# Especificação do Programa de Sugestão de Apostas para Mega-Sena

Este documento detalha a especificação para um programa Java que gera sugestões de apostas para a Mega-Sena, utilizando estatísticas de sorteios anteriores.

## 1. Visão Geral

O programa visa auxiliar apostadores da Mega-Sena, fornecendo sugestões de jogos baseadas em análises estatísticas aprofundadas e estratégias configuráveis. Ele será capaz de gerar um número especificado de apostas, utilizando dados históricos da Mega-Sena para ranquear as combinações mais promissoras.

## 2. Requisitos Funcionais

### 2.1. Geração de Sugestões de Apostas

O programa deverá ser capaz de gerar um número de sugestões de apostas informado como parâmetro POSIX.

### 2.2. Exibição de Mensagens e Internacionalização (NLS)

As mensagens exibidas pelo programa, incluindo datas, horas e valores, deverão ser compatíveis com o NLS (National Language Support) do ambiente operacional. Caso a detecção automática não seja possível, o idioma padrão será Português Brasil.

### 2.3. Base de Dados Estatísticos

As apostas sugeridas para o próximo concurso da Mega-Sena serão calculadas com base em estatísticas de sorteios disponíveis no site [*Mega Power-Loterias*](https://megapower-loterias.com.br/app/index.php).

### 2.4. Ranking de Apostas

As apostas sugeridas serão ranqueadas com base na soma da pontuação alcançada em todas as estratégias às quais forem submetidas para avaliação. As apostas com maior ranking serão as selecionadas como sugestões finais.

### 2.5. Estratégias de Avaliação

As estratégias de avaliação serão implementadas como classes que estendem uma classe abstrata ou implementam uma interface. Isso permitirá a criação de novas estratégias e a fácil integração de abordagens mais sofisticadas.

### 2.6. Formas de Geração de Apostas

As apostas a serem geradas e submetidas à cadeia de estratégias poderão ser de uma das três formas, definidas por parâmetro POSIX na chamada do programa:

* **Apostas Aleatórias:** Geração de um número especificado de apostas aleatoriamente.
* **Índice Lexicográfico:** Geração de apostas com base na escolha aleatória de um índice lexicográfico entre as 50.063.860 combinações possíveis. O índice 1 corresponde à aposta [01 - 02 - 03 - 04 - 05 - 06], e o índice 50.063.860 corresponde à aposta [55 - 56 - 57 - 58 - 59 - 60].
* **Avaliação de Todas as Combinações:** Avaliação de todas as combinações possíveis, da [01 - 02 - 03 - 04 - 05 - 06] à [55 - 56 - 57 - 58 - 59 - 60].

### 2.7. Armazenamento de Dados

O programa utilizará um banco de dados SQLite 3 para armazenar informações estatísticas. O banco de dados, denominado MegaPower.db, está disponível para download em <https://megapower-loterias.com.br/dados/MegaPower.db>.

Este banco de dados contém informações granulares e agregadas de todos os sorteios da Mega-Sena desde o primeiro concurso. Tabelas e views notáveis incluem:

* **DEZ\_OCORRENCIAS:** Contém informações granulares da ocorrência de cada dezena, desde o primeiro sorteio.
* **AGREG\_DIS\_DEZ\_POSICAO:** Mantém uma linha para cada dezena exibindo a quantidade de ocorrências da dezena por posição no sorteio.
* **FAT\_APOSTAS:** Armazenará as sugestões de apostas geradas, juntamente com a data da aposta, a data do sorteio, o concurso, a soma do ranking das estratégias e a comparação do ranking do resultado real do último concurso.

### 2.8. Abordagens de Estratégias (Não Exaustivo)

As estratégias a serem implementadas podem incluir, mas não se limitam a:

* **Ranking Gaussiano:** Baseado em múltiplos critérios.
* **Análise de Transição de Estados:** Utilizando a tabela EST\_MUD\_ESTADO (se presente no banco de dados).
* **Probabilidades Condicionais:** Por posição, baseadas no histórico de ocorrências.
* **Sistema de Scoring Robusto:** Com pesos estatísticos.
* **Outras Estratégias Relevantes:** Que a equipe de desenvolvimento julgar interessantes.

### 2.9. Análise de Desempenho Pós-Sorteio

Com base no resultado do último sorteio, o programa deverá analisar o desempenho das sugestões de apostas mantidas na tabela FAT\_APOSTAS.

## 3. Requisitos Não Funcionais

### 3.1. Linguagem de Programação

O programa será desenvolvido em Java.

### 3.2. Concorrência e Performance

* **Multithread:** O programa será multithread para processar os cálculos estatísticos de forma eficiente.
* **Pool de Conexões SQLITE 3:** Utilização do Apache Commons DBCP para gerenciar conexões com o SQLITE 3, evitando o problema de "*host busy*" e gargalos entre as threads. O pool terá um limite de conexões ativas e um timeout configurável de 5 segundos.
* **Paralelização:** Cálculos estatísticos distribuídos em múltiplos núcleos, utilizando Streams paralelos para processamento eficiente e ExecutorService para gerenciamento de threads.

### 3.3. Resiliência

* **Sistema de Retry:** Implementação de tentativas automáticas para falhas temporárias.
* **Backoff Exponencial:** Atraso exponencial entre as tentativas de retry.
* **Failover:** Mecanismo de failover após 3 tentativas de retry.

### 3.4. Cache Inteligente

Será implementado um cache local para armazenar scores calculados, com uma validade de 1 minuto para atualizações, visando otimizar o desempenho.

### 3.5. Gerenciamento de Recursos

* **Fechamento Adequado:** Garantia do fechamento adequado de conexões e threads.
* **AwaitTermination:** Utilização de AwaitTermination para um desligamento gracioso do programa.
* **Tratamento de Exceções:** Implementação de um tratamento robusto de exceções para garantir a estabilidade do programa.

## 4. Estrutura do Projeto (Sugestão)

* **Entidades:** Classes para representar apostas, sorteios, dezenas, etc.
* **DAO (Data Access Object):** Camada de acesso ao banco de dados para interagir com as tabelas DEZ\_OCORRENCIAS, AGREG\_DIS\_DEZ\_POSICAO, FAT\_APOSTAS, e outras.
* **Estratégias:** Pacote contendo as implementações das classes de estratégia (interface ou abstract class e suas concretas).
* **Serviços/Lógica de Negócio:** Classes para orquestrar a geração de apostas, aplicação de estratégias e cálculo de rankings.
* **Utilitários:** Classes para manipulação de NLS, tratamento de erros, gerenciamento de threads e pool de conexões.
* **Main:** Classe principal para a execução do programa e parsing dos parâmetros POSIX.

## 5. Próximos Passos

Após a implementação desta especificação, serão criadas versões subsequentes com o nome Especificacao\_V00n (onde 'n' é o número da versão).