações utilizadas por esse modelo.
* FormChristusWeb – contém a aplicação web propriamente dita com suas devidas configurações. Contempla as configurações de segurança e armazena os arquivos contendo os modelos dos formulários que são gerados. Contém também pacotes com as classes *Managed Bean*, Controladores EJB e DAO.

Durante o processo de compilação e montagem da aplicação, o módulo FormChristusModelo e compilado em um arquivo JAR (*Java Archive*), que é utilizado como biblioteca para o módulo FormChristusWeb. O arquivo WAR (*Web Archive*) resultante é publicado no servidor de aplicações *GlassFish*, ficando assim disponível para utilização pelos usuários. Esse módulo se comunica com um servidor de banco de dados *PostGres* também rodando no servidor. A Figura 10 mostra essa configuração de implantação do sistema.

Figura 10 – Diagrama de Implantação



Fonte: Modelagem do Sistema FormChristus

**Visão de Implementação**

A aplicação possui um único módulo web (formchristusweb.war), publicado no servidor de aplicações conforme descrito no item anterior.

**Componentes Reutilizados**

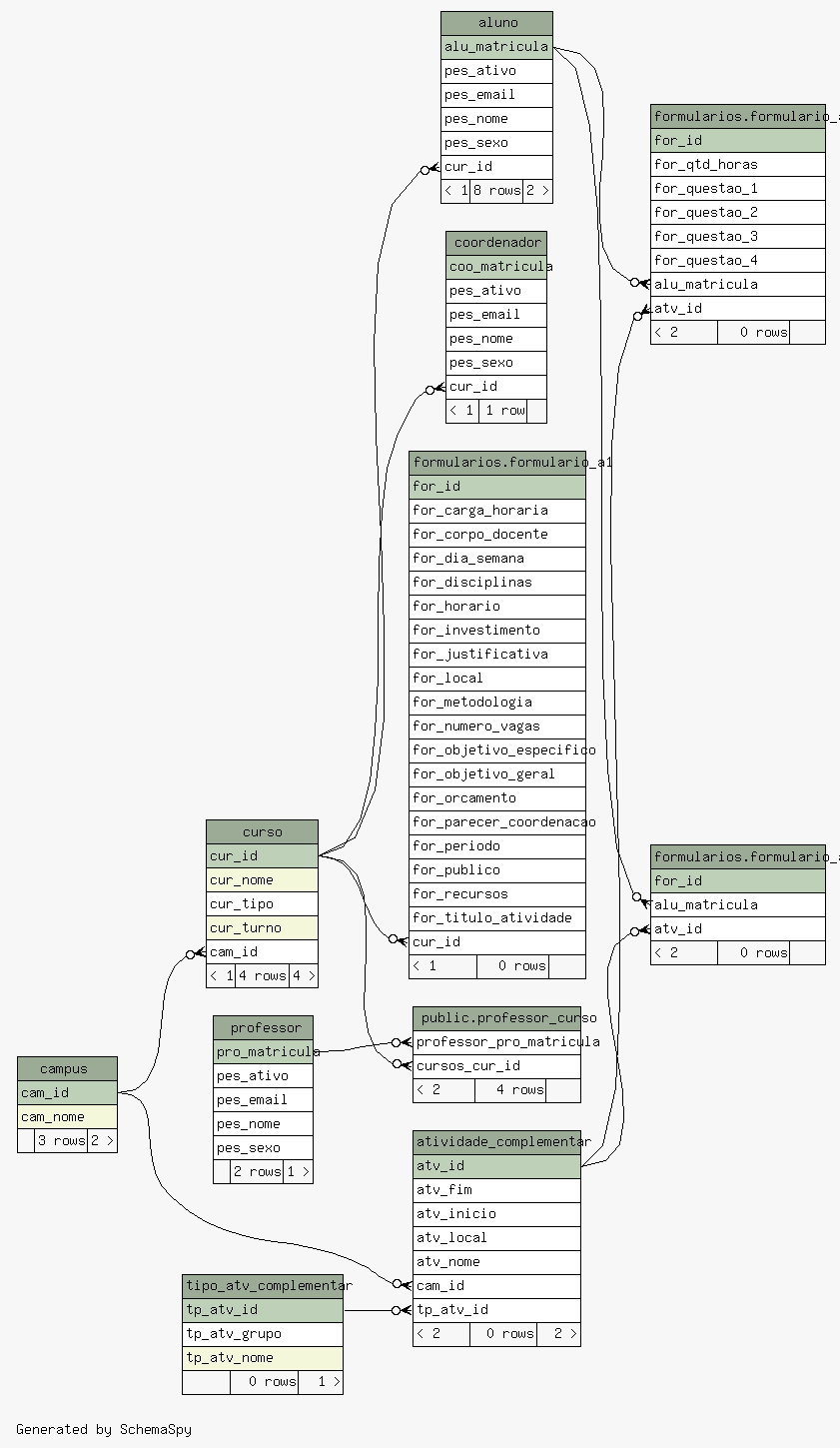
A seguir são listados os componentes básicos utilizados pelo sistema – que são suas dependências. As bibliotecas estão todas disponibilizadas em arquivos JAR. Cada biblioteca também tem suas dependências, que não são listadas aqui.

|  |  |
| --- | --- |
| **Nome do Componente** | **Versão** |
| Primefaces | 4.0 |
| Hibernate | 4.1.2 |
| Hibernate Validator | 4.3.1 |
| Spring Security | 3.0.7 |
| Jasper | 5.1.0 |

**Visão de Dados**

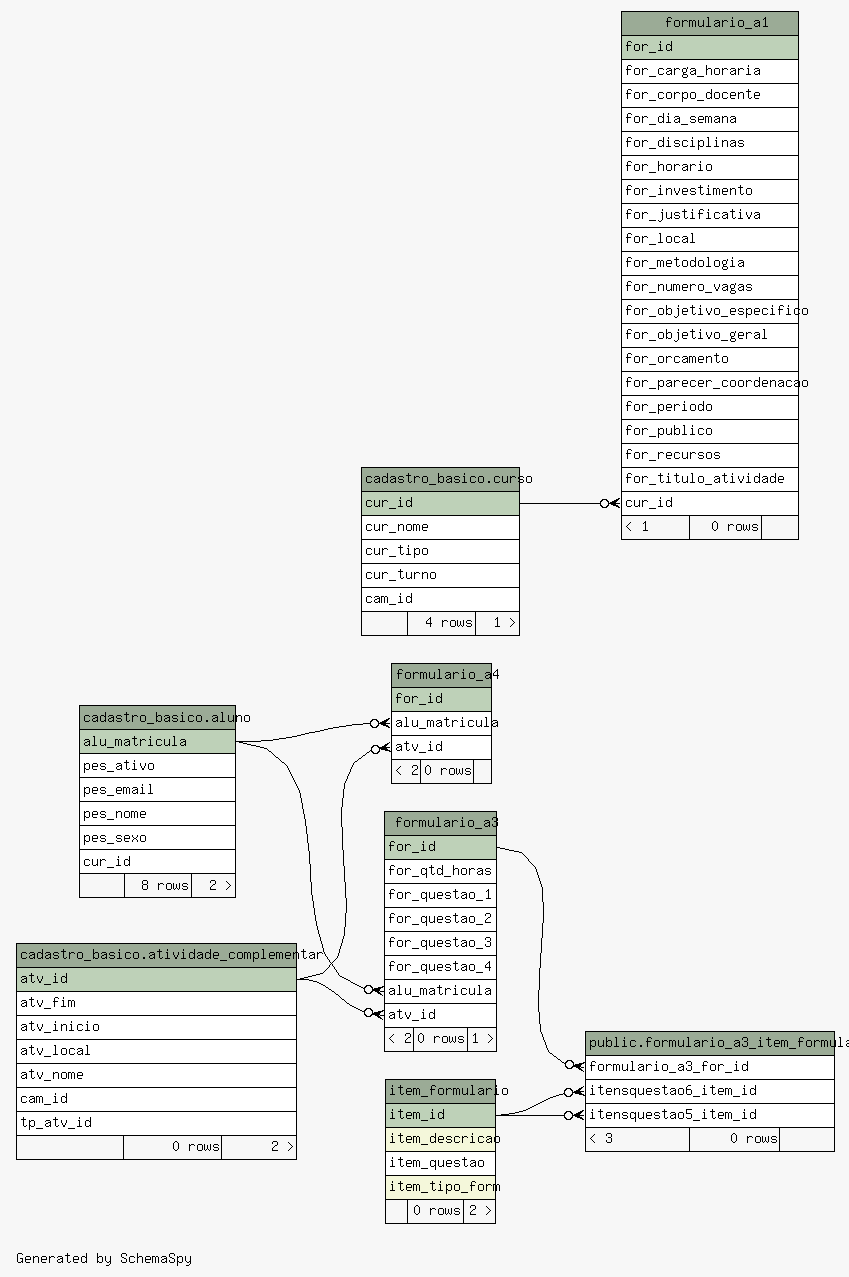
As tabelas da aplicação foram modeladas no banco de dados PostGresSQL, utilizando-se dois esquemas distintos. O primeiro, *cadastro\_basico*, contempla as tabelas contendo as entidades básicas do sistema (aluno, professor, coordenador, formulários, etc.). O segundo, denominado *formulários*, contempla relacionamentos envolvendo os formulários do sistema. As Figuras 11 e 12 apresentam cada um dos esquemas mencionados.

*Figura 11 – Tabelas do esquema cadastro\_basico*



Fonte: Modelagem do Sistema FormChristus

*Figura 12 – Tabelas do esquema formularios*



Fonte: Modelagem do Sistema FormChristus

|  |
| --- |
| **CONCLUSÕES** |

A partir de uma problemática identificada pelos autores no seu dia-a-dia acadêmico no Centro Universitário Unichristus, foi identificada uma oportunidade de pesquisa, materializada no projeto e desenvolvimento de uma aplicação para automatizar o preenchimento de formulários utilizados na instituição.

O projeto se mostrou uma excelente oportunidade para que os autores aperfeiçoassem os conhecimentos adquiridos ao longo do curso de Sistemas de Informação, notadamente nas disciplinas associadas à área de Engenharia de Software (Linguagens de Programação, Análise e Projeto de Sistemas, Engenharia de Software, Banco de Dados, entre outras), através do estudo, modelagem e desenvolvimento de um software para Web, utilizando a linguagem Java e tecnologias correlatas.

A construção do sistema permitiu assim que os participantes, a partir do conhecimento básico visto na faculdade, e adicionando suas próprias experiências de trabalho, pudessem evoluir profissionalmente na área de desenvolvimento de sistemas web, o que sem dúvida agregará importante valor para suas carreiras.

Para a Unichristus, o projeto apresentado se mostra uma oportunidade para melhoria dos seus processos de trabalho, uma vez que sua implementação permitirá um ganho de eficiência no uso dos formulários acadêmicos, através principalmente da redução dos tempos de processamento e da minimização dos erros, além da maior segurança dos processos e a disponibilidade da informação em formato digital, menos sujeita a perdas do que aquelas armazenadas em meio físico.

Embora ainda em versão inicial, o projeto pode ser aperfeiçoado para se tornar ainda mais útil para a instituição. Para tanto, podem ser sugeridas várias funcionalidades adicionais que tornariam o FormChristus ainda mais produtivo e interessante para melhoria dos processos de trabalho da instituição. Dentre esses, pode-se citar:

* Incorporação de novos formulários ao sistema, podendo-se chegar com o tempo a uma total automatização dos formulários utilizados na Unichristus;
* Integração com outros sistemas da instituição, proporcionando ganhos de eficiência e maior confiabilidade dos processos acadêmicos. Nesse sentido, por exemplo, pode-se citar o caso dos formulários associados a Atividades Complementares. Para essa funcionalidade o sistema poderia, por exemplo, manter um registro das horas de atividade complementar de cada aluno, alertando-o em caso de falta de horas em sua carga horária. Outro exemplo seria o controle da frequência das atividades complementares e registro das horas no sistema, incluindo inclusive impressão da lista de frequência das atividades (que não deixa de ser também um formulário);
* Tramitação eletrônica dos formulários, reduzindo a necessidade de impressões e manuseio de formulários em papel. Para isso, poderia ser agregado ao sistema mecanismo de assinatura eletrônica dos envolvidos, de modo que um formulário que necessite ser aprovado ou assinado por outro participante possa trafegar eletronicamente entre as partes e ser assinado eletronicamente, sem a necessidade de ser impresso.

Pode-se perceber que as possibilidades de evolução do sistema são muito grandes, cabendo somente aos envolvidos as ações necessárias para que tais ideias venham a se tornar realidade no futuro.

|  |
| --- |
| **OUTRAS ATIVIDADES**  (congressos, seminários, encontros, etc)  **Os comprovantes deverão ser anexados.** |

O projeto FormChristus foi apresentado no XI Encontro de Iniciação à Docência e X Encontro de Pesquisadores, ocorrido nos dias 13 e 14 de maio de 2014, no Centro Universitário Unichristus. A apresentação pode ser visualizada no endereço eletrônico: http://prezi.com/tnnbea0kw3bx/uma-proposta-para-automatizacao-de-formularios-na-unichristus/?utm\_campaign=share&utm\_medium=copy

|  |
| --- |
| **REFERÊNCIAS BIBLIOGRÁFICAS** |

CAY, Hostmann, GEARY, David. **Core Java Server Faces**. 3ª. Ed. Alta Books, 2012.

COELHO, Hebert. **JSF Eficaz: As melhores práticas para o desenvolvedor web Java**. 1ª. Ed. Casa do Código, 2012.

CORDEIRO, Gilliard. **Aplicações Java para a Web com JSF e JPA**. 1ª. Ed. Casa do Código, 2013.

DEVMEDIA [site]. EJB 3.1 versus CDI. Disponível em: <http:// http://www.devmedia.com.br/ejb-3-1-versus-cdi-revista-java-magazine-105/25040>. Acesso em: 25 mai. 2014.

JACOBI, Jonas e FALLOWS, John. **Pro JSF e Ajax. Construindo Componentes Ricos para a Internet**. 1ª. Ed. Ciência Moderna, 2007.

JASPERSOFT COMMUNITY [site]. Disponível em: <http://community.jaspersoft.com>. Acesso em: 24 mai. 2014.

MASTERTHEBOSS [site]. Using CDI to Manage EJB Dependencies. Disponível em: < http://www.mastertheboss.com/cdi/using-cdi-to-manage-ejb-dependencies>. Acesso em: 27 mai. 2014.

MATHAB RASHEED [site]. What is Java Server Faces, Jsf and Primefaces. Disponível em: <http://mahtabrasheed.wordpress.com/2013/01/25/what-is-java-server-faces-jsf-and-primefaces/>. Acesso em: 25 mai. 2014.

PRIMEFACES[site]. **Why Primefaces**. Disponível em http://www.primefaces.org/whyprimefaces.html. Acesso em 21 ago 2013.

SILVEIRA, Paulo, SILVEIRA, Guilherme, MOREIRA, Guilherme, STEPPAT, Nico e KUNG, Fábio. **Introdução à Arquitetura e Design de Software**. 1ª. Ed. Campus, 2011.

SOUZA, Thiago Hernanes. **Java + Primefaces + IReport. Desenvolvendo um Crud para Web**. 1ª. Ed. Ciência Moderna, 2013.

VMWARE EDUCATION [site]. Hibernate With Spring Training. Disponível em: < http://www.yaw.com.br/open/projetos/cdi-jsf-ejb3/>. Acesso em: 30 mai. 2014.

YAW TECNOLOGIA [site]. Aplicação Web com CDI, JSF e EJB3. Disponível em: < http://docs.spring.io/spring-security/site/docs/3.0.x/reference/introduction.html/>. Acesso em: 26 mai. 2014.

**ASSINATURAS**

|  |
| --- |
| Fortaleza, \_\_\_\_\_\_/\_\_\_\_\_/\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   1. **Professor(a) Orientador(a) Aluno(a)** |

**VISTO DA COORDENAÇÃO DE PESQUISA**

|  |
| --- |
| Parecer da Coordenação:  Fortaleza, \_\_\_\_\_\_/\_\_\_\_\_/\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Coordenação** |

**OBSERVAÇÃO:**

**O relatório final deverá ter, no mínimo, 20 páginas e seguir toda a ordem descrita.**

1. Modelo de Arquitetura de Software que separa a representação da informação da interação do usuário com ele. O modelo (*model*) consiste nos dados da aplicação, regras de negócio, lógica e funções. Uma visão (*view*) pode ser qualquer saída de representação dos dados. O controlador (*controller*) faz a mediação das entradas e saídas, convertendo-as em comandos para o modelo ou a visão. [↑](#footnote-ref-1)