# Especificacao.md

## Nome do Projeto

**Gerador de Apostas Inteligente da Mega-Sena**

## Objetivo

Desenvolver um programa em Java capaz de gerar sugestões de jogos para a Mega-Sena com base em estratégias estatísticas, com suporte a geração massiva de apostas, paralelização e persistência dos dados para análise futura.

## Requisitos Funcionais

### 1. Execução via Terminal (POSIX)

O programa será executado via linha de comando com argumentos POSIX para definir: - Quantidade de apostas a serem geradas aleatoriamente. - Índices lexicográficos específicos para geração de combinações. - Modo de varredura completa de todas as combinações (50.063.860 possibilidades).

### 2. Localização (NLS)

* Detectar automaticamente o locale do sistema.
* Caso não seja possível, usar o padrão pt-BR (Português Brasil).
* Todas as mensagens devem incluir timestamp e valores no formato local.

### 3. Estratégias de Avaliação

* Implementações de uma interface ou classe abstrata que recebe uma aposta e retorna um score.
* Pontuação final de uma aposta é a soma dos scores das estratégias aplicadas.

### 4. Geração de Apostas

Suporte a três modos: 1. Aleatório com número de apostas. 2. Por índice lexicográfico. 3. Varredura total (modo pesado, multithread).

### 5. Ranking das Apostas

* As apostas geradas serão ranqueadas pela soma dos scores recebidos pelas estratégias.
* Apenas as de maior pontuação serão sugeridas.

### 6. Banco de Dados

* Utilizar SQLite 3 (arquivo: MegaPower.db).
* Usar Apache Commons DBCP para pool de conexões.
* Timeout de conexão configurável (padrão: 5s).
* Persistir apostas sugeridas em FAT\_APOSTAS.

### 7. Estratégias Estatísticas (Exemplos)

* Ranking gaussiano.
* Análise de transições de estados (EST\_MUD\_ESTADO).
* Probabilidades condicionais por posição.
* Scoring com pesos estatísticos.

### 8. Avaliação Pós-Concurso

* Após cada novo sorteio, comparar as apostas sugeridas com o resultado real.
* Atualizar FAT\_APOSTAS com ranking do resultado real.

## Requisitos Não-Funcionais

### Multithreading

* Uso de ExecutorService para paralelização.
* ForkJoinPool ou parallelStream() para estratégia de processamento.

### Resiliência

* Sistema de Retry com backoff exponencial.
* Failover após 3 tentativas de conexão.

### Cache Inteligente

* Scores intermediários devem ser armazenados com validade de 1 minuto.
* Recalcular apenas dados expirados.

### Tratamento de Recursos

* Fechar conexões e threads corretamente.
* Uso de try-with-resources e awaitTermination.

### Tratamento de Exceções

* Logging estruturado.
* Categorias de erro: conexão, banco, lógica, parâmetros.

## Estrutura do Banco de Dados

### Tabelas principais

* DEZ\_OCORRENCIAS : Ocorrência das dezenas por concurso.
* AGREG\_DIS\_DEZ\_POSICAO: Frequência de cada dezena por posição.
* EST\_MUD\_ESTADO: Transição de estados estatísticos.
* FAT\_APOSTAS: Armazena apostas sugeridas, suas pontuações, data e resultado comparado.

### Views e outras tabelas úteis (a documentar):

* A consultar após importação do banco.

## Fonte dos Dados

* Banco disponível em: https://megapower-loterias.com.br/dados/MegaPower.db

## Versão

**V000** — Data: 2025-06-10

## Próximos passos

* Definição da estrutura de parâmetros POSIX.
* Implementação da interface EstrategiaAvaliacao.
* Protótipo de geração e ranking das apostas.
* Módulo de avaliação com base no resultado mais recente.

© 2025 - Projeto OpenMegaPower by [Seu Nome ou Equipe]