# **DOCUMENTO DE LEVANTAMENTO DO PROJETO JAVA GRID F1**

Dupla: Rafael Antunes Souza e Rafael Felipe Vieira

## **1. INTRODUÇÃO**

Este documento descreve a estrutura e justificativa de cada componente do sistema Java Grid F1, um simulador de campeonato de Fórmula 1 que gerencia equipes, pilotos, corridas e calcula classificações conforme o regulamento da FIA.

## **2. NECESSIDADES DO SISTEMA**

### **2.1 Requisitos Funcionais**

* Gerenciar cadastro de equipes e pilotos
* Registrar resultados de corridas
* Calcular pontuações conforme regulamento FIA
* Gerar classificações de pilotos e equipes
* Fornecer relatórios formatados

### **2.2 Requisitos Não-Funcionais**

* Interface textual intuitiva
* Código orientado a objetos bem estruturado
* Saída de dados padronizada
* Fácil manutenção e extensão

## **3. JUSTIFICATIVA DAS CLASSES**

### **3.1 Main.java**

**Necessidade**: Ponto de entrada do sistema e inicialização dos dados.  
 **Responsabilidades**:

* Criar todas as instâncias iniciais (equipes, pilotos, corridas)
* Configurar relacionamentos entre objetos
* Iniciar o fluxo principal do programa  
   **Por que existe?**:
* Centraliza a inicialização do sistema
* Configura todos os dados necessários antes da execução
* Isola a criação de objetos do resto da lógica

### **3.2 Piloto.java**

**Necessidade**: Representar os pilotos participantes do campeonato.  
 **Responsabilidades**:

* Armazenar informações do piloto (nome, número)
* Manter referência à equipe
* Acumular pontos no campeonato  
   **Por que existe?**:
* Modela a entidade fundamental do domínio
* Centraliza atributos e comportamentos específicos de pilotos
* Permite rastrear pontuação individual

### **3.3 Equipe.java**

**Necessidade**: Representar as equipes/construtores da F1.  
 **Responsabilidades**:

* Armazenar nome da equipe
* Gerenciar lista de pilotos
* Acumular pontos no campeonato de construtores  
   **Por que existe?**:
* Modela outra entidade fundamental do domínio
* Gerencia relacionamento "uma equipe tem vários pilotos"
* Calcula pontuação agregada dos pilotos

### **3.4 Corrida.java**

**Necessidade**: Representar cada grande prêmio do campeonato.  
 **Responsabilidades**:

* Armazenar nome da corrida
* Manter resultado (ordem de chegada dos pilotos)  
   **Por que existe?**:
* Modela os eventos que geram pontuação
* Armazena dados específicos de cada prova
* Serve como base para cálculos de classificação

### **3.5 InterfaceFIA.java**

**Necessidade**: Definir contrato para cálculos oficiais.  
 **Responsabilidades**:

* Estabelecer métodos obrigatórios para cálculos de pontuação
* Garantir implementação consistente das regras  
   **Por que existe?**:
* Define um padrão para implementação das regras
* Facilita futuras mudanças no sistema de pontuação
* Documenta explicitamente as operações necessárias

### **3.6 Campeonato.java**

**Necessidade**: Coordenar a lógica principal do sistema.  
 **Responsabilidades**:

* Gerenciar todas as entidades (corridas, pilotos, equipes)
* Implementar regras da FIA
* Calcular classificações  
   **Por que existe?**:
* Centraliza a "inteligência" do sistema
* Isola as regras de negócio da interface
* Coordena todas as operações de cálculo

### **3.7 GerenciadorMenu.java**

**Necessidade**: Prover interface interativa com usuário.  
 **Responsabilidades**:

* Apresentar menu de opções
* Processar entrada do usuário
* Navegar entre funcionalidades
* Coordenar exibição de relatórios  
   **Por que existe?**:
* Isola toda a lógica de interação com usuário
* Facilita manutenção da interface
* Organiza fluxo de navegação

### **3.8 FormatadorRelatorios.java**

**Necessidade**: Padronizar a exibição de dados.  
 **Responsabilidades**:

* Formatar tabelas e relatórios
* Controlar largura e alinhamento
* Gerar cabeçalhos e rodapés  
   **Por que existe?**:
* Centraliza toda formatação de saída
* Garante consistência visual
* Simplifica a geração de relatórios complexos

## **4. Diagrama de classes**

