# Sistema de Prevenção de Ações Jurídicas (Sipaju)

Este projeto está organizado de modo a propiciar o desenvolvimento de méto- dos, técnicas e ferramentas baseadas em inteligência computacional e aprendizagem de máquina para analisar e identificar a predisposição do consumidor de energia quanto à proposição de ações judiciais, e fornecer subsídios para as áreas gerenciais das distribui- doras atuarem preventivamente na promoção de melhorias.

A partir da base histórica de ações judiciais (no sistema CP-PRO), é possível integrar as outras dimensões da informação que estão presentes em sistemas separados da empresa: sistema comercial (SAP CCS), sistema da operação (OPER) e sistema de atendimento (CRM). As outras dimensões que podem ser analisadas são por exemplo: histórico de faturamento, histórico de consumo, histórico de reclamações, histórico de reclamações de interrupção no fornecimento de energia, tempo de atendimento, localização do cliente, entre outras.

Pretende-se construir ferramentas que automatizem a análise dessa massa de dados buscando encontrar causas semelhantes, mapeando os cenários, usando como base os métodos e técnicas de inteligência artificial. Assim, a partir dos dados históricos, pretende-

se buscar causas comuns que estão em cenário semelhantes para que a empresa possa atuar preventivamente junto ao cliente evitando a insatisfação e abertura de processo jurídico.

Baseado nos requisitos citados anteriormente, a solução computacional terá três módulos/etapas que seguem o processo de desenvolvimento de descoberta de conhecimento em bases de dados (KDD - Knowledge Discovery in Database). ([PIATESKI; FRAWLEY](#_bookmark89), [1991](#_bookmark89))(Figura [1](#_bookmark2)):

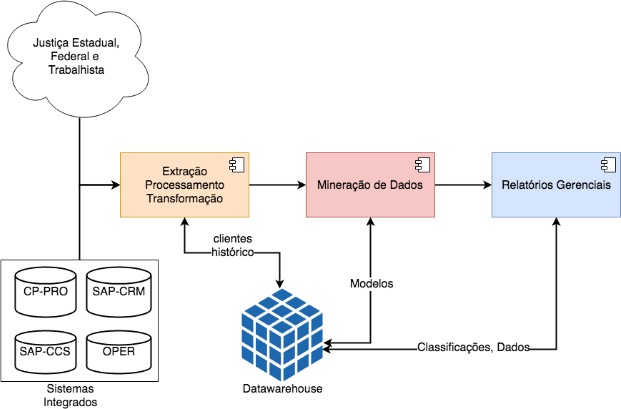


Figura 1 – Representação arquitetural da integração dos três módulos da ferramenta pro- posta

Os módulos/etapas previstos para serem implementados na solução computacional proposta são:

* Módulo I - Seleção, Pré-Processamento e Transformação dos dados: Este módulo terá a finalidade de extrair os dados históricos dos diversos sistemas via um modelo de integração de base de dados a ser implementado, aplicar algoritmos de sumari- zação dos dados descritivos, realizar a limpeza dos dados e selecionar as melhores informações com uso de métodos estatísticos. E, quando houver necessidade, trans- formar os dados para que possam ser utilizados em métodos de aprendizado de máquina.
* Módulo II - Mineração, Análise e Classificação de Consumidores: Este módulo aplica os algoritmos de descoberta de padrões para extrair conhecimento dos dados. A partir dos resultados, interpreta os dados para validar os resultados gerados pelos métodos, avaliando se os resultados poderão ser utilizados na ferramenta de apoio ao processo de tomada de decisão. Finalmente, o módulo classifica cada cliente em dois grupos com relação ao risco de proposição de ações judiciais: Grupo I - Risco de proposição; Grupo II - Sem risco de proposição. Os clientes classificados no Grupo

I serão categorizados de acordo com o macro assunto da sua insatisfação na relação consumerista que pode ensejar o possível processo judicial.

* Módulo III - Relatórios Gerenciais: Módulo responsável pela apresentação das in- formações resultantes do processo, para possibilitar que o setor jurídico visualize os clientes categorizados em macro assunto de risco, e viabilizar que cada área espe- cífica da empresa possa iniciar um plano de ação visando melhorar a satisfação do clientes e evitando, consequentemente, a abertura do processo judicial.

As vantagens pretendidas com a proposta consistem na redução da quantidade de ações judiciais iniciadas contra a companhia; redução dos custos associados às resposta necessárias às ações judiciais e aumento da satisfação do cliente.

Este relatório trata das atividades realizadas no mês 03 das etapas 01, 02 e 03 previstas no projeto. Neste mês foram executadas atividades referentes à modelagem da arquitetura da proposta, protótipo de interfaces e experimentos de inteligência computa- cional com dados relacionados ao problema numa base temporal.

Nesta seção do relatório apresentamos a especificação do Sistema de Prevenção de Ações Jurídicas - Sipaju em termos da Engenharia de Software. Serão apresentados o refinamento dos requisitos, o plano de entregas e a visão arquitetural da solução. Essa seção consiste numa continuação da definição realizada no âmbito do relatório do mês 2 e tem por objetivo ampliar o conhecimento e abstração acerca da solução a ser implementada.

## Decisões chave para a arquitetura

Esta seção descreve as principais decisões tomadas pela equipe de desenvolvimento do projeto, as quais irão nortear o seu desenvolvimento. Todas as decisões são justificadas e avaliadas de acordo com os seus impactos positivos, negativos e drivers, que são os motivos norteadores (normalmente requisitos).

|  |  |
| --- | --- |
| **Decisão Chave**: | Usar uma rede de componentes interconectados |
| **Drivers** | Requisitos funcionais de aquisição de dados e visualização. |
| **Descrição** | Utilização de componentes em todos os níveis do sistema  para coleta, transformação, tratamento, apresentação e for- matação da informação. |
| **Vantagens** | 1. Flexibilidade, pois o desenvolvimento utilizando o pa-   radigma orientado a componentes permite a substituição e a adição de novos componentes ao sistema. Até mesmo, através da combinação de componentes existentes podem- se gerar novas funcionalidades.   1. Manutenção, uma vez que os componentes possuam in- terfaces bem definidas, o sistema ficará desacoplado e o im- pacto de mudanças fica restrito a uma quantidade mínima de componentes |
| **Desvantagens** | Complexidade, pois as soluções baseadas neste paradigma  tendem ser mais complexas do que as soluções monolíticas. |
| **Observações** | Os componentes se integram utilizando o modelo baseado  em camadas. Consideramos como componente “um pacote coerente de implementação de software que pode ser desen- volvido independentemente e entregue como unidade; tem interfaces explícitas e bem definidas para os serviços que oferece; tem interfaces explícitas e bem definidas para os serviços que requer; e pode ser composto com outros com- ponentes, após a customização de algumas propriedades, mas sem modificar os componentes em si. |

|  |  |
| --- | --- |
| **Decisão Chave**: | Usar arquitetura em camadas |
| **Drivers** | Requisitos não funcionais de interface. Qualidade de código  elevada. |
| **Descrição** | Modelar a arquitetura do Sipaju utilizando o modelo base-  ado em camadas. |
| **Vantagens** | Fornecer um maior nível de flexibilidade, pois através deste  paradigma, os componentes ou módulos de uma determi- nada camada só poderão se comunicar de acordo com uma política baseada em fachadas. Por exemplo, separação en- tre a lógica de apresentação (interface com o usuário) e a lógica de negócio. Permitir maior independência entre par- tes do código. Com a separação em camadas, a maioria dos efeitos de manutenção de código dentro das camadas surtirá efeito apenas dentro de uma camada. |
| **Desvantagens** | Complexidade adicionada devido à organização do código  em camadas. |
| **Observações** | Nenhuma |

|  |  |
| --- | --- |
| **Decisão Chave**: | Organizar a arquitetura em camadas usando o padrão  Model-View-Controller (MVC) |
| **Drivers** | Desacoplamento, qualidade de código e facilidade de ma-  nutenção. |
| **Descrição** | Estabelecer um modelo de arquitetura baseado no padrão  MVC, com 3 grandes camadas: (1) View: responsável pela parte de apresentação da informação para os usuários. (2) Model: trata com o estado da informação (persistência), esta grande camada será representada pelas subcamadas de lógica de negócio e pela camada de aquisição de dados. (3) Controller: gerencia o fluxo de navegação do usuário pelo sistema, além de acessar o estado da informação armaze- nada no Model para apresentação ao usuário, esta camada será formada pela camada de lógica de negócio. |
| **Vantagens** | Maior flexibilidade, menor acoplamento no código, maior  facilidade de manutenção. |
| **Desvantagens** | Complexidade adicionada devido à organização do código  em camadas. |
| **Observações** | Decisão implementada usando o .NET framework que  ajuda a separar o código no modelo MVC. |

|  |  |
| --- | --- |
| **Decisão Chave**: | Utilizar o .NET framework para desenvolvimento da arqui-  tetura em camadas MVC |
| **Drivers** | Qualidade de código e facilidade de manutenção |
| **Descrição** | Aplicação de um framework para desenvolvimento do có-  digo de cada camada de aplicação. |
| **Vantagens** | Desacoplamento do código produzido, já que os frameworks  permitem isso. Aumentar a facilidade de manutenção do código. Aumentar quantidade de código testado, pois os frameworks já possuem uma bagagem considerável de tes- tes. Aumentar a produtividade da equipe. |
| **Desvantagens** | Suporte limitado ao mapeamento Objeto Relacional. Su-  porte geralmente apenas ao Entity Framework. Desenvol- vimento futuro do Framework dependerá da Microsoft. |
| **Observações** | O .NET é o framework de desenvolvimento homologado  pela TI da Equatorial. |

|  |  |
| --- | --- |
| **Decisão Chave**: | Utilizar o SGBD Oracle 12c e componentes especiais de  persistência para garantir a persistência dos dados |
| **Drivers** | Requisitos funcionais de aquisição de dados e visualização.  Requisitos não funcionais de integração (SGBD, Sistema- sExternos). |
| **Descrição** | Desenvolvimento de componentes e módulos responsável  pela persistência das informações coletadas/tratadas em SGBD Oracle. |
| **Vantagens** | O Oracle é o SGBD de maior confiança do mercado atual,  além de permitir maior segurança, confiabilidade e desem- penho para os seus usuários. Além de ser o SGBD utilizado e homologado pela TI da Equatorial |
| **Desvantagens** | Complexidade arquitetural. Requer conhecimento especia-  lizado para projetar a arquitetura e o banco. |
| **Observações** |  |

|  |  |
| --- | --- |
| **Decisão Chave**: | Coletar Dados dos sistemas a serem integrados via APIs,  WebService e Eqtl.Info |
| **Drivers** | Requisitos funcionais de aquisição de dados e visualização.  Requisitos não funcionais de integração (SGBD, Sistemas Externos). |
| **Descrição** | Desenvolvimento de componentes e módulos que tratarão  a integração com a plataforma SAP via serviços, API’s e SGBD Oracle além do Eqtl.Info. |
| **Vantagens** | Reutilização de implementações existentes no ambiente de  produção. |
| **Desvantagens** |  |
| **Observações** |  |

|  |  |
| --- | --- |
| **Decisão Chave**: | Utilizar uma *thread* por fonte de informação |
| **Drivers** | Requisitos funcionais de obtenção de dados e Requisitos  não funcionais de integração. |
| **Descrição** | Manter uma thread para cada componente de aquisição de  dados. |
| **Vantagens** | Flexibilidade. Independência no processo de carga das fon-  tes de dados. Maior robustez. Maior tolerância a falhas. |
| **Desvantagens** | Softwares *multithreaded* são inerentemente mais complexos  que software que não utilizam esse tipo de recurso |
| **Observações** | Devido a sua dificuldade, isto requer o desenvolvimento  de código para gerenciar o funcionamento das *threads* que executarão as coleta nas diversas fontes de dados |

|  |  |
| --- | --- |
| **Decisão Chave**: | Usar tecnologias de código aberto para o desenvolvimento  dos mapas digitais |
| **Drivers** | Requisitos funcionais de geração de relatórios espaciais. |
| **Descrição** | Utilização de tecnologias de código aberto para implemen-  tação das visões que apresentam mapas digitais na web. |
| **Vantagens** | Desenvolvimento rápido de interfaces ricas para os aplica-  tivos de mapas. Aumento do desempenho das aplicações com estas tecnologias, já que elas acarretam um tempo de carga inicial, porém nas próximas iterações como o soft- ware, o usuário trabalha com as aplicações em tempo real. |
| **Desvantagens** |  |
| **Observações** |  |

|  |  |
| --- | --- |
| **Decisão Chave**: | Desenvolver um módulo para apresentação de informações  através de gráficos |
| **Drivers** | Qualidade de código, Flexibilidade e facilidade de manu-  tenção. |
| **Descrição** | Utilizar um módulo que gere gráficos e apresente para os  usuários a informação já processada. |
| **Vantagens** | Desacoplar o processamento dos gráficos do restante do  processamento dos dados. Flexibilidade para trocar a tec- nologia geradora de gráficos. Fácil manutenção, pois este é um módulo que tende a mudar bastante com as solicitações dos usuários e as mudanças tendem a impactar menos no restante do código de processamento da informação. |
| **Desvantagens** |  |
| **Observações** | Utilizar ferramentas livres como Chart.js |

|  |  |
| --- | --- |
| **Decisão Chave**: | Utilizar o Windows Server com IIS como servidor |
| **Drivers** | Requisitos não funcionais de custo (Hardware e Software  Básico). |
| **Descrição** | O ambiente servidor que se espera para o Sipaju executar  é o Windows e o servidor de aplicações IIS. |
| **Vantagens** | Baixo custo das licenças, alta confiabilidade destas aplica-  ções, softwares bastante testados e robustos. |
| **Desvantagens** |  |
| **Observações** | Ambiente homologado pela TI da Equatorial |

|  |  |
| --- | --- |
| **Decisão Chave**: | Usar framework de teste de unidade |
| **Drivers** | Qualidade de código, requisitos não funcionais de quali-  dade. |
| **Descrição** | Utilização do framework NUnit pelos desenvolvedores e tes-  tadores para implementação dos testes de unidade e do Se- lenium para teste de interface. |
| **Vantagens** | 1. É uma ferramenta de teste de unidade livre, open source, centrada na comunidade para o .NET Framework. 2. Melhor qualidade de código produzido, garantia da ve- rificação do código produzido pelos desenvolvedores. |
| **Desvantagens** | Aumento no tempo de desenvolvimento |
| **Observações** | Além dos testes de aceitação, os desenvolvedores e testado-  res devem especificar e implementar testes de unidade para todo o código produzido para implementar as funcionali- dades do Sipaju |

|  |  |
| --- | --- |
| **Decisão Chave**: | Implementar aspectos de segurança com recursos apropri-  ados do framework de desenvolvimento |
| **Drivers** | Requisitos funcionais e não funcionais de segurança, quali-  dade de código. |
| **Descrição** | Utilizar o suporte à segurança que o framework escolhido  provê, obedecendo os mecanismos de autenticação utiliza- dos pela empresa. |
| **Vantagens** | Reuso de código do framework. Segurança implementada  com código bem testado. |
| **Desvantagens** |  |
| **Observações** |  |

## Refinamento de Requisitos

Em especificação anterior, foram apresentados requisitos de negócio, funcionais, não funcionais e casos de uso. Especificamente os requisitos funcionais levantados são apresentados na Tabela [3](#_bookmark9).

Tabela 3 – Requisitos Funcionais do Sipaju

|  |  |
| --- | --- |
| **Requisitos**: | **Descrição** |
| RF-1 | Extrair informações dos clientes da Equatorial, localizado em múlti-  plas bases e mantê-los de maneira estruturada |
| RF-2 | Predizer o risco judicial de cada consumidor da Equatorial usando  como base informações do relacionamento temporal da empresa com o mesmo |
| RF-3 | Apresentar o risco judicial classificado em perfis de risco e assunto do  processo, onde deve ser destacado os perfis com maior risco |
| RF-4 | Permitir a configuração dos perfis de risco, seus limites, identificação  e documentação |
| RF-5 | Permitir a configuração dos assuntos base através de adição de novos  assuntos, remoção ou edição de existentes |
| RF-6 | Visualizar as características que contribuem para o risco previsto to-  mando como base os nível de cliente e grupos de clientes (bairro, ci- dade, região) |
| RF-7 | Calcular o risco dos clientes continuamente, em intervalos configurados |
| RF-8 | Permitir a adaptação e evolução do módulo inteligente através de in-  terfaces próprias que exijam pouca ou nenhum conhecimento técnico |
| RF-9 | Devolver resposta formatada a ser integrada a sistemas externos como  SAP, ou Eqtl.Info |

Em validação com a cliente, verificou-se a necessidade de inclusão do RF-9 que implica na devolução de resposta do processamento integrada a sistema externos. A va- lidação dos requisitos funcionais leva também a definição de módulos funcionais destes requisitos, conforme apresentado na Tabela [4](#_bookmark10).

Tabela 4 – Mapeamento Requisitos Funcionais com Módulos do Sistema

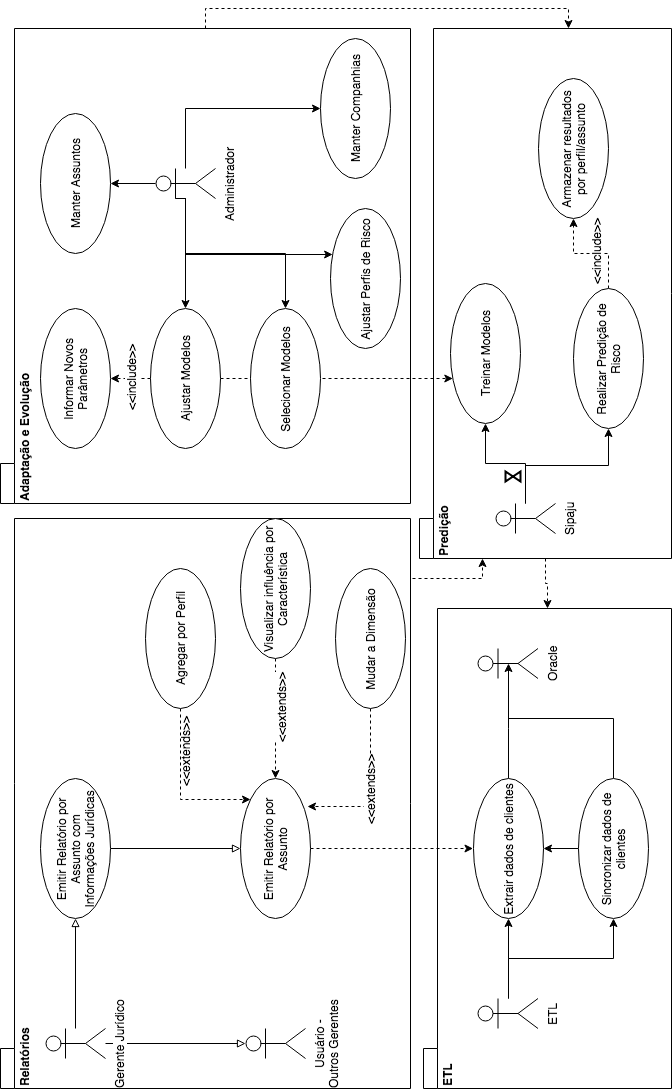
|  |  |
| --- | --- |
| **Módulo**: | **Requisito Funcional** |
| Predição | RF-2, RF-3, RF-6, RF-7, RF-9 |
| ETL | RF-1, RF-9 |
| Adaptação e Evolução | RF-4, RF-5, RF-7 |
| Web de Relatórios | RF-4, RF-6, RF-8 |

Através da definição de agrupamento de funcionalidades de módulos e requisi- tos funcionais, foram traçados modelos de casos de uso de maneira separada para cada módulo. A ideia consiste em validar as funcionalidades observadas pelos usuários (aqui representados na forma de atores). Foram identificados os seguintes atores para o Sipaju:

* Administrador: responsável por configuração e manutenção do sistema.
* ETL/Oracle: atores externos que representam sistemas/aplicações que executam de forma autônoma dentro da arquitetura
* Sipaju: representa o próprio sistema que atualiza-se e mantém-se automaticamente ao longo do tempo
* Usuário - Outros Gerentes: representa o acesso autenticado aos resultados do pro- cessamento
* Gerente Jurídico: possui as mesmas funcionalidades de Usuário - Outros Gerentes adicionando o acesso privilegiado a dados

A visão geral de casos de uso por módulos é apresentada na Figura [2](#_bookmark11). Essa figura também apresenta dependências entre casos de uso assim como de módulo que serão explicitadas na modelagem da arquitetura.

Figura 2 – Visão geral de casos de uso agregada por módulos



Assim, os seguintes casos de uso propostos serão implementados:

* Módulo de Predição
  + US-1: Realizar Predição de Risco
  + US-2: Armazenar resultados por perfil/assunto
  + US-3: Treinar Modelos
* Módulo de ETL
  + US-4: Extrair dados de clientes
  + US-5: Sincronizar dados de clientes
* Módulo de Adaptação e Evolução
  + US-6: Ajustar Modelos
  + US-7: Informar Novos Parâmetros
  + US-8: Selecionar Modelos
  + US-9: Ajustar Perfis de Risco
  + US-10: Manter Assuntos
  + US-11: Manter Companhias
* Módulo de Relatórios
  + US-12: Emitir Relatório por Assunto
  + US-13: Agregar por Perfil
  + US-14: Visualizar influência por Característica
  + US-15: Mudar a Dimensão
  + US-16: Emitir Relatório por Assunto com Informações Jurídicas

Para o atendimento do requisito funciona RF-9 adicionado, foram alterados os casos de uso US-2 e US-4, que são detalhados em seguida nas Tabelas [5](#_bookmark12) e [6](#_bookmark13).

Tabela 5 – Detalhamento US-2: Armazenar resultados por perfil/assunto

|  |  |
| --- | --- |
| **Use Case ID**: | US-2 |
| **Nome:** | Armazenar resultados por perfil/assunto. |
| **Atores:** | Sipaju |
| **Descrição:** | Recebe os resultados do processamento e guarda no banco  de dados permitindo a recuperação estruturada. |
| **Pré-condições:** | Evento de dados gerados por US-1 |
| **Pós-condições:** | Dados persistidos de maneira estruturada |
| **Curso Normal:** | 1. Estrutura informação 2. Persiste em banco de dados |
| **Cursos Alternativos:** | Não existem |

Tabela 6 – Detalhamento US-4: Extrair dados de clientes

|  |  |
| --- | --- |
| **Use Case ID**: | US-4 |
| **Nome:** | Extrair dados de clientes |
| **Atores:** | ETL |
| **Descrição:** | Realiza a integração com as múltiplas fontes de dados, ex-  trai os dados usando como parâmetro básico a conta con- trato do cliente, e mantém esses dados em banco de dados estruturado para o Sipaju. Adicionalmente envia os dados de processamento para sistemas/outras fontes de dados que sejam necessárias e configuradas para esta finalidade |
| **Pré-condições:** | Acesso e conexão estabelecida com as fontes de dados |
| **Pós-condições:** | Banco de dados alimentado com novos clientes, relaciona-  mentos, processos, etc. |
| **Curso Normal:** | 1. ETL estabelece conexão com fontes de dados 2. Executa cada uma das regras criadas para importação de dados 3. Realiza as transformações necessárias pré- parametrizadas 4. Armazena dados extraídos na base do Sipaju |
| **Curso Alternativo:** | 1. ETL estabelece conexão com fontes de dados/Sistema 2. Envia dados de processamento em formato definido |
| **Cursos Alternativos:** | Não existem |

## Visão Arquitetural

A partir da definição funcional, é possível planejar a construção dos artefatos de software a partir de uma arquitetura comum. A Figura [3](#_bookmark15) apresenta uma representação de componentes e interfaces do Sipaju. Os componentes são construções funcionais de soft- ware que podem ser reaproveitadas. Dentro de um sistema, ele possuem relacionamentos de dependência de uma biblioteca, sistema operacional, dentre outros e dependência da implementação de outras funcionalidades presentes em outros componentes. As funcio- nalidades implementadas em outros componentes e disponibilizadas publicamente para integração são colocadas em interfaces (círculo no modelo).

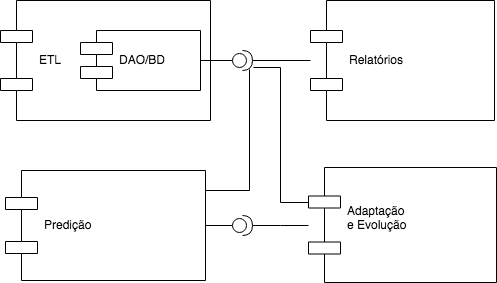


Figura 3 – Visão de componentes e integração de interfaces entre os módulos

Na Figura [3](#_bookmark15) observamos cinco componentes: Adaptação e Evolução; DAO/BD; ETL; Relatórios; e Predição.

O componente ETL, é formado por múltiplas implementações, dependentes da fer- ramenta de ETL e possui internamente um componente DAO/DB. A ideia é representar a camada de acesso a dados (DAO) e banco de dados (DB) através de um componente que fornece uma interface. Essa interface fornecida é consumida por todos os outros com- ponentes. A ideia consiste em que a base de dados seja o principal ponto de integração de dados visto que diferentes componentes terão diferentes plataformas de implementação, mas compartilham a mesma fonte de dados.

O componente de Predição fornece a interface para ajustar modelos, esta interface é utilizada pelo componente Adaptação e Evolução. Tanto o componente Adaptação e Evolução quando o Relatórios são componentes com interface de usuário.

Especificamente, os componentes de ETL e Predição podem ser representados pelos diagramas de classes apresentados nas Figuras [4](#_bookmark16) e [5](#_bookmark17). O componente de predição pode ser abstraído na forma de uma coleção de métodos que são compostos por estatísticas, parâmetros, modelos e agregados a outros métodos e algoritmos. Os algoritmos podem ser especializados de acordo com o problema em questão. O assunto, por sua vez, que terá um ou mais métodos que possa resolvê-lo, é agregado a preditores que representam métodos construídos e treinados.

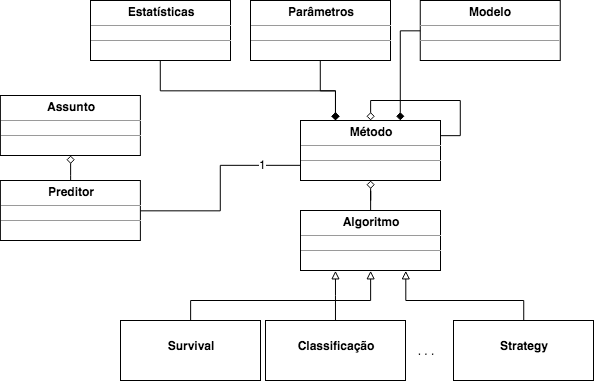


Figura 4 – Visão de classes do componente de Predição

Interessante destacar que a arquitetura proposta na forma de abstração permite a fácil adaptação de novos algoritmos (através de herança e polimorfismo), combinações de métodos já existentes, adaptação de métodos de um assunto para outro assunto assim como o desacoplamento de implementação, sendo viável portanto buscar implementações mesmo que em linguagens diferentes via integração de interface.

Essa característica é extremamente desejável para projetos de P&D onde não existe a priori uma definição clara dos métodos que serão desenvolvidos. A versatilidade, por- tanto, é uma característica extremamente desejável nesse tipo de aplicação. Essa tam- bém é uma vantagem tecnológica para a manutenção e evolução do software durante o tempo, ao permitir adaptação contínua independente dos módulos previamente construí- dos, contanto que os novos módulos sigam o mesmo padrão de especificação previamente estabelecido.

Já o componente de ETL pode ser visto como uma agregação de regras gerenciadas pela classe Manager que atua sobre as entidades no banco através de objetos DAO. As implementações específicas dos DAO e das Entidades também deverão ser usadas pelos outros componentes via interface específica disponibilizada.

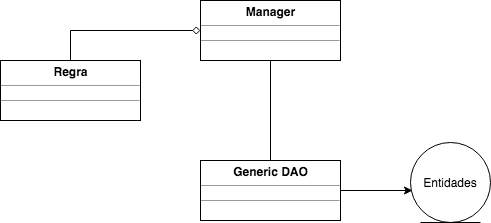


Figura 5 – Visão de classes do componente ETL

Genericamente podemos apresentar o diagrama de classes dos componentes Re- latórios e Adaptação e Evolução através do diagrama apresentado na Figura [6](#_bookmark18). Nesse modelo percebe-se que todas as interfaces herdaram de um template comum, contendo componentes comuns a todas as páginas. Isso ajuda a navegação, interação e usabilidade. Todas as interfaces serão controladas por classes controladoras, que implementam o mo- delo Model-View-Controller, comum em aplicações Web pela versatilidade, separação de papéis em 3 camadas, e facilidade de manutenção.

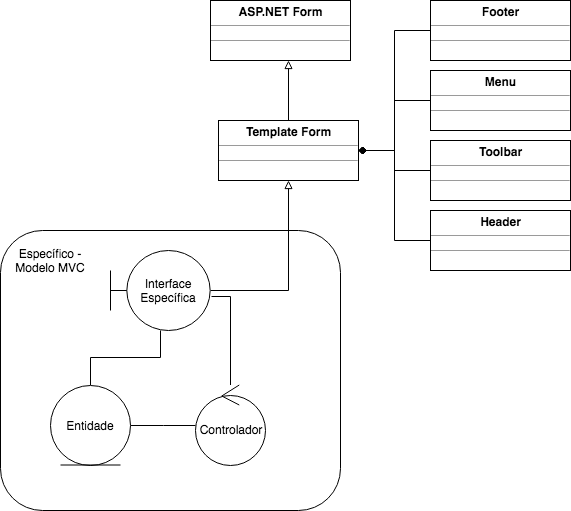


Figura 6 – Modelo genérico da visão de classes dos componentes Relatório e adaptação e evolução

## Visão de Implantação

A visão de implantação apresenta como os componentes do sistema estão organi- zado em nós de computação. E como eles devem estabelecer comunicação. Aqui se utiliza uma arquitetura distribuída, com modelo de interações cliente-servidor, onde o processa- mento será realizado em serviços background.

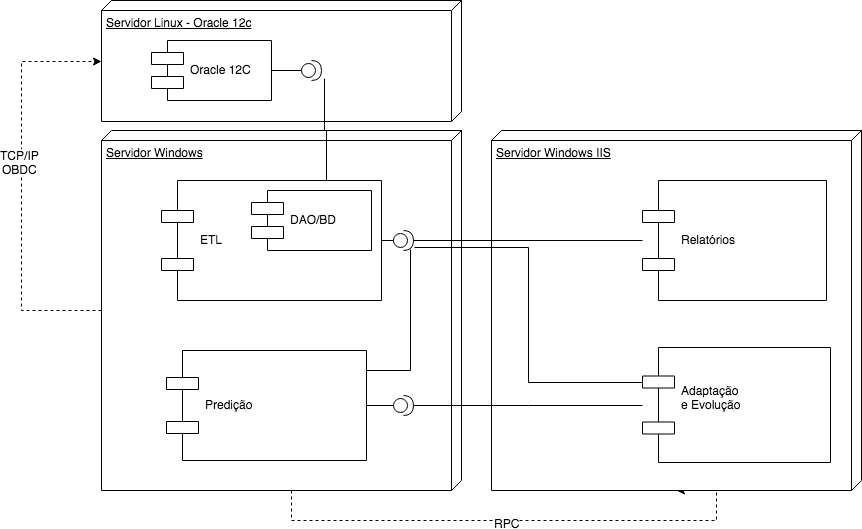
A Figura [7](#_bookmark20) apresenta o modelo de implantação do Sipaju.

Figura 7 – Diagrama deImplantação do Sipaju Espera-se para o sistema os seguintes nós e finalidades:

* Servidor Windows: responsável por rodar os serviços de extração e sincronização de dados e de predição de predisposição jurídica
* Servidor Linux - Oracle 12C: pool do Oracle para armazenar dados gerados
* Servidor Windows IIS: servidor para as aplicações web e interfaces de gerenciamento do Sipaju