**Documento de Visão de Projeto**



**E-commerce Sarajevo**

**Histórico de Versões**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** | **Revisor** |
| 10/04/25 | 1.0 | Modelagem e desenvolvimento | Fábio | - |

|  |  |
| --- | --- |
| **Cliente** | FATEC - Interno |
| **Documento** | Documento de Visão de Projeto: *E-commerce Sarajevo* |
| **Data** | 10 de abril de 2025 |
| **Autor** | **Fábio Casagrande**  fabiocasagrande999@gmail.com |

**Página de Assinaturas**

|  |  |  |
| --- | --- | --- |
| Revisado e Aprovado por: |  |  |
|  |  | 27/11/10 |

**Índice**

1. Objetivo 4

1.1 Escopo 4

1.2 Referências 4

2. Necessidades de Negócio 4

Objetivo

Este documento tem como principal finalidade registrar as necessidades de negócio, justificar o desenvolvimento do sistema proposto e apresentar o entendimento atual sobre os requisitos do cliente. Visa descrever de forma resumida o novo sistema de e-commerce de livraria, que deverá satisfazer os requisitos levantados. Além disso, busca alinhar as expectativas dos stakeholders e formalizar o início do projeto, apresentando uma visão geral da arquitetura do sistema.

* 1. Escopo

O escopo deste documento abrange o desenvolvimento de uma aplicação web de e-commerce voltada para a venda de livros. O sistema atenderá tanto clientes quanto administradores, oferecendo funcionalidades como cadastro e busca de livros, gerenciamento de pedidos, controle de estoque, e funcionalidades administrativas. Este módulo será construído utilizando o framework Spring Boot, fazendo parte de uma solução integrada de comércio eletrônico.

Este documento visa ainda descrever os principais aspectos arquiteturais da aplicação, incluindo sua estrutura em camadas, divisão em pacotes e componentes, e as principais decisões técnicas envolvidas no projeto.

* 1. Referências

Para a elaboração deste documento, foram consideradas as seguintes referências:

* Levantamento de requisitos funcionais, não funcionais e regras de negócio definidos previamente.
* Material didático e orientações fornecidas pelo professor da disciplina.

Este documento serve de base para o seguinte artefato:

* Documento de Requisitos

Necessidades de Negócio

Um sistema informatizado de e-commerce voltado para a comercialização de livros é necessário para que livrarias possam expandir sua presença digital, automatizar processos comerciais e oferecer uma experiência de compra eficiente aos seus clientes. Através da solução, será possível gerenciar o catálogo de livros disponíveis, realizar vendas online, acompanhar o status de pedidos e controlar o estoque de forma integrada.

Além disso, o sistema permitirá que administradores tenham acesso a relatórios gerenciais com base nas transações realizadas, auxiliando na tomada de decisões estratégicas e operacionais. Com a informatização dessas rotinas, espera-se melhorar o atendimento ao cliente, reduzir erros manuais e ampliar o alcance comercial da livraria.

Objetivo do Projeto

Desenvolver uma plataforma web de e-commerce especializada na venda de livros, que possibilite:

* Gerenciar o carrinho de compras e realizar pedidos online;
* Controlar o estoque de forma automatizada;
* Permitir o cadastro e gerenciamento de clientes;
* Emitir relatórios de vendas, pedidos e movimentações do sistema;
* Utilizar arquitetura baseada em microsserviços com Spring Boot e persistência de dados em banco relacional.

A solução deverá ser acessível por meio de navegadores modernos, com interface amigável e responsiva, visando proporcionar agilidade nas operações da livraria e uma boa experiência ao usuário final.

Declaração Preliminar de Escopo

Esta seção descreve, em alto nível, o escopo do projeto. Os requisitos serão melhor detalhados nos documentos de Requisitos e Dicionário WBS.

* 1. Descrição

O sistema proposto é uma plataforma web de e-commerce voltada à comercialização de livros. A solução visa atender pequenas e médias livrarias que desejam digitalizar suas operações, oferecendo funcionalidades essenciais para a gestão de produtos, pedidos, usuários e controle de estoque. O sistema será desenvolvido em Java com uso do framework Spring Boot, com foco em desempenho, segurança e escalabilidade.

* 1. Produtos a serem entregues

Os seguintes itens são considerados produtos do projeto, na sua etapa 1:

* Plataforma de e-commerce de livros, etapa 1, implementada de acordo com a especificação elaborada na fase de análise (código objeto e código fonte);
* Documentação de especificação técnica e funcional, elaborada na fase de concepção;
  1. Requisitos
     1. Requisitos Funcionais
* O sistema deve permitir o cadastro, edição, exclusão e visualização de livros no catálogo;
* O sistema deve permitir a realização de pedidos por parte dos clientes;
* O sistema deve gerenciar o status dos pedidos: pendente, em preparação, enviado, entregue;
* O sistema deve permitir o cadastro e autenticação de usuários (clientes e administradores);
* O sistema deve permitir o controle de estoque dos livros;
* O sistema deve gerar relatórios administrativos com dados de vendas, estoque e pedidos;
* O sistema deve possibilitar a busca e filtragem de livros por critérios como título, autor, gênero, preço, etc.

* + 1. Requisitos Não Funcionais
* O sistema deverá ser desenvolvido utilizando a linguagem Java;
* O framework Spring Boot será utilizado na construção do backend da aplicação;
* O banco de dados utilizado será o MySQL;
* A arquitetura do sistema deverá seguir o padrão MVC (Model-View-Controller);
* A interface deverá ser responsiva, adaptando-se a diferentes tamanhos de tela.
* A aplicação deverá ser compatível com os principais navegadores modernos:
* Google Chrome
* Mozilla Firefox
* Microsoft Edge
  + 1. Regras de Negócio
* Um livro só poderá ser comprado se houver estoque disponível;
* Um pedido só poderá ser concluído mediante confirmação de pagamento;
* Os administradores poderão cadastrar, editar e remover livros do catálogo;
* O sistema deve registrar data e hora dos pedidos realizados;
* Clientes só poderão acessar o histórico de pedidos mediante login autenticado;
* Apenas usuários com perfil de administrador poderão acessar os relatórios de vendas e estoque.

Premissas

As premissas descritas abaixo foram consideradas verdadeiras durante a fase de iniciação do projeto e servirão de base para o planejamento detalhado a ser realizado posteriormente. Premissas não documentadas podem comprometer o sucesso do projeto, portanto, este registro inicial é fundamental.

* O projeto será orientado pelo professor Rodrigo Rocha.
* A equipe de desenvolvimento estará disponível para reuniões semanais de alinhamento durante o semestre letivo.
* A aplicação será implementada utilizando a linguagem Java com o framework Spring Boot.
* O banco de dados MySQL será utilizado e disponibilizado para o projeto sem custo adicional.
* O escopo atual do projeto contempla apenas a etapa inicial da plataforma, podendo ser expandido em fases futuras.
* Os interessados no projeto (stakeholders) estarão disponíveis para fornecer feedback e validações parciais durante o processo de desenvolvimento.

Influência das Partes Interessadas

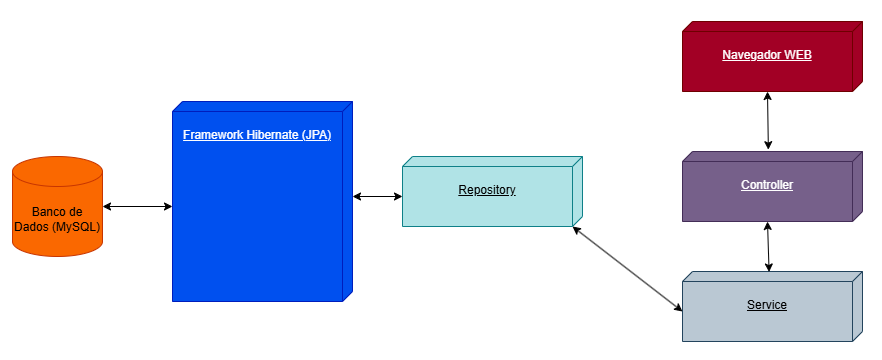
Durante a fase de iniciação do projeto, foram identificadas as seguintes partes interessadas, bem como suas possíveis influências sobre os objetivos, escopo e execução do projeto:

* **Professor Rodrigo Rocha**: orientador do projeto. Possui influência direta sobre a condução das atividades, qualidade da documentação e validação dos marcos do projeto. Seu apoio e direcionamento são fundamentais para o andamento e aprovação do projeto.
* **Fábio Casagrande**: responsável pela análise, projeto, implementação e testes do sistema. Tem interesse em aplicar boas práticas de engenharia de software e garantir a entrega do produto conforme os requisitos.
* **Usuários finais (clientes da livraria)**: esperam uma interface intuitiva, funcionalidade completa no processo de compra e segurança nas transações. Sua aceitação será um dos fatores críticos de sucesso.
* **Administradores da plataforma (donos da livraria)**: interessados na eficiência do sistema para gerenciar produtos, pedidos e relatórios. Sua colaboração é essencial para validações e melhorias nos processos.

A identificação dessas partes interessadas permite antecipar necessidades, alinhar expectativas e definir estratégias para mitigar riscos e fortalecer os fatores críticos de sucesso.

Representação Arquitetural

Os sistemas serão desenvolvidos tendo como base a arquitetura ilustrada na Figura 1. Toda a arquitetura segue padrões tradicionais de projeto definidos pelo GoF, assim como conceitos da arquitetura J2EE, que continuam válidos como referência, mesmo com o uso de tecnologias modernas como SpringBoot.



**Figura 1 - Modelo Arquitetural**

A camada de apresentação será abrigada no Container WEB, e será composta por controladores e páginas que implementam o padrão Model-View-Controller (MVC). Para esta camada, será utilizada uma solução baseada em Spring MVC e templates HTML com Thymeleaf.

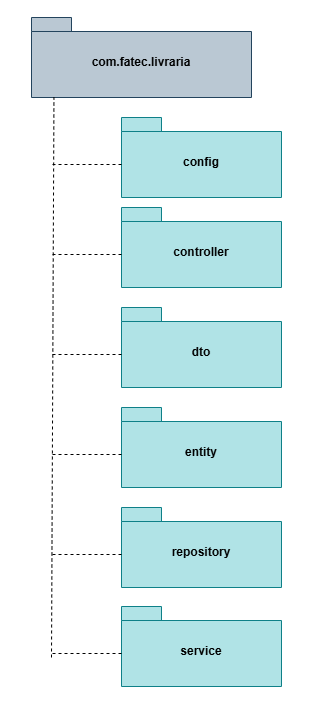
As Classes de Domínio representam as entidades do sistema, contendo atributos e métodos de acesso (getters e setters), que refletem os dados manipulados pela aplicação (como livros, usuários, pedidos, etc).

As Classes de Negócio são responsáveis pelas regras de negócio do sistema e serão implementadas como serviços na arquitetura Spring (annotated com @Service). Estas classes fazem uso de repositórios JPA para realizar operações de persistência no banco de dados.

A camada de persistência será implementada com o Hibernate (JPA), e terá a função de intermediar o acesso ao banco de dados relacional (MySQL), conforme mostrado na figura.

Neste projeto, será adotada uma arquitetura modular baseada no framework Spring Boot, utilizando componentes como @Controller, @Service e @Repository para garantir uma separação clara de responsabilidades entre as camadas da aplicação. A persistência de dados será realizada por meio do Hibernate (JPA), que facilita a comunicação com o banco de dados relacional. Essa estrutura promove organização, manutenibilidade e facilita a escalabilidade do sistema.

O diagrama a seguir, Figura 2, representa a organização das classes dentro dos pacotes da aplicação baseada em Spring Boot. A estrutura foi pensada de forma a garantir a separação de responsabilidades e a facilidade na manutenção do sistema.



**Figura 2 - Diagrama em pacotes**

A aplicação será estruturada com pacotes distintos para as camadas de apresentação (controller), regra de negócio (service), persistência de dados (repository), modelo de dados (model), além de pacotes auxiliares como dto e config.

Essa abordagem elimina a necessidade de diferenciação entre aplicações J2SE e J2EE, bem como o uso de componentes complexos como EJBs. A arquitetura aproveita os recursos nativos do Spring Boot, como a anotação @Service, @Repository, e @RestController, e se alinha aos padrões modernos de desenvolvimento de aplicações Java corporativas.

Essa padronização permite maior reutilização de código, clareza arquitetural e aderência às boas práticas recomendadas pela comunidade Java.

* 1. Restrições Arquiteturais

Foram identificadas algumas orientações e restrições pertinentes ao desenvolvimento deste sistema:

* Utilização do JDK 21 (Java 21) como plataforma base de desenvolvimento;
* Utilização do Spring Boot como framework principal da aplicação, oferecendo suporte completo para desenvolvimento modular, injeção de dependências e controle transacional;
* Utilização do framework Hibernate (JPA) para mapeamento objeto-relacional (ORM);
* Utilização do SGBD MySQL como banco de dados relacional;
* Utilização de fetch API (JavaScript) para realizar requisições assíncronas entre o front-end e a aplicação backend;
* A aplicação será executada em servidores compatíveis com Spring Boot embutido, eliminando a necessidade de servidores de aplicação tradicionais como Tomcat externo ou Oracle Application Server;
  1. Objetivos e Restrições Arquiteturais

Alguns requisitos registrados que impactam diretamente a arquitetura do sistema são:

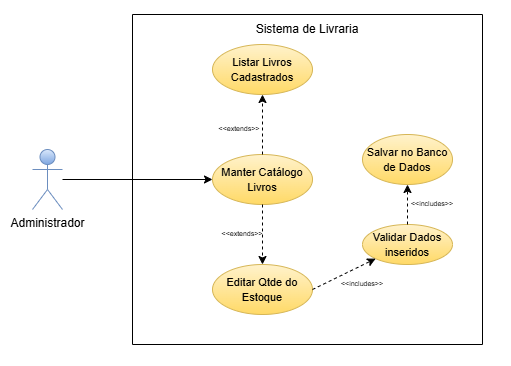
* A necessidade de garantir um modelo de persistência desacoplado, baseado em JPA/Hibernate, facilitando a portabilidade entre diferentes SGBDs;
* A utilização do Spring Boot como plataforma de desenvolvimento principal, devido à sua robustez, flexibilidade e integração com múltiplos módulos (segurança, dados, validação etc.);
* A padronização da linguagem Java em sua versão mais recente e estável (Java 21), com foco em desempenho e segurança;
* A adoção de práticas modernas de desenvolvimento web, com comunicação entre cliente e servidor realizada via requisições assíncronas (fetch/AJAX);
* A preferência por software livre e de código aberto, respeitando as premissas de baixo custo e facilidade de manutenção;
* A utilização do SGBD MySQL como repositório de dados oficial da aplicação.

1. Visão de Use Case

Esta seção apresenta os Casos de Uso arquiteturalmente significativos, que foram selecionados considerando-se o pacote do Modelo de Casos de Uso que representa o sistema da Livraria Virtual.

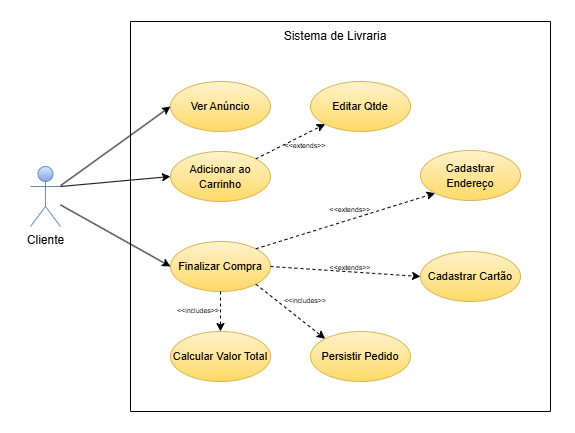
A classificação dos casos de uso, em termos de significância, foi realizada com base na observação de pelo menos um dos seguintes critérios:

* Casos de uso que estendem outros Casos de Uso;
* Casos de Uso que são incluídos em outros Casos de Uso;
* Casos de uso que acessam sistemas externos ou persistem dados importantes.
  1. Diagrama de Casos de Uso Arquiteturalmente Significativos
     1. Diagrama de Casos de Uso Manter Catálogo de Livros



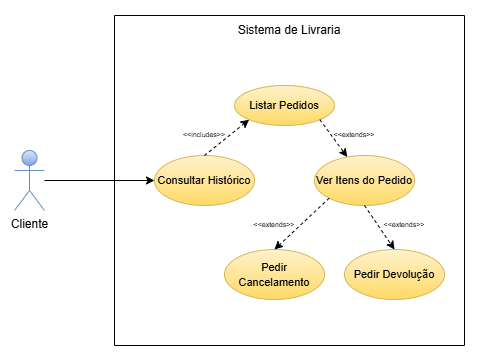
**Figura 3.1 - Caso de Uso de Manter Catálogo de Livros**

* + 1. Diagrama de Casos de Uso Realizar Pedido



**Figura 3.2 - Caso de Uso de Realizar Pedido**

* + 1. Diagrama de Casos de Uso Histórico de Pedidos



**Figura 3.3 - Caso de Uso de** **Histórico de Pedidos**

* 1. Descrição dos Casos de Uso Arquiteturalmente Significativos
     1. Manter Catálogo de Livros

Este caso de uso permite que o administrador cadastre, edite e exclua livros do acervo da livraria. O sistema deve validar os dados inseridos e persistir as alterações no banco de dados.

* + - Ator principal: Administrador
    - Fluxo principal:
      * Administrador acessa o painel de administração.
      * Escolhe a opção "Gerenciar Estoque".
      * Preenche ou edita o campo Estoque.
      * Submete os dados.
      * Sistema valida e persiste no banco via camada Service e Repository.
    - Envolve as camadas: Controller → Service → Repository → Hibernate (JPA) → MySQL
    1. Realizar Pedido

O usuário navega pelo catálogo, adiciona livros ao carrinho e finaliza a compra. Este caso envolve controle de estoque, cálculo de preço e persistência do pedido.

* + - Ator principal: Usuário
    - Fluxo principal:
      * Usuário navega e adiciona livros ao carrinho.
      * Acessa o carrinho e clica em "Finalizar Pedido".
      * Preenche os dados de pagamento e envio.
      * Sistema valida os dados, cria o pedido e atualiza o estoque.
    - Envolve: Controller → Service (com lógica de negócio e cálculo) → Repository
    1. Histórico de Pedidos

Permite que o usuário logado consulte os pedidos realizados, com detalhes como data, status e itens comprados. Pode envolver paginação e filtros.

* + - Ator principal: Usuário
    - Fluxo principal:
      * Usuário acessa "Meus Pedidos".
      * Sistema recupera os pedidos do banco e exibe em ordem cronológica.
    - Pode envolver: chamadas assíncronas via fetch (AJAX), paginação e cache.

1. Visão de Lógica

Esta visão apresenta elementos de design significativos do ponto de vista da arquitetura, descrevendo a organização do Sistema XXXX em pacotes, bem como a organização desses pacotes em camadas.

O Diagrama com as camadas do sistema XXX é ilustrado na figura 5.1.



Figura 5.1 – Diagrama de camadas do XXXX

**Apresentação**: Contém classes para as interfaces gráficas com os usuários (GUI). Através destas interfaces os usuários conseguem interagir com o XXXX, com o intuito de incluir, alterar e excluir produtos.

**Negócio**: Contém classes que controlam a execução das funcionalidades do XXXX.

**Persistência**: Contém classes responsáveis por persistir as entidades de modelo. Por exemplo,

contém as classes que permitem ler e gravar os objetos no banco de dados relacional.

A Figura 5.2 ilustra o diagrama de camadas com as tecnologias utilizadas no desenvolvimento, já descritas na figura 5.1.

Neste momento, é importante ressaltar que a camada de apresentação envolve componentes que são executados na porção servidora e na porção cliente. Na porção servidora, são executados os componentes que montam as páginas html/jsp de resposta e controlam o fluxo de interação com o usuário. Assim, envolve interpretação de páginas jsp, utilização de helpers para montagem das páginas de resposta, delegação de solicitação para a camada de negócio, obtenção de respostas da camada de negócio, dentre outros. Já na porção cliente, estão os códigos javascript que são executados nos navegadores dos usuários com o intuito de facilitar a interação. Por exemplo, podem fazer uma pré-verificação dos dados digitados (como o preenchimento de campos obrigatórios), solicitação de páginas via ajax, de modo a tornar mais dinâmica a interação com o usuário, dentre outros.



Figura 5.2: Camadas do xxx com as dependências de tecnologia

* 1. Camada de Apresentação

Nesta camada, temos o pacote form que contém todos os arquivos relacionados à exibição de informações para usuário, o que engloba páginas JSP e HTML, imagens, javascript, dentre outros. Já o pacote de controle desta camada, contém as Actions do Struts responsáveis pela comunicação com as classes da camada de negócio. Estes pacotes podem ser vistos na Figura 5.3.



Figura 5.3: Camada de Apresentação

## Camada de Negócio

Nesta camada, temos o pacote controle que contém as classes responsáveis por controlar as regras de negócio da aplicação. O pacote model, contém as classes que representam o modelo, ou seja, aquelas que contém as informações sobre o Sistema XXX. O pacote controller contém as classes de controle do negócio. O pacote complementador contém classes que auxiliam na complementação de informações e o pacote filtro contém as classes que auxiliam na filtragem de informações.

A imagem 4.4 ilustra os pacotes descritos.



Figura 5.4: Camada de Negócios

* + 1. Pacote Controller

A figura 5.5 ilustra as principais classes de controle.



Figura 5.5: Classes de controle

* + 1. Pacote Model

A figura 5.6. ilustra as principais classes do modelo.



Figura 5.6: Classes do Modelo

## Camada de Persistência

Nesta camada temos o pacote dao que contém as classes e interfaces responsáveis por persistir as informações do XXXX no BD relacional. O pacote hibernate contido em dao, possui as classes que dependem diretamente do Hibernate, que é o framework utilizado para realizar o mapeamento objeto relacional.



Figura 5.7: Camada de Persistência.

## Realização dos Casos de Uso Significativos



1. Visão de Implantação

Esta seção descreve as configurações da rede física (hardware) na qual o XXX será implantado e executado.

Trata-se de uma visão do Modelo de Implantação que, para a configuração em questão, indica os nós físicos (computadores, CPUs), que executarão o subsistema XXXX, e as respectivas interconexões (barramento, LAN, etc). A figura 6 ilustra o modelo de implantação para o XXXX.



Figura 6: Visão de Implantação do XXXX

Na Figura 6 observa-se os seguintes nós físicos:

* **Web Client Application (WCA):** Aplicativos com interface de usuário via navegador, construídos com base no Framework Struts.
* **Servidor DB**: Nó que contém o BD Central do Sistema XXX.

Descrever demais elmentos

1. Visão de Implementação

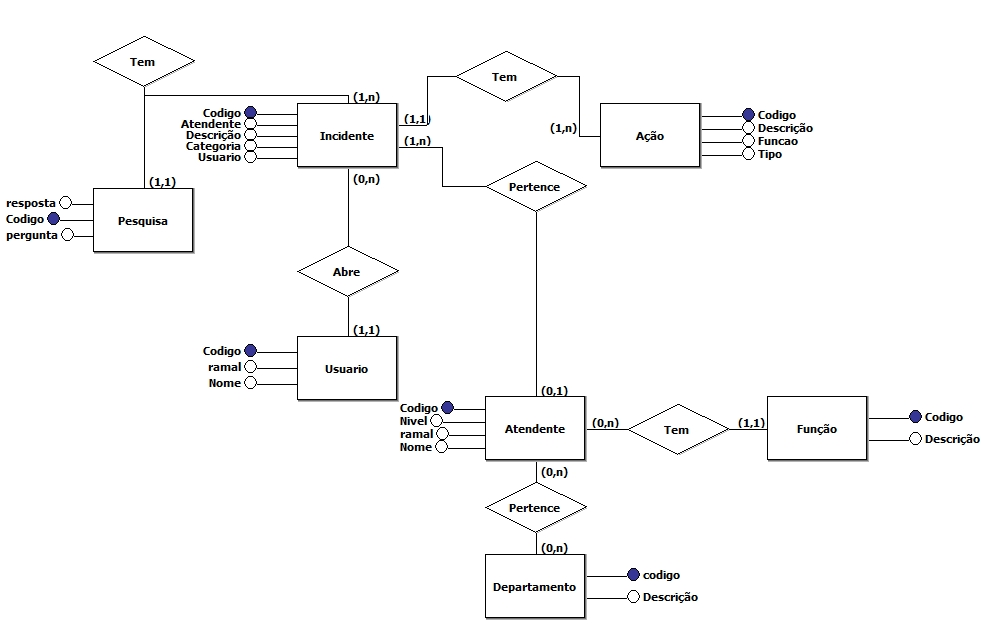
Esta visão descreve a estrutura geral de implementação, a decomposição do software em camadas de implementação.

A estrutura geral de implementação para o SISCAD é baseada na estrutura da Visão Lógica, assim, não há necessidade de detalhar os diagramas de camadas e pacotes de implementação, uma vez que são fortemente baseados naqueles desenvolvidos para Visão Lógica.

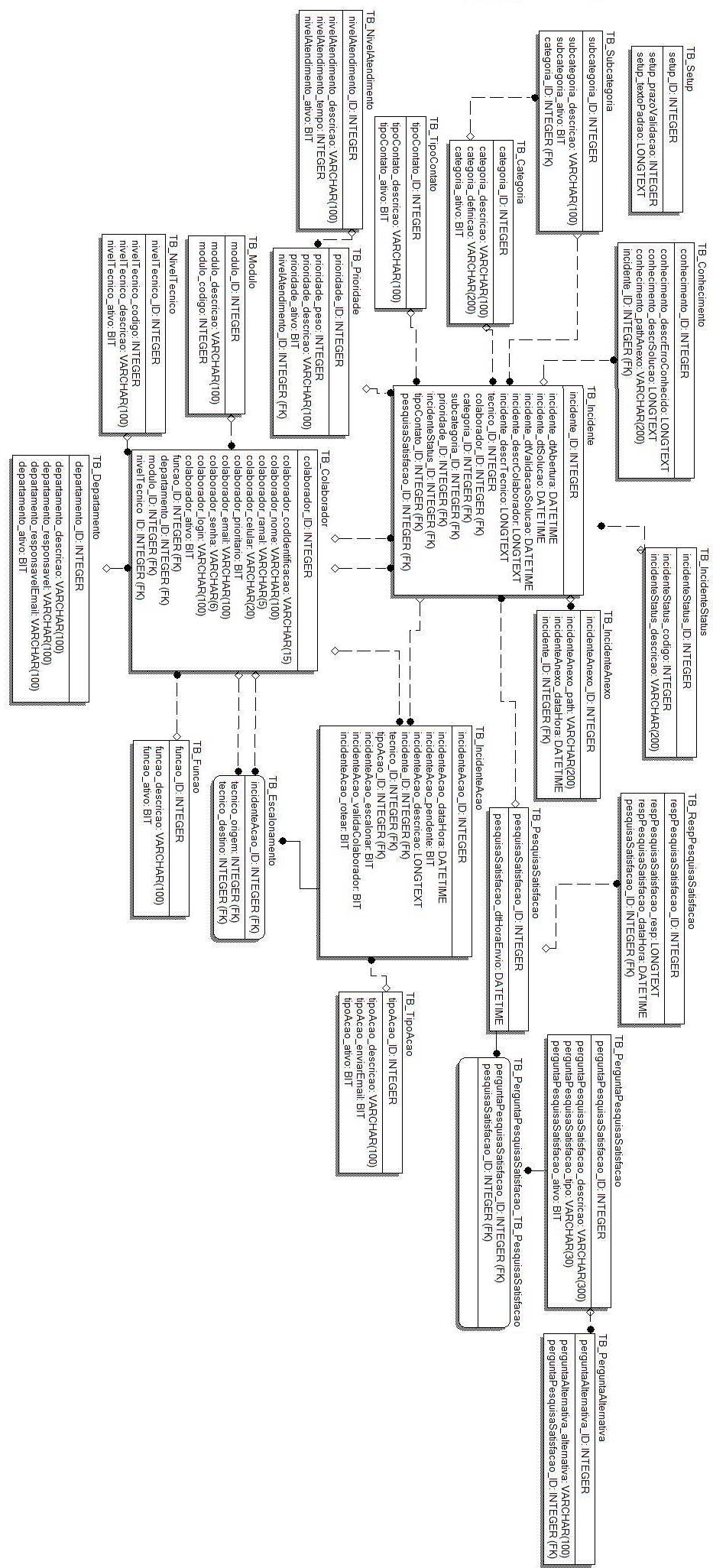
1. Visão de Dados

O mecanismo de persistência utilizado no sistema SISCAD utiliza-se o banco de dados Relacional Oracle juntamente com o framework para mapeamento objeto-relacional, Hibernate. O controle de transações adotado envolve a utilização do Spring Framework em conjunto com o Hibernate.

As figuras XXX e YY, apresentam a visão lógica e física da base de dados do SISCAD.

****

Figrua XXX – Modelo Lógico



Figrua XXX – Modelo Físico

A Tabela 2 define o mapeamento das principais classes de modelo para entidades do modelo lógico do BD Oracle.

Note que existem alguns campos nas entidades lógicas do BD que não estão mapeadas diretamente com as classes de modelo da Visão Lógica contidas neste documento:

Os campos que começam com “IN” são campos definidos através de domínios do BD. As classes de modelo que possuem os atributos que representam tais campos não constam neste documento, mas podem ser consultadas no documento [REF – Modelo de Design].

Os campos #NRSQ e DT REGISTRO presentes em todas as entidades lógicas do BD, correspondem a atributos presentes em superclasses das classes de modelo, que não constam neste documento, mas podem ser consultadas no documento [REF – Modelo de Design].

Tabela 2: Mapeamento Objeto-Relacional

|  |  |
| --- | --- |
| **Classe** | **Entidade** |
|  |  |
|  |  |

1. Tamanho e Performance

O sistema XXXX será usado para o controle do comércio exterior e conseqüentemente terá uma grande base.

Seus servidores provavelmente irão passar por períodos de picos de utilização (por exemplo, próximo a finalização de prazos de entrega de documentos e/ou procedimentos fiscais, tal como ocorre com os servidores que recebem declarações de imposto de renda).

As estimativas do número de usuários e de carga de utilização em períodos de pico de utilização, bem como maiores informações sobre questões relacionadas ao tamanho e desempenho do sistema XXX podem ser obtidas no documento de requisitos não funcionais.

1. Qualidade

O sistema XXX será usado para o controle do comércio exterior brasileiro, conseqüentemente tratando de altos volumes financeiros e um grande número de operações de importação/ exportação diariamente.

Eventuais erros e/ou falhas na sua operação podem levar a prejuízos significativos tanto em termos financeiros quanto na imagem da UMC, portanto na fase de design deve-se levar em consideração como fatores prioritários a confiabilidade e robustez do sistema.

Adicionalmente, o sistema XXX pode ser alvo de ataques de “hackers” para roubar ou simplesmente corromper informações, possibilidade aumentada pela interface do sistema disponível na Internet, para evitar que tais ataques sejam bem sucedidos uma infra-estrutura de segurança deve ser especificada e projetada.

Maiores informações sobre questões relacionadas aos requisitos de qualidade do sistema XXXX podem ser obtidas no documento de requisitos não funcionais.

1. Cronograma Macro.

Os prazos podem ser dados em semanas ou meses dentro do projeto ou ate mesmo data.

|  |  |
| --- | --- |
| **Resultado** |  |
| Plano Preliminar | Semana 2 |
| Plano Fase 1 | Semana 3 |
| Especificação Fase 1 | Semana 5 |
| Piloto Fase 1 | Semana 11 |
| Solução Testada Fase 1 | Semana 12 |
| Plano Fase 2 | Semana 13 |
| Especificação Fase 2 | Semana 14 |
| Piloto Fase 2 | Semana 20 |
| Solução Testada Fase 2 | Semana 21 |

Obs: Os prazos apresentados são uma estimativa inicial considerando as informações disponíveis nesta etapa do projeto. Um cronograma detalhado será elaborado na fase de planejamento e, eventualmente, estes prazos podem ser modificados.

1. Referências

Unified Modeling Language: <http://www.omg.org/technology/documents/formal/uml.htm>

RUP. Rational Unified Process.