Histórico de Versões

Data	Versão	Descrição	Autor	Revisor	Aprovado por
11/12/2015	1.0	Versão Inicial	Leandro Deitos	Marcelo Sampaio(15/12/2015	
20/05/2016	2.0	Incremento dos módulos da R2.1	Leandro Deitos	Arildo José Guêno	
07/12/2017	3.0	Incremento dos módulos da R3.7	Arildo José Guêno	Cleto May	
07/11/2018	4.0	Revisão do conteúdo	Arildo José Guêno	Cateane Scarpa	
26/03/2019	5.0	Revisão do conteúdo	Arildo José Guêno	Cateane Scarpa	

Índice

Documento de Arquitetura de Software

1. Objetivo do Documento

O Documento de Arquitetura de Software provê uma visão geral da arquitetura através de diferentes tipos de visões para descrever os diferentes aspectos do Sistema de Informações de Departamento Penitenciário Nacional – SISDEPEN, sistema de acompanhamento da execução penal destinado ao cumprimento da Lei n° 12.714/2012, que possibilitará a gestão unificada dos dados referentes à execução das penas, da prisão cautelar e da medida de segurança, mapeando o efetivo cumprimento dos prazos e possibilitando o acompanhamento das informações processuais relevantes ao cálculo da pena pelas instituições integrantes do sistema de Justiça, pela pessoa custodiada/interessados, além de apoiar a gestão das unidades prisionais.

Para tanto, o sistema deverá informar ao Magistrado, ao membro do Ministério Público, ao membro da Defensoria Pública e ao Advogado o implemento dos prazos dos direitos e benefícios em sede de sistema penal, possibilitando, desta forma, grande avanço e modernização no acesso à Justiça. Também deverá prover dados estatísticos para publicação em portal público e contará com funcionalidades de acesso ao cidadão para agendamento de visitas ao sistema prisional.

Tendo em vista que a responsabilidade pela gestão da execução penal é destinada às Unidades da Federação e que essas possuem independência na definição de sua realização, o SISDEPEN deverá permitir que outros sistemas em operação nos estados possam integrar seus dados ao banco nacional; além de também necessitar se integrar a outros sistemas, como por exemplo do Judiciário, Secretaria de Segurança Pública e outros, para obtenção de dados iniciais e/ou complementares.

Nesta versão, o documento restringe-se aos elementos arquiteturais suficientes para atender as necessidades da aplicação até a sexta release (R6), abordando os módulos Gestão Prisional (Gestão de Visitas, Apreensões e Assistências), administrativo (módulos web para a configuração e administração do sistema), segurança (configuração e controle de acesso ao sistema), infopen (coleta de informações de censo prisional) e custodiados (coleta de informações do custodiado).

2. Objetivos e Restrições da Arquitetura

Visando o atendimento do cenário acima descrito, bem como à segurança necessária inerente às características do negócio, a arquitetura proposta tem por objetivo:

1. Apresentar uma aplicação que favoreça a usabilidade e a iteração do usuário com o sistema;
2. Garantir a segurança, a integridade e a confiabilidade dos dados, sejam os informados pela aplicação online, da aplicação offline ou por integração externa;
3. Garantir o não repúdio das informações prestadas, seja de forma online ou recebidas por integração;
4. Permitir a auditoria da manipulação dos dados da aplicação pelos usuários;
5. Disponibilizar mecanismos de controle de acesso às funcionalidades, além de restringir níveis de acesso diferenciados aos dados entre os usuários;

6. Viabilizar o reaproveitamento dos componentes de negócio entre as diversas fontes de entrada de dados que venham a ser necessária (web, webservice, desktop, integração por arquivos, etc);

7. Permitir a escalabilidade na produção da aplicação, em seus diversos aspectos, visto que o universo de usuário é vasto e diversificado;

1. Elementos Arquiteturalmente Significativos

O SISDEPEN utiliza-se de componentes descritos através das APIs que compõem a especificação *JAVA Enterprise Edition* 6[i] (JEE6) e do framework de desenvolvimento *Demoiselle* 2.5 como base arquitetural, sendo construído na linguagem JAVA na versão 1.8 e publicado em servidor de aplicação JBOSS EAP 6.4 ou superior.

Dentre os principais elementos arquiteturais utilizados destacam-se:

- **Apresentação e iteração com usuário:** HTML5, CSS3 e *AngularJS*[ii] (objetivo 1)*;*

- **Camada de serviços REST:** Implementação do JAX-RS disponibilizada pelo servidor de aplicação;

- **Autenticação, controle de acesso e SSO:** Serviço de diretório LDAP (objetivo 3) e componente de segurança próprio (sisdepen-segurança) para definição e controle de níveis de acesso (objetivo 5);

- **Integração e controle transacional entre módulos distribuídos:** Implementação do EJB 3.0 disponibilizada pelo servidor de aplicação (objetivos 6 e 7);

- **API de Persistência e controle transacional**: JPA2.0 e JTA com o uso da implementação disponibilizada pelo *Hibernate ORM*[iii] associado aos componentes demoiselle-jpa e demoiselle-jta, além da extensão *Hibernate Envers*[iv], responsável pela auditoria dos dados inseridos, alterados ou excluídos pelos usuários (objetivos 2 e 3);

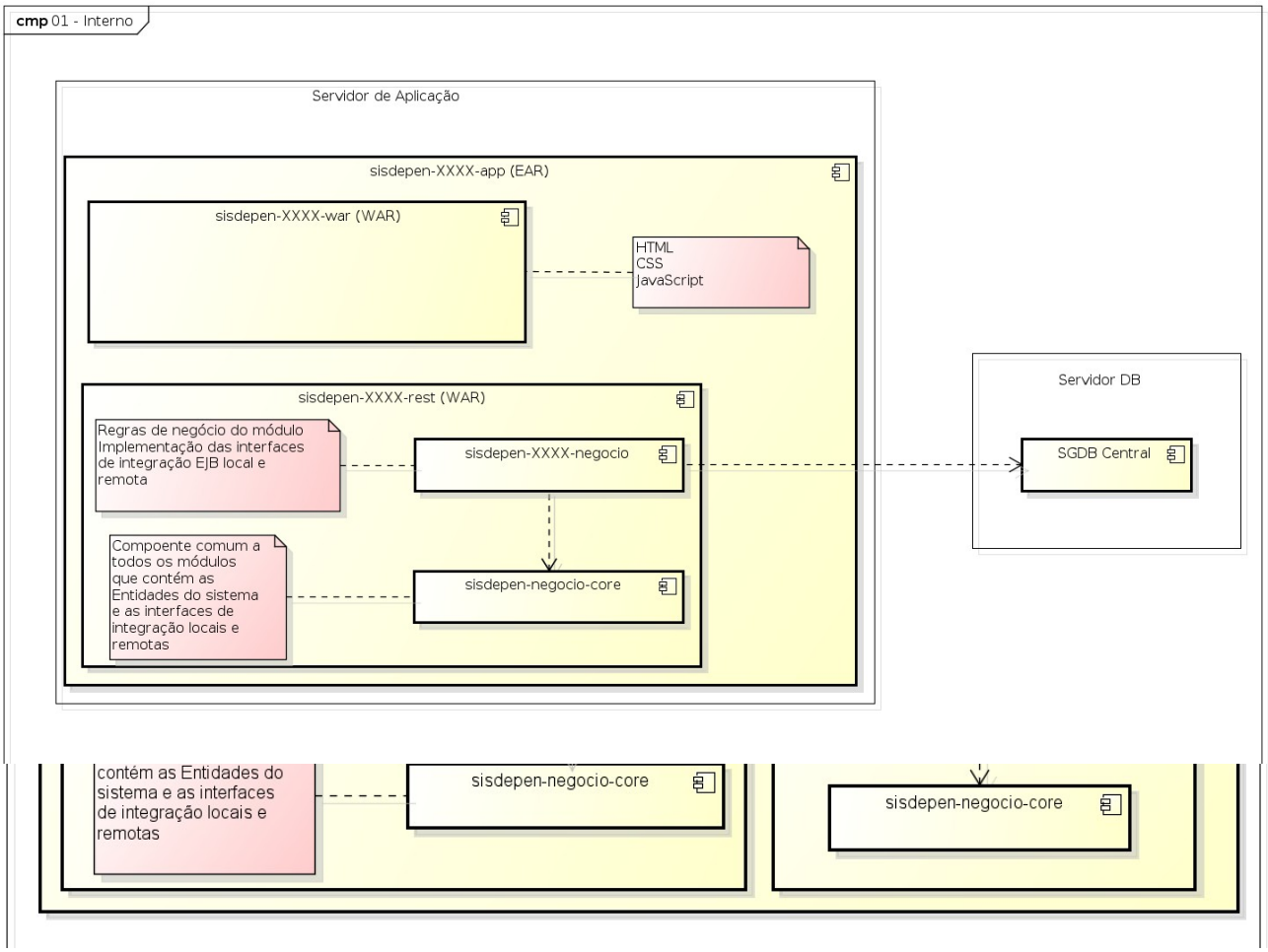
- **API de Validação**: API BeanValidation 1.0 implementada pelo *Hibernate Validator*[v] associado ao componente *ExtendedValidation*[vi]*;*

- ***Mecanismo de busca indexada****: Hibernate Search*[vii]*,* mecanismo de busca indexada que permite busca por aproximação (semelhança) de conteúdo de campos texto*;*

1. Descrição da Arquitetura

O sistema será distribuído sob o empacotamento EAR[viii], conforme figura 1, viabilizando a distribuição dos vários módulos que o comporão em ambientes especialmente dimensionados para atender as necessidades de equilíbrio entre de volume de acesso e a disponibilidade esperada para cada um.

Esta estratégia permite que um ou mais módulos lógicos do sistema sejam implantados em um único ambiente e, caso necessário, seja separado e redistribuído em outro, com impactos e esforço de adequação reduzido, de forma a readequar o desempenho da aplicação frente a mudanças de carga de uso de cada módulo.



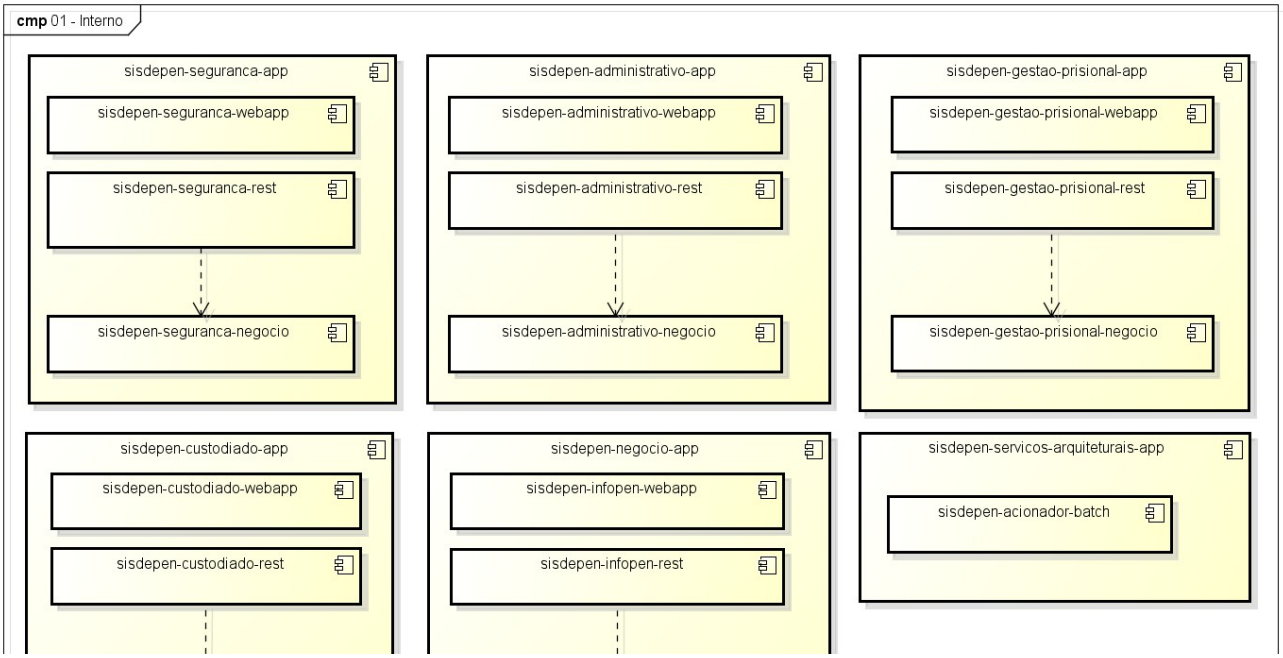
Neste modelo, os componentes de negócio dos módulos não apresentarão interdependência direta, ou seja, nenhum componente de negócio poderá referenciar diretamente outro módulo de negócio, e sim através de interfaces de integração, que serão implementadas pelos respectivos módulos como EJB remoto.

1.1. Camadas e Subsistemas

Os módulos do sistema serão divididos em três projetos, identificados pelos sufixos a saber:

- **webapp:** WAR destinado a empacotar os códigos HTML, CSS e Javascript que comporão a apresentação do sistema;
- **rest:** WAR que publicará os serviços REST do módulo para comunicação com os clientes da aplicação;
- **negocio:** JAR que conterá as classes com as regras de negócio, validação e persistência das entidades do módulo, bem como as implementações das interfaces de integração EJB local e remota.

Conforme a figura 2, a partir da release 2 os módulos estarão empacotados e distribuídos em seis ambientes distintos: segurança, administrativo, negocio, custodiado, gestão prisional e serviços arquiteturais.



Como já descrito, quando um componente de negócio de um módulo necessitar invocar métodos de outro módulo deve fazê-lo através das interfaces de integração do módulo desejado. No caso em que os módulos que se integrem sejam empacotados no mesmo EAR, deve-se utilizar a interface de integração local (Local EJB), enquanto que nos casos em que estejam empacotados em EAR distintos, utilizar-se-á a interface de integração remota (Remote EJB). Desta forma estará garantida, sempre que a operação sendo executada exigir, a integridade transacional entre os módulos distribuídos.

O fluxo de comunicação deve respeitar as camadas do sistema a saber: apresentação (REST, no caso da aplicação WEB) ↔ Fachada do módulo (Facades) ↔ componentes de negócio do módulo (BC) ↔ Persistência (DAO). Este modelo permite que os componentes de negócio sejam reaproveitados para qualquer canal de apresentação (WEB, WebService etc) e que a complexidade exigida para executar as operações de negócio da aplicação seja encapsulada pela fachada de cada funcionalidade.

Na figura 3 é possível ver o fluxo das informações entre as camadas da aplicação em uma transação típica de negócio.

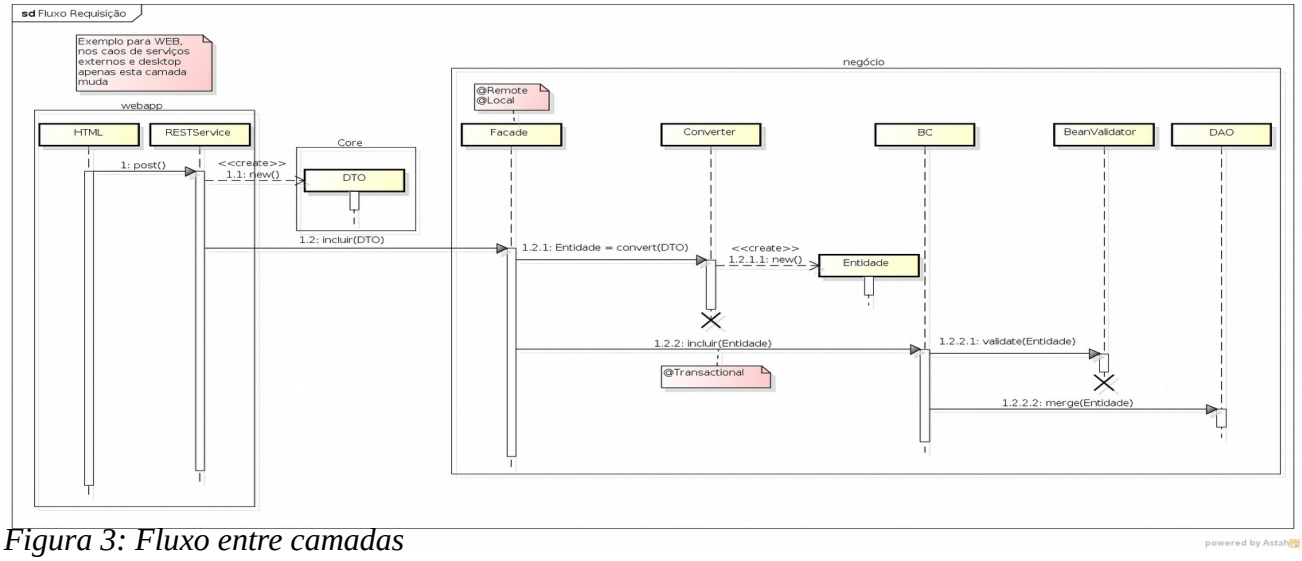


Figura 3: Fluxo entre camadas

A auditora da manipulação dos dados da aplicação pelos usuários será realizada em todas as operações que modificarem informações, como inserção, atualização e exclusão de registros, e será realizada tempestivamente na mesma transação de negócio que originar a mudança, a fim de garantir a integridade da informação de auditoria.

Para este fim, conforme descrito no item 3 deste documento, optou-se pelo uso do componente *Hibernate Envers,* mecanismo que segue o fluxo representado na figura 4.

1.

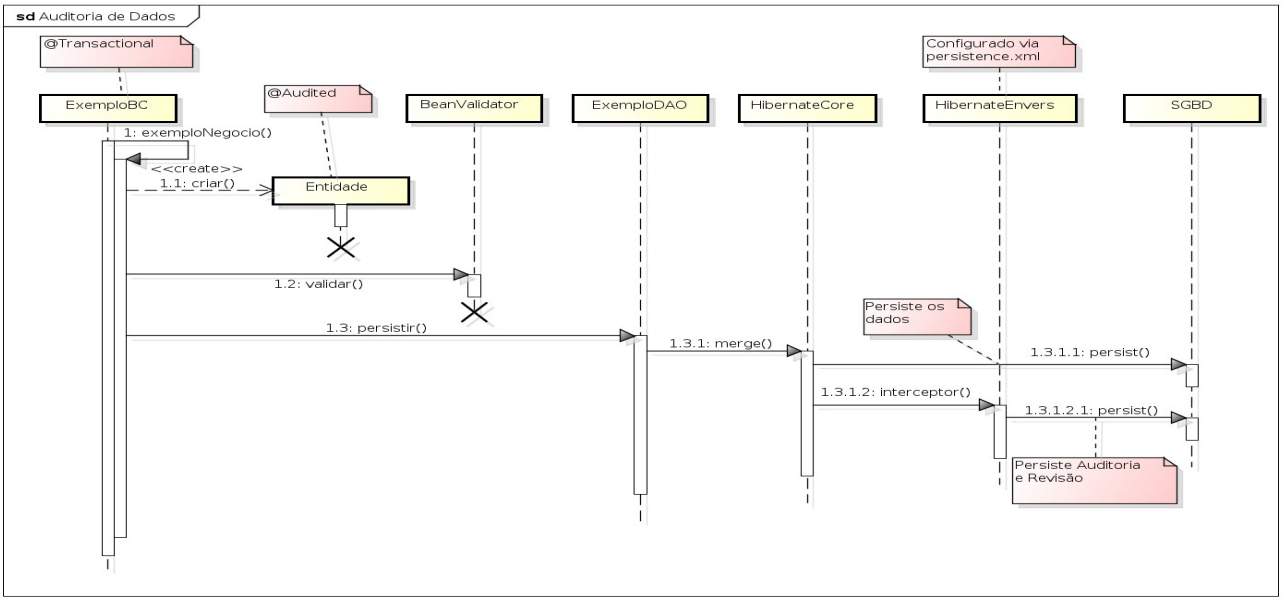
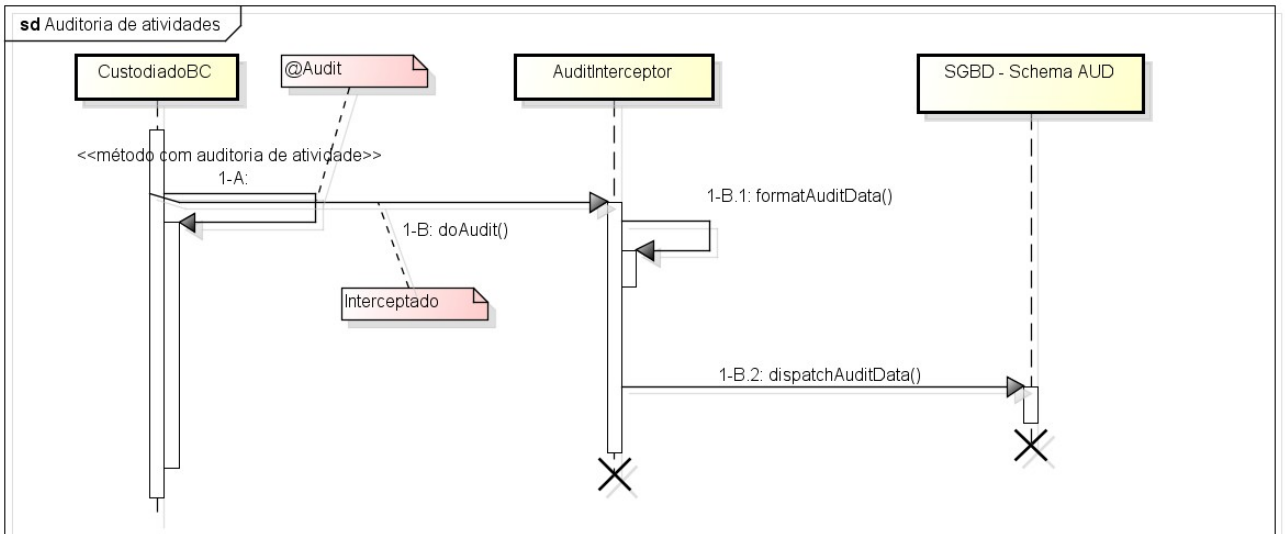


Figura 4: Fluxo da auditoria de dados

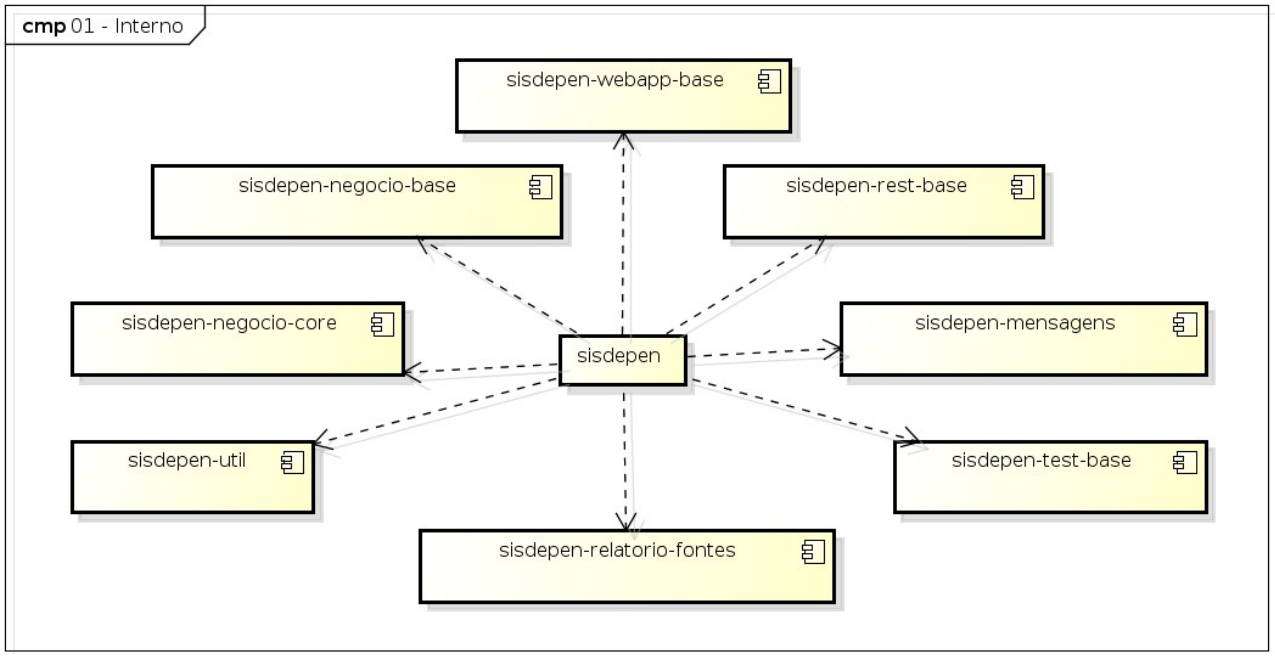
O módulo de custodiados necessita de um mecanismo suplementar de auditoria, a fim de registrar todas as operações de detalhamento. Desta forma será possível rastrear quais usuários tiveram acesso aos dados de custodiados, bem como o momento em que esta operação foi efetuada.

Para este fim, utilizou-se de um componente que realiza a interceptação da execução de um método, extrai e formata os dados informados como parâmetro(s) para posteriormente enviar ao repositório de auditoria, conforme pode ser observado na figura 5.



1.1. Visão de Implementação

Para o atendimento do SISDEPEN foram construídos os componentes arquiteturais descritos na figura 6, os quais servem de base para a implementação dos módulos de negócio já mencionados.



Abaixo segue a descrição da funcionalidade de cada componente:

- **sisdepen-negocio-core:** Destina-se a centralizar todos os elementos comuns e/ou destinados a integração dos módulos, como por exemplo as entidades do sistema, os DTO que representam uma simplificação destas entidades para comunicação com as camadas de apresentação, as interfaces de integração locais ou remotas, etc;

- **sisdepen-negocio-base:** Disponibiliza abstrações e elementos comuns aos projetos de negócio dos módulos;

- **sisdepen-rest-base**: Disponibiliza abstrações e elementos comuns aos projetos que disponibilizam os serviços REST dos módulos;

- **sisdepen-webapp-base**: Disponibiliza elementos JS e HTML comuns aos projetos WEB dos módulos;

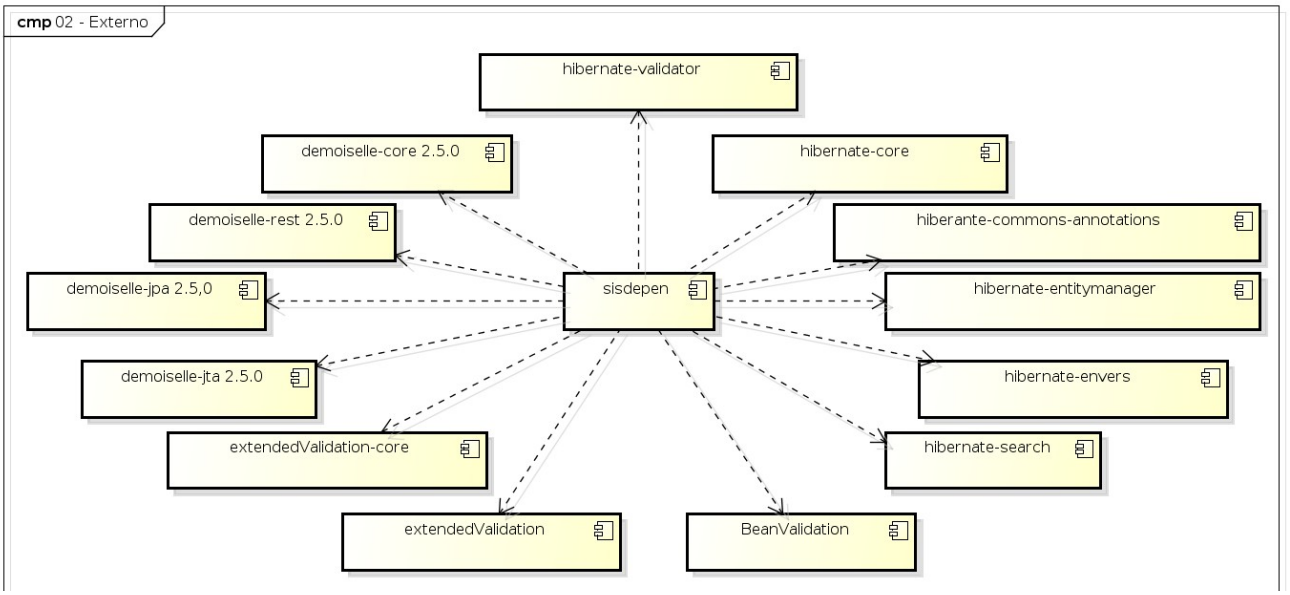
- **sisdepen-util:** Reúne os utilitários de uso comum para o sistema, como mecanismo de log, envio de e-mail e outros.

- **sisdepen-mensagens:** Armazena os arquivos de *properties* que contém as mensagens do sistema, bem como as constantes que apontam para a chave de obtenção destas.

- **sisdepen-relatorio-fontes:** Componente que contém as fontes utilizadas em relatórios.

- **sisdepen-test-base:** Utilitário de suporte para testes unitários da aplicação.

Além dos componentes arquiteturais internos, o sistema utiliza-se de diversas bibliotecas de mercado, cujos principais são apresentados na figura 7.



1.

2. Decisões e Justificativas

O projeto será construído usando como base o *Framework Demoiselle 2.5*, por se tratar de um produto construído pela própria empresa a fim de padronizar o desenvolvimento de sistemas. O *framework* é construído sob a plataforma *JEE* 6.0 e o servidor de aplicações será o *JBOSS EAP 6.4*.

Para aumentar a produtividade e padronizar o código, a interface com o usuário será construída usando *HTML5* em conjunto com o *Framework Bootstrap* (biblioteca de componentes visuais) e o *Framework AngularJS* (responsável pela ligação da tela com os serviços REST).

Devido à necessidade de consulta por similaridade de conteúdo de campos texto do cadastro de custodiado optou-se pelo uso do *Hibernate Search*, devido a performance e flexibilidade de configuração de analisadores de sintaxe de linguagens, possibilitando utilizar múltiplas linguagens como português e inglês, por exemplo.

No módulo de assistências, para facilitar e visualização e agendamento de eventos em calendário, foi utilizada na camada de visualização a biblioteca *FullCalendar*.

[i] http://docs.oracle.com/javaee/6/tutorial/doc/bnaaw.html
[ii] https://angularjs.org/
[iii] http://hibernate.org/orm/
[iv] http://hibernate.org/orm/envers/
[v] http://hibernate.org/validator/
[vi] https://github.com/ldeitos/repository/tree/extendedValidation-1.0
[vii] http://hibernate.org/search/
[viii] https://docs.oracle.com/javaee/6/tutorial/doc/bnaby.html