



Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE
Engenharia de Software

FTV Scout: Sistema para Notação e Análise de Desempenho no Futevôlei.

Autor: Felipe Direito Corrieri de Macedo
Orientador: Prof. Me. Ricardo Ajax Kosloski
Coorientador: Prof. Dr. Leonardo Lamas Leandro Ribeiro

Brasília, DF
2025



Felipe Direito Corrieri de Macedo

FTV Scout: Sistema para Notação e Análise de Desempenho no Futevôlei.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Orientador: Prof. Me. Ricardo Ajax Kosloski

Coorientador: Prof. Dr. Leonardo Lamas Leandro Ribeiro

Brasília, DF

2025

Felipe Direito Corrieri de Macedo

FTV Scout: Sistema para Notação e Análise de Desempenho no Futevôlei./
Felipe Direito Corrieri de Macedo. – Brasília, DF, 2025-
72 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Me. Ricardo Ajax Kosloski

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE , 2025.

1. Software para Notação. 2. Análise de Desempenho. I. Prof. Me. Ricardo Ajax Kosloski. II. Universidade de Brasília. III. Faculdade de Ciências e Tecnologias em Engenharia. IV. FTV Scout: Sistema para Notação e Análise de Desempenho no Futevôlei.

CDU 02:141:005.6

Felipe Direito Corrieri de Macedo

FTV Scout: Sistema para Notação e Análise de Desempenho no Futevôlei.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2025 – Data da aprovação do trabalho:

Prof. Me. Ricardo Ajax Kosloski
Orientador

Prof. Dr. Leonardo Lamas Leandro Ribeiro
Coorientador

Profa. Dra. Marília Mendonça Forte Gomes
Convidado 1

Profa. Ma. Cristiane Soares Ramos
Convidado 2

Brasília, DF
2025

Agradecimentos

Primeiramente, agradeço ao meu primo Victor, que me ajudou a elaborar a ideia inicial do projeto, bem como fez a ponte entre as áreas de Engenharia de Software e a Ciência do Esporte. Agradeço ao meu orientador, Ricardo, e ao coorientador, Leonardo, que toparam o desafio e fizeram contribuições riquíssimas e fundamentais para a realização de um trabalho interdisciplinar como este.

Agradeço à minha família, em especial aos meus pais, Denise e Ricardo, que sempre estiveram presentes durante toda a minha formação humana e vida acadêmica, dando-me todo o tipo de suporte necessário para chegar até aqui. Também ao meu irmão, Rafael, que, por nascer antes de mim, abriu as portas para que eu também fosse bem recebido na família.

Agradeço à minha companheira, Luísa, que está comigo desde o início da minha graduação, e sempre me deu a força e o apoio necessários para que eu pudesse chegar inteiro até aqui, além de toda a parceria em viagens, conversas e momentos de lazer, que guardo com carinho e que mantiveram o equilíbrio entre a vida pessoal e acadêmica.

Por fim, agradeço aos meus amigos Gabriel, Gatto, Amanda, Paulinho, Jone e Jovi, que conheço desde o ensino fundamental, por compartilharmos momentos de risadas e descontração, tão importantes para renovar as energias e manter a vida alegre. Assim como agradeço a todos os amigos que vieram depois, com quem sei que posso contar para o que der e vier.

Resumo

O futevôlei, um esporte brasileiro que cresce sua popularidade a cada ano, ainda carece de ferramentas tecnológicas acessíveis para auxiliar a análise de desempenho dos atletas e das equipes. Este trabalho apresenta um sistema de software que visa suprir essa lacuna, permitindo a notação de ações de jogo em tempo real a fim de gerar recursos para a análise estatísticas. O projeto desenvolvido é uma solução *open-source* e gratuita, que pode ser utilizada por qualquer pessoa interessada em melhorar seu desempenho no esporte. A proposta combina uma interface de toque (*touch*) e o reconhecimento de voz para suprir a demanda de uma notação rápida, necessária para o uso em tempo real. Além disso, o sistema conta com um *dashboard* de visualização de dados e análise estatística, que permite a análise de desempenho dos atletas e equipes, facilitando a identificação de padrões táticos e técnicos. O serviço foi desenvolvido para ser utilizado localmente, sem custos com servidores ou serviços na nuvem. Tem o objetivo de ser acessível, intuitivo, fácil de usar, e independente de conexão com a internet — muito importante para os locais onde são praticados o esporte, como praias e quadras de areia.

Palavras-chave: Open Source. Futevôlei. Análise de Desempenho. Tempo Real. Análise Notacional.

Abstract

Footvolley, a Brazilian sport that has been growing in popularity each year, still lacks accessible technological tools to support the performance analysis of athletes and teams. This work presents a software system designed to address this gap, enabling real-time notational tracking of game actions and statistical analysis. The developed project is an open-source and free solution that can be used by anyone interested in improving their performance in the sport. The proposed system combines a touch-based interface and voice recognition to meet the demand for fast notation, essential for real-time usage. Additionally, the system features a data visualization and statistical analysis dashboard, which facilitates performance evaluation and the identification of tactical and technical patterns. The service was developed to run locally, with no need for servers or cloud-based solutions. It aims to be accessible, intuitive, easy to use, and independent of internet connectivity — an essential feature for environments where the sport is practiced, such as beaches and sand courts.

Key-words: Open Source. Footvolley. Match Analysis. Real-Time. Notational Software.

Lista de ilustrações

Figura 1 – Framework de Análise Desportiva	18
Figura 2 – Poster do Framework Scrum	24
Figura 3 – Espiral do Processo de Engenharia de Requisitos	26
Figura 4 – Arquitetura de aplicações Web usando o padrão MVC	31
Figura 5 – Pipeline de Integração Contínua (GitHub Actions)	41
Figura 6 – Comunicação Cliente-Servidor Local	45
Figura 7 – Componentes de Software da Aplicação	46
Figura 8 – Casos de Uso	48
Figura 9 – Diagrama Entidade-Relacionamento	54
Figura 10 – Tela Inicial	55
Figura 11 – Tela de Notação da Partida	56
Figura 12 – Resumo da Partida	58
Figura 13 – Estatísticas por Dupla	59
Figura 14 – Estatísticas Individuais	60
Figura 15 – Mapa de Ações - Ataque	61
Figura 16 – Mapa de Ações - Recepção	61
Figura 17 – Histórico de Partidas de um Atleta	62
Figura A.1 – Sports Analysis Framework	71

Lista de tabelas

Tabela 1 – Categorias e variáveis	19
Tabela 2 – Comparação entre Sistemas de Análise de Desempenho	22
Tabela 3 – Histórias de Usuários para o Épico de Gestão de Partidas.	50
Tabela 4 – Histórias de Usuários para o Épico de Notação em Tempo Real	51
Tabela 5 – Histórias de Usuários para o Épico de Visualização e Análise de Dados	52
Tabela 6 – Requisitos Não Funcionais.	53
Tabela 7 – Cronograma Estimado de Desenvolvimento.	63
Tabela B.1 – Categorias e variáveis	72

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
CSV	<i>Comma-Separated Values</i>
DER	Diagrama de Entidade e Relacionamento
GQM	<i>Goal Question Metric</i>
GPL-3.0	<i>GNU General Public License v3.0</i>
IA	Inteligência Artificial
US	<i>User Story</i>
OE	Objetivo Específico

Sumário

1	INTRODUÇÃO	13
1.1	Considerações Iniciais	13
1.2	Contextualização	14
1.2.1	Questão de Pesquisa	14
1.2.2	Objetivos	15
1.2.2.1	Objetivo Geral	15
1.2.2.2	Objetivos Específicos	15
1.3	Justificativa	15
1.4	Prévia dos Capítulos	16
2	REFERENCIAL TEÓRICO	17
2.1	Modelos analíticos para avaliação do desempenho no esporte	17
2.1.1	Importância da análise de desempenho no esporte	17
2.1.2	Conceito e modelos de avaliação de desempenho esportivo	17
2.1.3	Especificidades da análise notacional no futevôlei	18
2.2	Sistemas Computacionais para Análise de Desempenho no Esporte	19
2.2.1	Introdução ao uso de sistemas computacionais no esporte	19
2.2.2	Características dos sistemas de notação e análise estatística	20
2.2.3	Exemplos de sistemas comercializados atualmente	20
2.2.3.1	Data Volley 4	21
2.2.3.2	SportsCode	21
2.2.3.3	Lince Plus	21
2.2.4	Análise comparativa entre os sistemas computacionais existentes	22
2.3	Engenharia de Software	23
2.3.1	Ciclo de Vida: SCRUM/XP	23
2.3.1.1	Movimento Ágil	23
2.3.1.2	SCRUM	24
2.3.1.3	XP	25
2.3.2	Engenharia de Requisitos	26
2.3.3	Persistência: Banco de Dados	27
2.3.4	Qualidade de Software	28
2.3.4.1	Conceito de Qualidade de Software	28
2.3.4.2	Qualidade do Produto e em Uso	28
2.3.4.3	Métricas de Software	29
2.3.4.4	Como Estabelecer Métricas: Modelo GQM	30

2.3.5	Arquitetura de Software	31
2.3.6	UML - Linguagem de Modelagem Unificada	32
2.3.6.1	Técnicas Essenciais da UML	32
2.3.7	Testes de Software	32
2.3.7.1	Testes Unitários	33
2.3.7.2	Testes de Componente	33
2.3.7.3	Testes de Sistema	34
2.3.7.4	Testes de <i>Release</i>	34
2.3.8	Gestão e Configuração de Software - DevOps	34
3	METODOLOGIA	36
3.1	Metodologia de Pesquisa	36
3.1.1	Quanto à Abordagem: Pesquisa Quantitativa	36
3.1.2	Quanto à Natureza: Pesquisa Aplicada	36
3.1.3	Quanto aos Objetivos: Pesquisa Exploratória	37
3.1.4	Quanto aos Procedimentos Técnicos	37
3.1.5	Metodologia de Desenvolvimento do Software	37
3.1.5.1	Ciclo de Vida	37
3.1.5.2	Gerência de Configuração de Software	39
3.1.6	Engenharia de Requisitos	39
3.1.6.1	Estudo de Viabilidade	40
3.1.6.2	Elicitação e Análise	40
3.1.6.3	Especificação	40
3.1.6.4	Validação	41
3.1.7	Metodologia de Avaliação e Validação	41
3.1.7.1	Plano de Testes	41
3.1.7.1.1	Testes de Desenvolvimento	41
3.1.7.1.2	Teste de <i>Release</i> e Desenho do Estudo de Caso	42
3.1.7.2	Plano de Métricas e Definição do Modelo GQM	42
4	SOLUÇÃO IMPLEMENTADA	45
4.1	Arquitetura da Solução	45
4.1.1	Descrição dos Componentes de Software	46
4.2	Requisitos de Software	48
4.2.1	Diagrama de Casos de Uso	48
4.2.2	Backlog do Produto	49
4.2.2.1	ÉPICO 1: Gestão de Partidas	49
4.2.2.2	ÉPICO 2: Notação em Tempo Real	50
4.2.2.3	ÉPICO 3: Visualização e Análise de Dados	51
4.2.2.4	ÉPICO 4: Requisitos Não Funcionais	52

4.2.3	Modelo de Dados: Diagrama Entidade-Relacionamento	53
4.2.4	Apresentação da Aplicação	54
4.2.5	Cronograma de Desenvolvimento	62
5	CONSIDERAÇÕES FINAIS	64
5.1	Considerações sobre os Objetivos	64
5.2	Resultados	65
5.3	Trabalhos Futuros	65
5.4	Fragilidades Externas da Aplicação	66
5.5	Conclusões	66
	REFERÊNCIAS	68
	ANEXOS	70
	ANEXO A – FIGURA ORIGINAL	71
	ANEXO B – TABELA ORIGINAL	72

1 Introdução

1.1 Considerações Iniciais

O futevôlei é um esporte brasileiro criado na década de 60 e, a cada ano, aumenta sua relevância na mídia e o interesse do público pela atividade. Apesar desta popularização crescente, inclusive internacionalmente, ainda é um esporte que carece de investimentos e estudos sobre a prática, principalmente se comparado a outros esportes similares, como o vôlei e o vôlei de praia ([LIMA et al., 2023](#); [BORGES; SENATORE; LAMAS, 2024](#)).

Em esportes competitivos, métricas são essenciais para analisar o desempenho das equipes. No futevôlei, elas definem ações discretas, como saque, ataque, recepção, levantamento, e, essas ações, geram um determinado resultado, que pode ser coletado ao decorrer de toda a partida. Essas marcações de ações do jogo, chamadas de notações, geram dados objetivos sobre a partida, possibilitando eliminar suposições e percepções individuais sobre o jogo, e tornam a análise estatística possível. Essa prática é chamada de Análise de Desempenho. ([GÓMEZ-RUANO, 2018](#))

A Análise de Desempenho permite fundamentar treinos e táticas específicas para melhorar as condições de vitória da equipe. Com base nos dados coletados, define-se uma estratégia de treinamento que é aplicada e posteriormente validada em competição. Após cada partida, novos dados são coletados e analisados para recalibrar o processo de treinamento conforme o necessário. Esse ciclo iterativo de planejamento (desenvolvimento da estratégia), treino (aperfeiçoamento) e performance (jogo, onde ocorre novas coletas de dados), constitui o Ciclo de Aprendizagem da Equipe, definido em [Rangel et al. \(2023\)](#), muito importante para a evolução dos atletas, e, conseqüentemente, do esporte.

Atualmente, no futevôlei, a ausência de tecnologia específica que auxilie a coleta de dados no esporte representa um desafio significativo ([BORGES; SENATORE; LAMAS, 2024](#)). Esta lacuna faz com que atletas, treinadores e profissionais do esporte tenham dificuldade em ter uma análise de desempenho eficaz e, conseqüentemente, limita-se os recursos disponíveis para realizar o Ciclo de Aprendizagem da Equipe. Por isso, o objetivo principal deste trabalho é desenvolver uma ferramenta de software que facilite a coleta de dados, a partir de métricas definidas para gerar insumos que viabilizem a análise aprofundada de partidas e desempenhos dos atletas.

A construção deste trabalho conta com um professor doutor em Ciência do Esporte que possui diversos artigos e publicações na área, como em: [Borges, Senatore e Lamas \(2024\)](#) e [Rangel et al. \(2023\)](#). Foi fundamental a colaboração do professor especialista para definir as métricas e o embasamento teórico necessário para a construção de uma

solução de software que atenda às necessidades do esporte. Ele atuou como o *stakeholder* principal, identificando as necessidades e as expectativas do projeto.

Para garantir a boa usabilidade do produto, ele deve ser responsivo, adaptando-se a dispositivos móveis, como *tablets*, ou computadores para viabilizar o uso em diversos ambientes e situações. Ele também deve ser intuitivo, ou seja, é importante que sua interface seja clara e objetiva para diminuir a curva de aprendizagem dos usuários e facilitar o uso de todo o público-alvo.

Além disso, é necessário que o produto seja escalável, desenvolvido em uma arquitetura de software moderna para facilitar evoluções futuras. Uma solução que dê abertura para acoplamentos de novas implementações, como o uso de visão computacional ou algoritmos de predições, por exemplo. Funções estas que estão fora do escopo deste trabalho, mas que tem a implementação facilitada em uma estrutura de fácil manutenção e boa inteligibilidade.

Para isso, entende-se que a solução deva ser de código aberto (*open-source*), ou seja, gratuito e livre para a utilização, democratizando o uso de um sistema inovador e com muito potencial para o esporte por toda a sociedade.

A ferramenta proposta pode auxiliar os profissionais da Educação Física, em sua atuação como treinadores e professores da modalidade, a demonstrar, na prática, a aplicação de estratégias mais eficazes dependendo do contexto em que o esporte está inserido. Isto pode fomentar a competitividade e a prática do futevôlei em diversos nichos da sociedade, além do claro valor que as equipes competitivas podem usufruir.

1.2 Contextualização

1.2.1 Questão de Pesquisa

Considerando a lacuna na coleta e amostragem de dados para desenvolver táticas e treinos específicos do futevôlei, e considerando o possível impacto no esporte, o problema de pesquisa do trabalho é:

- Como desenvolver uma aplicação de software que auxilie na coleta de dados em tempo real no futevôlei, a fim de gerar insumos para análises de desempenho baseadas nas notações de ações do jogo e suas métricas?

1.2.2 Objetivos

1.2.2.1 Objetivo Geral

Disponibilizar uma ferramenta de software que sistematize as notações em tempo real de ações de partidas de futevôlei, a fim de gerar subsídios para a análise de desempenho e a visualização dos dados coletados.

1.2.2.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

1. Identificar as métricas existentes para definir as ações de jogo (notações).
2. Estabelecer um modelo de análise para identificação de padrões táticos e técnicos no futevôlei.
3. Projetar uma arquitetura de sistema de software que permita a coleta de dados (notações) em tempo real.
4. Determinar e implementar, com base no modelo definido, gráficos para visualização de dados que facilitem a análise estatística.
5. Desenvolver a solução de software projetada.
6. Aplicar a solução de software em jogos reais.

1.3 Justificativa

O futevôlei, um esporte muito praticado no Brasil, ainda carece de uma tecnologia para auxiliar a análise de desempenho das equipes nas partidas ([BORGES; SENATORE; LAMAS, 2024](#)). A implementação de um sistema de software que desempenhe essa função pode impulsionar significativamente o desenvolvimento do esporte.

Como mostrado no artigo de [Ramos et al. \(2023\)](#), a análise de partidas tem impacto positivo na performance dos jogadores. Evidenciar os pontos fortes e fracos de cada equipe, como as áreas de ataque mais efetivas ou zonas de defesa que precisam de ajustes, são dados que fazem a diferença no treinamento diário dos atletas.

Com isso, o produto propõe possibilitar a coletar dados detalhados sobre ações de jogo como saques, ataques, levantamentos, recepções, bem como os resultados obtidos em cada uma delas, os erros e acertos. Esta atividade, chamada de notação, é fundamental para sistematizar a análise de desempenho e a análise estatística, objetivos finais mais importantes do produto.

Além disso, a partir da sistematização das análises e notações feitas diretamente no software, espera-se que o produto gere insumos para equipes competitivas aperfeiçoarem técnicas e treinos, além de contribuir para a profissionalização da análise estatística de desempenho dos atletas. Também se espera que haja um impacto positivo entre participantes do esporte em geral, como amadores e iniciantes, que desejam melhorar suas habilidades e compreender seus pontos fortes e aspectos de melhoria.

O trabalho preenche uma lacuna de tecnologia para a notação e análise de desempenho no futevôlei, elementos que agregam muito valor em outros esportes e espera-se que tenha o mesmo impacto no esporte brasileiro. Além disso, incentiva o desenvolvimento de novas tecnologias específicas e estudos que podem contribuir para a evolução do futevôlei.

1.4 Prévia dos Capítulos

Além deste capítulo introdutório que conta com as considerações iniciais, questão de pesquisa, objetivos e justificativa, o trabalho está estruturado da seguinte forma:

- **Capítulo 2:** REFERENCIAL TEÓRICO; aprofunda os conceitos da ciência do esporte e da engenharia de software, necessários para a fundamentação da pesquisa e para o desenvolvimento do trabalho.
- **Capítulo 3:** METODOLOGIA; descreve as metodologias de pesquisa, de desenvolvimento do software e de avaliação e validação.
- **Capítulo 4:** SOLUÇÃO IMPLEMENTADA; detalha o desenvolvimento da solução, as tecnologias usadas e imagens da aplicação em funcionamento.
- **Capítulo 5:** CONCLUSÃO; apresenta os resultados do trabalho e considerações finais.

2 Referencial Teórico

2.1 Modelos analíticos para avaliação do desempenho no esporte

2.1.1 Importância da análise de desempenho no esporte

Segundo [Ramos et al. \(2023\)](#), a análise de jogo e as notações conseguem fundamentar as tomadas de decisão das equipes. Por meio das coletas de dados e as análises de desempenho, é possível planejar as sessões de treinamento de forma mais específica, focando, objetivamente, nos pontos que precisam de melhoria juntamente com as necessidades dos atletas.

O estudo de caso descrito por [Ramos et al. \(2023\)](#), mostra a evolução de uma equipe de vôlei ao decorrer de um campeonato. Foi realizado dois ciclos de coletas de dados e treinamentos específicos. Entre os dados coletados, eles contaram com entrevistas, gravações de treinamentos e notações de quadra. O experimento foi realizado sob o acordo de sempre basear as decisões apoiadas nas notações e análises estatísticas geradas. A equipe de treinadores conseguiu realizar os ajustes de treino com base nas reais falhas táticas e técnicas da equipe. Por meio das análises sistematizadas de desempenho, reduziu-se as suposições e análises empíricas que são comumente utilizadas por treinadores de equipes menos profissionais. As notações e a análise de desempenho foram fundamentais para a evolução da equipe no campeonato.

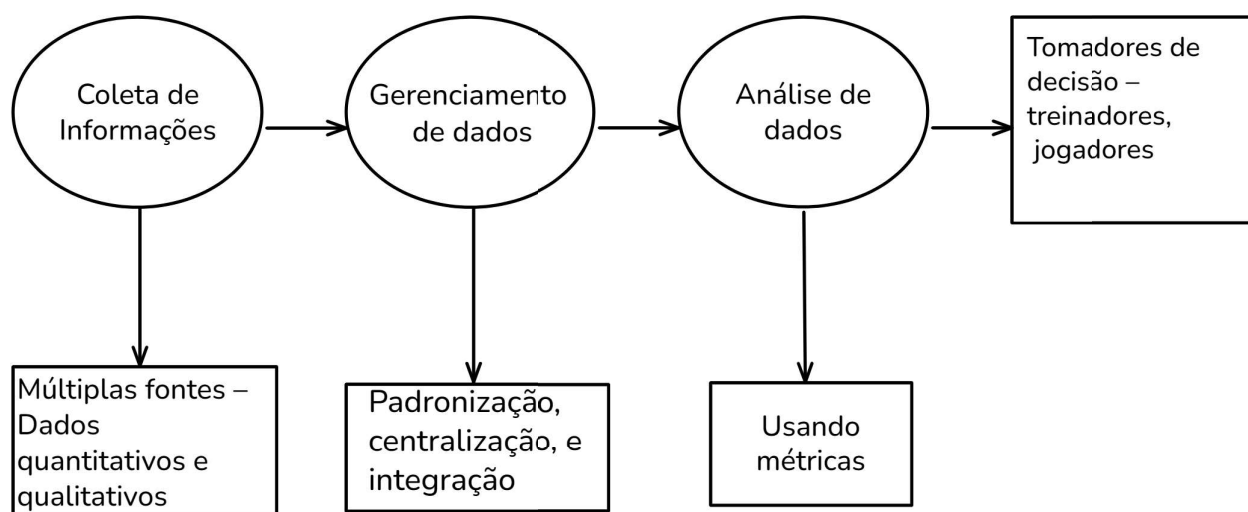
2.1.2 Conceito e modelos de avaliação de desempenho esportivo

A análise de desempenho esportivo é definida como a manipulação e manejo de dados estruturados, o uso de modelos de algoritmos de predição, e o uso de sistemas da informação. Tudo isso, com o objetivo de obter vantagens no processo de decisão dos caminhos que a equipe deve tomar ([MORGULEV; AZAR; LIDOR, 2018](#)).

Os dados podem ser coletados de diversas formas, mas precisam seguir um padrão para viabilizar a análise e o uso de métricas. Com isso, pode ser necessário a manipulação para a adequação dos dados coletados e transformá-los em dados estruturados.

Após o processo de coleta e padronização dos dados, são utilizadas métricas para realizar a análise. O resultado desse processo gera os insumos necessários para ajudar a tomada de decisão da comissão técnica. O fluxo deste processo está descrito na Figura 1.

Figura 1 – Framework de Análise Desportiva



Fonte: [Morgulev, Azar e Lidor \(2018\)](#), tradução nossa

2.1.3 Especificidades da análise notacional no futevôlei

O futevôlei, por ser um esporte similar ao vôlei de praia, mas com influências do futebol, compartilha algumas notações com o esporte original. Entretanto, possui ações e regras exclusivas que exigem uma abordagem notacional própria. Por isso, devemos expandir essa análise e entender as especificidades do futevôlei.

Estudos recentes têm buscado padronizar a análise notacional no futevôlei, destacando a importância da sistematização da notação para o esporte. A análise desenvolvida por [Borges, Senatore e Lamas \(2024\)](#), elucida de forma detalhada ações táticas e técnicas realizadas durante as partidas. Ações como saque, recepção, levantamento, ataque e bloqueio foram abordados no contexto específico do jogo, assim como os nomes dos golpes realizados.

Para realizar a análise notacional proposta neste trabalho, é preciso definir previamente o conjunto de ações que será padronizado e inserido na aplicação. Neste trabalho, adotamos como base o conjunto de ações e variáveis descritas na Tabela 1, que oferece uma representação sistematizada dos elementos observáveis durante uma partida de futevôlei.

Tabela 1 – Categorias e variáveis

Ações de Jogo		
Saque	— Jogador envolvido	— Zona de destino
	— Técnica do saque	— Trajetória da bola
	— Zona de execução	— Resultado do saque
Recepção/Defesa	— Jogador envolvido	— Zona de execução
	— Técnica da recepção/defesa	— Zona de destino
		— Resultado da recepção/defesa
Levantamento	— Jogador envolvido	— Zona de execução
	— Técnica do levantamento	— Zona de destino
		— Resultado do levantamento
1º toque de correção	— Jogador envolvido	— Zona de execução
	— Técnica do 1º toque	— Zona de destino
		— Resultado do 1º toque
Ataque	— Jogador envolvido	— Zona de destino
	— Técnica do Ataque	— Resultado do ataque
	— Zona de execução	
Bloqueio	— Jogador envolvido	— Zona de execução
	— Técnica do bloqueio	— Zona de destino
		— Resultado do bloqueio

Fonte: [Borges, Senatore e Lamas \(2024\)](#), tradução nossa

A Tabela 1 está organizada em três colunas: ação, entidade envolvida e resultado. Essas variáveis compõem uma notação detalhada e específica para o futevôlei, permitindo o registro preciso das jogadas. Para uma compreensão mais completa de cada uma dessas variáveis — incluindo imagens ilustrativas dos golpes e explicações adicionais — recomenda-se a consulta ao artigo de [Borges, Senatore e Lamas \(2024\)](#).

2.2 Sistemas Computacionais para Análise de Desempenho no Esporte

2.2.1 Introdução ao uso de sistemas computacionais no esporte

A evolução tecnológica alterou a forma com que o desempenho esportivo é analisado. O monitoramento das ações do esporte tornou-se muito mais preciso dada a evolução da capacidade computacional. A aplicação da tecnologia nos esportes e o incentivo à pesquisa nesta área vem crescendo a cada década.

Diferentes modalidades esportivas demandam tecnologias específicas para mensurar a evolução de atletas e equipes. Processos de análise que antes eram manuais e subjetivos, foram progressivamente substituídos por sistemas computacionais que oferecem maior padronização, velocidade e precisão. Tecnologias como *big data*, inteligência artificial, sensores (*wearables*), e sistemas de *sports tracking* são exemplos de inovações que

redefiniram a forma como o desempenho é coletado e interpretado. Embora cada tecnologia possua aplicações distintas, o objetivo central é comum: utilizar o poder computacional para promover a evolução do esporte e da análise de desempenho ([MORGULEV; AZAR; LIDOR, 2018](#)).

Entretanto, muitos desses serviços são caros e precisam de um grande investimento inicial para serem usados. Isto faz com que o nicho de usuários seja muito específico, focado em equipes profissionais com grande aporte financeiro para adquirir esses sistemas. São poucos os serviços abertos à comunidade, e a maioria precisa de uma equipe de suporte especializada para manusear da forma adequada.

2.2.2 Características dos sistemas de notação e análise estatística

No contexto dos esportes coletivos, existem diversos tipos de sistemas que auxiliam a análise desportiva. Mas, para este referencial teórico, o objeto de análise serão os sistemas computacionais com notação de ações de jogo para análise estatística. Estes sistemas contam com softwares especializados que conseguem trazer clareza sobre os dados do jogo para gerar uma análise estruturada sobre diversos pontos quantitativos da equipe na partida.

Sendo assim, existem algumas características em comum desses sistemas. O ponto principal é a padronização desta notação. Diferentemente de uma notação manual, em que cada pessoa pode determinar ações de jogo diferentes, ou formas de notação distintas, os sistemas de notação propõe padronizar estas ações e, assim, possibilitar a comparação de desempenho entre um jogo e outro.

Com os dados devidamente estruturados e padronizados, é possível realizar as análises estatísticas de forma mais precisa, sempre mantendo a equivalência de notações entre um jogo e outro. Outro ponto valioso é a visualização de dados e estatísticas. É comum que esses sistemas possuam uma estrutura gráfica para análise detalhada dos dados coletados. No vôlei, esporte mais parecido com o futevôlei, essa visualização gira em torno de taxa de erros, taxa de acertos, áreas da quadra onde ocorreram as ações, eficiência no levantamento, estatísticas de conversão de pontos, desempenho de bloqueio, entre outros dados detalhados que podem ser inferidos sobre atletas específicos ou toda a equipe ([GIATSI, 2023](#); [SILVA; LACERDA; JOÃO, 2014](#)).

2.2.3 Exemplos de sistemas comercializados atualmente

Nesta seção, abordaremos os sistemas existentes para realizar as notações e análises de desempenho e suas especificidades.

2.2.3.1 Data Volley 4

O Data Volley é considerado o software padrão-ouro na elite do voleibol mundial, utilizado pela maioria das seleções nacionais e clubes profissionais. Sua principal característica é um sistema de código textual extremamente detalhado, que permite registrar não apenas a ação (e.g., ataque), mas também suas múltiplas variáveis, como trajetória da bola, zona de origem e destino, e o resultado (e.g., ponto, erro, defesa). A entrada de dados é tipicamente realizada via teclado, exigindo que o operador memorize uma complexa sintaxe de códigos, o que resulta em uma longa curva de aprendizado.

A ferramenta oferece análises estatísticas profundas em tempo real, relatórios personalizáveis e integração total com vídeo. Apesar de sua completude, o Data Volley 4 representa um alto investimento financeiro. Seus planos de licenciamento, que podem variar de 299\$ (versão Lite) a 899\$ (versão Pro) por licença, tornam-no inacessível para modalidades com menor investimento.

2.2.3.2 SportsCode

O SportsCode, atualmente parte do ecossistema da empresa Hudl, é uma das ferramentas de análise de vídeo e desempenho mais utilizadas em esportes de alto rendimento, como futebol, basquete e rugby. Seu paradigma de funcionamento é centrado em uma linha do tempo de vídeo, onde o analista utiliza uma "janela de código" (*code window*) com botões customizáveis para marcar eventos ao vivo ou em pós-jogo.

O *software* é reconhecido por sua capacidade de criar *playlists* de lances específicos com poucos cliques, suas ferramentas de desenho sobre o vídeo para feedback visual e suas análises de matriz, que cruzam diferentes variáveis. Assim como o Data Volley 4, o *SportsCode* é uma solução profissional com custo elevado e uma curva de aprendizado considerável, voltada para analistas de desempenho em tempo integral. Sua complexidade e custo o distanciam das necessidades da comunidade do futevôlei, que demanda uma solução mais simples, intuitiva e de baixo custo.

2.2.3.3 Lince Plus

Segundo [Gabin et al. \(2012\)](#), o Lince Plus é um software multiplataforma de análise observacional desenvolvido para ser uma ferramenta genérica e adaptável a diversas atividades físicas e esportivas. Diferentemente de soluções focadas em um único esporte, ele permite que o próprio usuário crie seus "instrumentos de observação", que são painéis com botões personalizados para registrar os eventos de interesse.

Seu principal atrativo é ser um projeto gratuito e de código aberto (*open-source*), o que elimina a barreira financeira para aquisição, além de ser documentado cientificamente. Embora sua flexibilidade seja uma vantagem, ela também implica que toda a estrutura de

análise para um esporte específico, como o futevôlei, precisa ser construída pelo usuário. Por ser uma ferramenta de propósito generalista, não possui otimizações de fluxo de trabalho para a dinâmica de esportes de rali rápido, onde a velocidade de *input* sequencial é crítica.

2.2.4 Análise comparativa entre os sistemas computacionais existentes

A partir das características observadas nos sistemas Data Volley 4, SportsCode e Lince Plus, e considerando as necessidades específicas do futevôlei que foram discutidas, é possível fazer uma comparação entre as ferramentas existentes e as exigências do futevôlei. Esta comparação, presente na Tabela 2, permite evidenciar as limitações dos sistemas existentes e reforçar a importância do desenvolvimento de uma solução notacional específica e adaptada ao esporte brasileiro.

Tabela 2 – Comparação entre Sistemas de Análise de Desempenho

Critério	Data Volley 4	SportsCode	Lince Plus
Foco Principal	Notação textual especializada para Vôlei.	Análise de desempenho por vídeo com marcação em linha do tempo.	Ferramenta genérica para estudos de observação sistemática.
Método de Input	Sintaxe de código complexa via teclado.	Janelas de botões customizáveis que marcam o vídeo.	Painéis de observação (botões)
Curva de Aprendizado	Muito Alta. Exige memorização de códigos e treinamento longo.	Alta. Ferramenta profissional com muitas camadas de funcionalidade.	Média. A criação de um instrumento de análise eficaz exige planejamento.
Custo	Alto.	Alto.	Gratuito (software de código aberto).
Flexibilidade	Muito Baixa. A lógica é intrínseca às regras e fluxo do Vôlei.	Alta. Desenhado para ser adaptado a vários esportes.	Muito Alta. É sua principal característica.

Fonte: Elaborado pelo Autor (2025)

A partir da comparação, é possível avaliar a viabilidade de cada sistema para a análise notacional do futevôlei:

- **Data Volley 4** é inviável. Sua estrutura está muito atrelada ao vôlei, com dependência de regras como rodízio e posições fixas. Além disso, o custo elevado torna-o inacessível.
- **SportsCode** Apesar de flexível, seu uso requer alto investimento e conhecimento técnico. A complexidade da ferramenta e o fluxo baseado em linha do tempo dificultam seu uso por treinadores e atletas no dia a dia, principalmente em tempo real.
- **Lince Plus** é a opção mais próxima do ideal, mas ainda limitada. É gratuito e personalizável, permitindo uma adaptação completa. No entanto, sua operação dependeria de muitos cliques para realizar as notações. Para uma análise em tempo real de futevôlei, seria lenta e pouco fluida.

Sendo assim, conclui-se que nenhuma das soluções avaliadas atende de forma completa às demandas específicas do futevôlei. As ferramentas existentes apresentam aspectos como a limitação de adaptabilidade ao esporte, falta de agilidade na marcação em tempo real e valor não acessível para a comunidade.

2.3 Engenharia de Software

2.3.1 Ciclo de Vida: SCRUM/XP

2.3.1.1 Movimento Ágil

O ciclo de vida de um software é a definição das etapas e processos que são seguidos ao decorrer do seu desenvolvimento. A partir das limitações de modelos tradicionais e sequenciais para lidar com a natureza dinâmica dos projetos de software, surgiu no final da década de 1990 um conjunto de métodos de desenvolvimento rápido, hoje conhecidos como Métodos Ágeis. A filosofia por trás dessa abordagem foi consolidada no "Manifesto para o Desenvolvimento Ágil de Software", que se baseia em quatro valores fundamentais:

Indivíduos e interações mais que processos e ferramentas;
Software em funcionamento mais que documentação abrangente;
Colaboração com o cliente mais que negociação de contratos;
Responder a mudanças mais que seguir um plano.([MANIFESTO...](#), 2001)

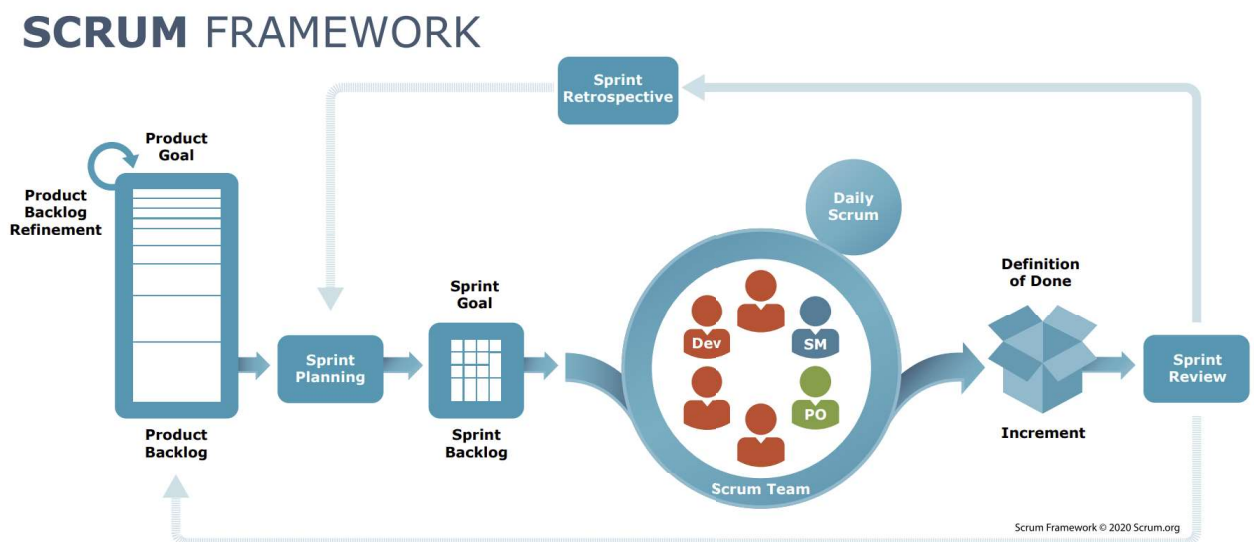
Esses valores são sustentados por princípios como a entrega incremental, o envolvimento contínuo do cliente e a simplicidade, tanto no software desenvolvido quanto no processo de desenvolvimento. Com isso, dentre as diversas implementações desses princípios, o SCRUM e o *Extreme Programming* (XP) destacam-se como duas das abordagens

mais influentes e utilizadas, principalmente em conjunto e dentro de um ciclo de vida de desenvolvimento de software.

2.3.1.2 SCRUM

O SCRUM é um *framework* para ajudar as equipes a desenvolver projetos complexos de forma adaptada. Ele não é uma metodologia rígida e completa, mas sim uma estrutura com interações, eventos e práticas que podem ser feitas para concluir as etapas de desenvolvimento. Com isso, ele se propõe a guiar e organizar a equipe com o objetivo de alcançar a meta do produto (*Product Goal*). A dinâmica do SCRUM está apresentada na figura 2.

Figura 2 – Poster do Framework Scrum



Fonte: [Scrum.org](https://www.scrum.org) (2025)

O processo, conforme ilustrado na figura, é estruturado a partir dos seguintes papéis, artefatos e eventos definidos:

Papéis: O Product Owner é responsável por maximizar o valor do produto, gerenciando o Product Backlog. O Scrum Master atua como um facilitador, garantindo que a equipe siga os princípios do Scrum. A Equipe de Desenvolvimento é o grupo que constrói o software.

Artefatos: O Product Backlog é a lista ordenada de tudo que é desejado para o produto. A partir dele, no planejamento de cada ciclo, é criado o Sprint Backlog, um plano que contém os itens do Product Backlog selecionados para a Sprint atual.

Eventos: O ciclo de vida é executado em Sprints, que são iterações com duração fixa (geralmente de duas a quatro semanas). Cada Sprint inicia com um Sprint Planning (Planejamento da Sprint), onde o trabalho a ser realizado é planejado. Diariamente, a

equipe se reúne no Daily Scrum para sincronizar o progresso e identificar impedimentos. Ao final do ciclo, o trabalho concluído é apresentado aos stakeholders na Sprint Review (Revisão da Sprint), e a equipe reflete sobre seu processo na Sprint Retrospective (Retrospectiva da Sprint) para identificar melhorias.

2.3.1.3 XP

Além da importância da organização da equipe definida no SCRUM, também é fundamental práticas de desenvolvimento de software eficazes. Por isso, surge a oportunidade de usar a metodologia *Extreme Programming* (XP), que são práticas com o objetivo de realizar implementações com alta qualidade, entrega contínua, e diminuir riscos ao decorrer do projeto.

O XP se baseia em valores como comunicação, simplicidade, *feedback*, respeito e coragem para implementar práticas e técnicas de desenvolvimento de software. Esta é a maior contribuição do XP para o SCRUM, preenchendo a lacuna de "como" desenvolver esse produto. Dentre suas práticas, destacam-se:

Integração Contínua: Esta prática consiste em integrar o trabalho de todos os desenvolvedores de forma frequente, muitas vezes ao longo da *sprint* e de forma mais atômica possível. Cada integração dispara a construção automatizada do sistema e a execução de todos os testes, garantindo que problemas de integração sejam descobertos e corrigidos imediatamente.

Programação em Pares: Prática em que a equipe se separa em duplas com o objetivo de melhorar a qualidade do código e equiparação do conhecimento entre toda a equipe. Para isso, esta prática conta com um "piloto" e um "navegador" por dupla. O piloto escreve o código, enquanto o navegador revisa, analisa e identifica possíveis problemas a cada passo do desenvolvimento.

Refatoração: É a reescrita e/ou reestruturação do código existente de forma gradual e sempre que for necessário. O objetivo é facilitar a manutenção, legibilidade, reduzir complexidade e deixar o código o mais limpo possível, evitando a degradação estrutural que pode vir a ocorrer com o tempo.

Testes Automatizados: Há várias formas de praticar os testes automatizados dentro do XP. Uma das formas de testes mais conhecidas é o TDD (*Test-Driven Development*), onde as funcionalidades são construídas a partir dos testes desenvolvidos previamente. Outra forma é fazer o uso da pipeline de integração contínua (*CI - Continuous Integration*), onde a cada entrega são executados uma série de testes para diminuir os defeitos que podem ser introduzidos no código. Testes automatizados possibilitam sua execução a qualquer hora do desenvolvimento, mantendo a consistência durante o desenvolvimento ou refatoração do código.

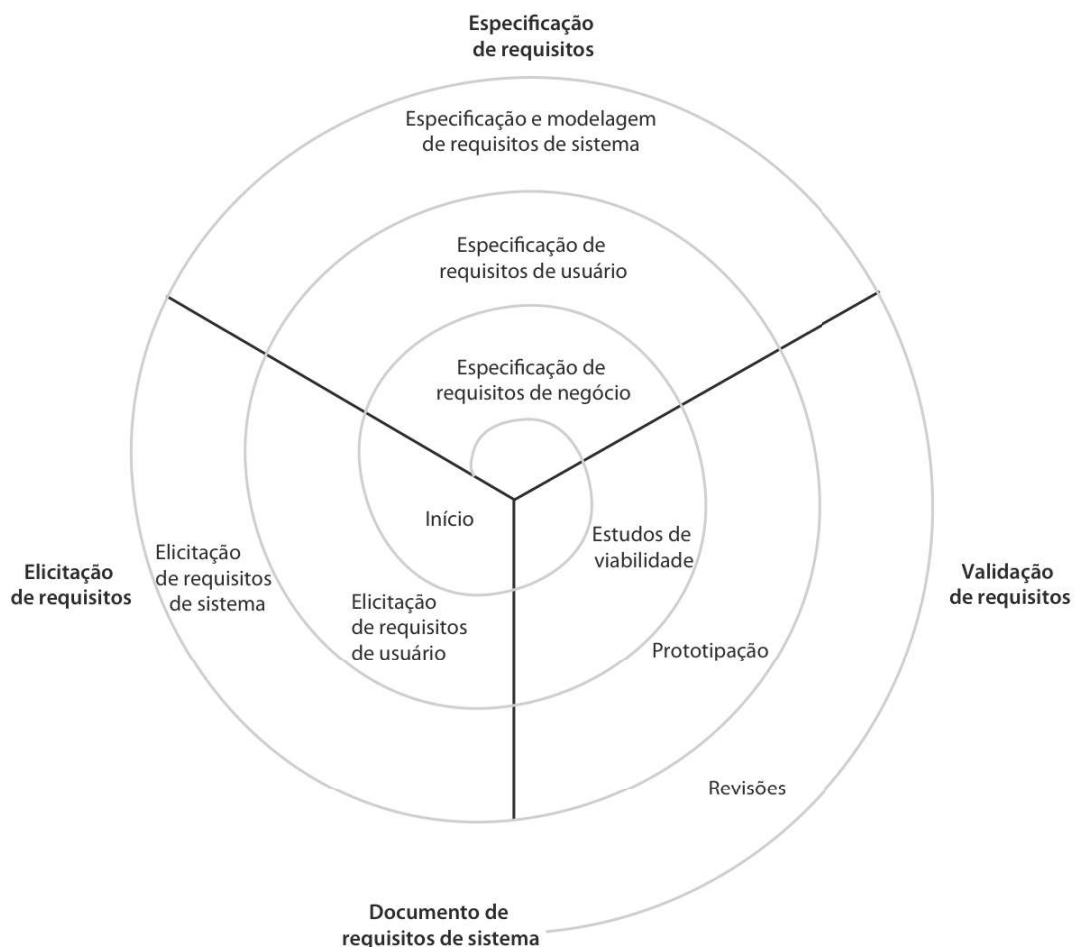
Sendo assim, a metodologia ágil *Extreme Programming* (XP) complementa o a organização de projeto estruturada pelo SCRUM com técnicas e práticas de programação fundamentais para a construção de um produto de software. A partir desta junção, é formado o ciclo de vida SCRUM/XP.

2.3.2 Engenharia de Requisitos

Segundo [Sommerville \(2011\)](#), um requisito é uma descrição de um serviço que o sistema deve realizar, ou uma restrição sobre esta operação. Eles são organizados em requisitos funcionais, que descrevem ações que o software deve executar, e requisitos não funcionais, que definem restrições de desempenho, segurança, usabilidade, ou sobre o próprio processo de desenvolvimento. Tudo que é regra externa à equipe de desenvolvimento também é um requisito não funcional.

As atividades que compõe a engenharia de requisitos são mostradas na figura 3 a seguir.

Figura 3 – Espiral do Processo de Engenharia de Requisitos



Fonte: [Sommerville \(2011\)](#)

Para estabelecer esses requisitos, existem quatro atividades relacionadas. A primeira é o estudo de viabilidade, que avalia se é possível desenvolver a partir dos aspectos técnicos e financeiros da equipe. Após isso, segue-se para a elicitación e análise, onde há o aprofundamento das necessidades dos *stakeholders* por meio de técnicas como entrevistas e observação. A terceira atividade é a especificação, quando são descritos formalmente os requisitos levantados, gerando um documento de especificação de requisitos.

Por fim, a validação verifica que o documento realmente cumpre com as expectativas do cliente e tenta garantir a consistência dos requisitos entre todas as equipes do projeto. Vale pontuar que a pesar de serem atividades apresentadas atômicamente, não é necessário realizá-las de forma sequencial, principalmente em projetos ágeis em que podem haver iterações e paralelismos ao longo do desenvolvimento.

2.3.3 Persistência: Banco de Dados

Um banco de dados é uma coleção de dados relacionados que representa algum aspecto do mundo real. Segundo [Elmasri e Navathe \(2011\)](#), um banco de dados é projetado, construído e populado com dados para um propósito específico, possuindo um público-alvo e aplicações de interesse. Para gerenciar essa coleção de dados, utiliza-se um Sistema de Gerenciamento de Banco de Dados (SGBD), que é o software responsável por definir, construir, manipular e compartilhar o banco de dados entre diversos usuários e aplicações, garantindo a persistência e a integridade das informações.

Atualmente, os dois principais tipos são os relacionais e os não relacionais (NoSQL). Os bancos de dados não relacionais abrangem diversos modelos, como de documentos, chave-valor ou grafos, e são frequentemente utilizados em cenários que exigem alta escalabilidade e flexibilidade de esquema. Contudo, o modelo relacional continua sendo o mais dominante para muitas aplicações do dia a dia. Este modelo organiza os dados em tabelas (ou relações), que são uma coleção de linhas e colunas, estabelecendo uma estrutura lógica e bem definida para a persistência e consulta dos dados ([ELMASRI; NAVATHE, 2011](#)).

No modelo relacional, a estrutura fundamental é a relação, que é representada como uma tabela. Cada tabela possui um conjunto de atributos (colunas), que definem as propriedades dos dados a serem armazenados, e cada tupla (linha) representa um registro único de dados que combina os valores para cada atributo. A integridade e o relacionamento entre as tabelas são mantidos por meio de chaves. A chave primária é um atributo (ou conjunto de atributos) que identifica unicamente cada tupla dentro de uma tabela, enquanto a chave estrangeira é um atributo em uma tabela que referencia a chave primária de outra, estabelecendo assim um vínculo formal entre elas e permitindo a construção de consultas complexas e consistentes sobre o conjunto de dados ([ELMASRI; NAVATHE, 2011](#)).

2.3.4 Qualidade de Software

2.3.4.1 Conceito de Qualidade de Software

Existem diversas definições de qualidade de software desenvolvidas por distintos autores que tentam descrever esse tema, sendo assim, não há uma concordância ou explicação única para este conceito. Entretanto, é possível suficientemente definir um software com qualidade aquele que está em conformidade com os requisitos funcionais e performáticos descritos e explicitamente documentados. Além de condizer com os padrões de desenvolvimentos acordados e as características implícitas de um software desenvolvido profissionalmente ([GALIN, 2004](#)).

Dessa forma, um software de qualidade é aquele que não apenas executa suas funções corretamente, mas também possui atributos desejáveis em sua construção, como padrões de projeto, testes e boas práticas implícitas ao desenvolvimento com qualidade.

2.3.4.2 Qualidade do Produto e em Uso

Para definir a qualidade de um software, estabeleceu-se a série de normas conhecidas como SQuaRE (*System and Software Quality Requirements and Evaluation*), definida na ISO/IEC 25000 ([International Organization for Standardization, 2014](#)). Estas normas são estruturadas em cinco categorias bem definidas:

- **ISO/IEC 2500n – Divisão de Gerenciamento:** Define os conceitos gerais e o vocabulário da família ISO 25000, além de guiar o uso das demais partes da norma.
- **ISO/IEC 2501n – Divisão de Qualidade:** Estabelece modelos de qualidade de produto de software, como o modelo que inclui características como usabilidade, confiabilidade, desempenho, entre outras
- **ISO/IEC 2502n – Divisão de Medição:** Fornece métricas e métodos para medir a qualidade de produtos de software com base nos modelos definidos.
- **ISO/IEC 2503n – Divisão de Requisitos de Qualidade:** Trata da especificação de requisitos de qualidade, possibilitando que stakeholders definam claramente as expectativas quanto à qualidade do produto.
- **ISO/IEC 2504n – Divisão de Avaliação da Qualidade:** Apresenta diretrizes e processos para avaliar a qualidade de software com base nos requisitos e medições definidos anteriormente.

O conjunto dessa norma forma um *framework* abrangente para realizar a análise de qualidade de produtos de software. Concomitantemente, funcionam como um guia para o desenvolvimento com foco em qualidade, definindo as características do produto

— como funcionalidade, desempenho e segurança —, além de processos para a avaliação do produto em relação aos seus requisitos.

2.3.4.3 Métricas de Software

Para analisar objetivamente as características de qualidade, é essencial realizar medições. Para isso, foram desenvolvidas as métricas de software, que consistem em atributos passíveis de quantificação. A medição se baseia em um valor numérico ou em um perfil associado a um componente, sistema ou processo (SOMMERVILLE, 2011).

Exemplos de métricas incluem: o tamanho de um produto em linhas de código; o índice Fog (GUNNING, 1962), que é uma medida da legibilidade de uma passagem de texto escrito; o número de defeitos relatados em um produto de software entregue, e o número de pessoas/dia requerido para desenvolver um componente de sistema. (SOMMERVILLE, 2011, p. 466)

As métricas de software podem ser classificadas a partir de duas perspectivas principais, conforme aponta Sommerville (2011): a natureza do atributo e a forma de medição.

A primeira perspectiva distingue atributos internos de atributos externos. É um desafio na medição de qualidade relacionar características perceptíveis ao usuário (externas), como usabilidade ou confiabilidade, a parâmetros mensuráveis do software em si (internos). Os atributos internos são aqueles que podem ser avaliados antes da execução do produto, analisando diretamente o código-fonte, as especificações ou a complexidade arquitetural. Por outro lado, os atributos externos são medidos a partir do comportamento do software em execução.

Essa distinção está diretamente ligada à segunda perspectiva, que classifica as medições em diretas ou indiretas. Frequentemente, os atributos internos são passíveis de medição direta, pois são independentes de outros atributos. Por exemplo, o número de linhas de código é medido diretamente a partir do próprio artefato. Em contrapartida, os atributos externos, como a "usabilidade", geralmente exigem uma medição indireta, pois seu valor é derivado da medição de outros atributos. A usabilidade, por exemplo, pode ser avaliada indiretamente por meio de métricas como a taxa de sucesso na conclusão de tarefas e o tempo gasto para realizar uma função. Assim, estabelece-se um vínculo entre propriedades concretas do software e a percepção de qualidade do usuário.

Contudo, a classificação das métricas não é suficiente para medir a qualidade do software. Por isso, é fundamental existir um processo estruturado para definir as métricas mais relevantes para os objetivos de interesse. É nesse contexto que se destaca a abordagem GQM.

2.3.4.4 Como Estabelecer Métricas: Modelo GQM

Para definir as métricas para a avaliação de qualidade do software, é necessário um modelo para o seu estabelecimento. A abordagem GQM (*Goal-Question-Metric*), oferece uma metodologia *top-down* que vincula as medições de software a metas de alto nível. Esta abordagem define que a qualidade é medida com base em objetivos específicos (*Goals*), dos quais são levantadas perguntas (*Questions*) para refinar a análise. Por fim, para cada uma dessas questões são definidas métricas (*Metrics*) para quantificar objetivamente a medida de qualidade do objetivo avaliado (BASILI; CALDIERA; ROMBACH, 1994).

- **Nível conceitual (Meta/ *Goal*)** Este é o nível mais alto e, por ser uma abordagem *top-down*, é o primeiro a ser definido. Os objetivos podem ser definidos a partir da visão de produto — como documentos, artefatos e entregas, que são produzidos durante o ciclo de vida do sistema — ; processos — ou seja, normalmente são atividades relacionadas ao software que tem prazos para serem efetuados, como planejamento de estratégia de desenvolvimento, processo de teste, entre outros —; ou recursos — que correspondem a insumos necessários para aquele projeto, como hardware, software, pessoas e ferramentas.
- **Nível Operacional(Questão/ *Questions*)** Consiste em uma série de questões que ajudam a detalhar a forma com que o objetivo será alcançado. São perguntas que permitem analisar a partir de quais aspectos é possível definir a qualidade desse objeto — seja ele um produto, processo ou recurso.
- **Nível Quantitativo(Métrica/ *Metrics*)** Este nível determina quais métricas objetivas ou subjetivas serão usadas para responder objetivamente às questões definidas. As métricas objetivas são aquelas baseadas em cálculos, expressões numéricas ou dados mensuráveis de forma matemática. Já as métricas subjetivas são medidas a partir da relatividade e estimativas pessoais ou de grupo, como o grau de satisfação do usuário, ou a percepção de qualidade sobre o objeto analisado.

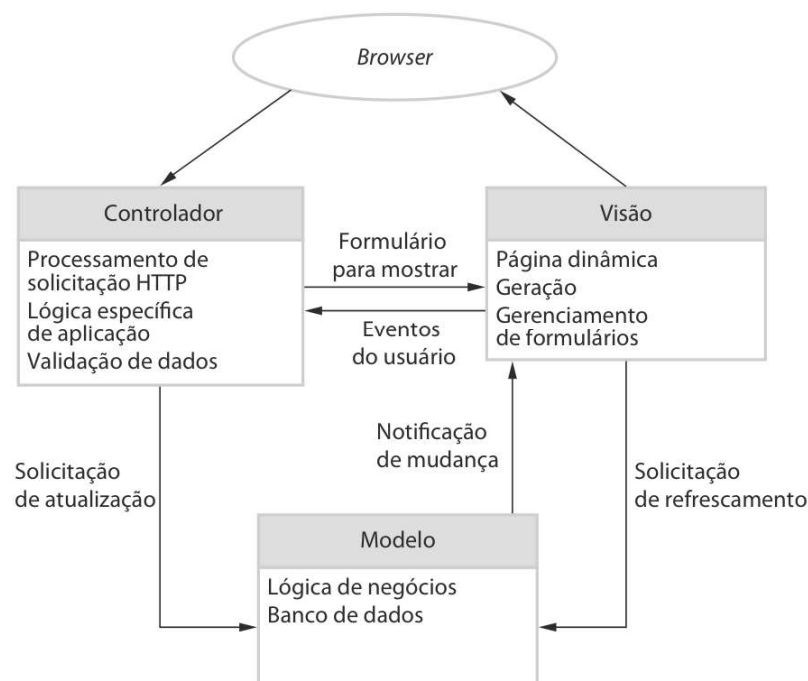
A partir dessa construção hierárquica guiada pelo GQM e definida em Basili, Caldiera e Rombach (1994), temos todos os objetivos, questões e métricas disponíveis para avaliar e analisar a qualidade do software. Com isso, a avaliação da qualidade é feita de forma inversa à sua construção, pois a interpretação parte dos resultados coletados das métricas para fornecer as respostas das perguntas que permitem avaliar se o objetivo foi alcançado. Ou seja, a análise segue o caminho inverso (*bottom-up*), partindo das métricas para definir a qualidade dos objetivos (Metas).

2.3.5 Arquitetura de Software

Segundo [Sommerville \(2011\)](#), a arquitetura de software é a relação entre as estruturas, componentes e sistemas de software. Esta arquitetura define como o sistema será construído e organizado. Isso impacta diretamente no desempenho, escalabilidade e complexidade do produto, por isso, deve ser uma das primeiras e principais etapas para a construção de um projeto de software. Com isso, existem diversos modelos, definições e estruturas de arquiteturas diferentes. Dentre os diversos padrões existentes, este referencial foca apenas naquelas que fundamentam o desenvolvimento deste trabalho, como a Arquitetura de Camadas e o modelo Cliente-Servidor.

A Arquitetura em Camadas é a organização do sistema em camadas lógicas distintas e sobrepostas, onde cada uma possui responsabilidades bem definida e presta serviço apenas para a camada imediatamente superior. Uma aplicação comum pode estruturar-se, por exemplo, em três camadas: a camada de interface, onde o usuário interage com o sistema, chamada de frontend. Outra chamada de Camada Lógica, responsável por desenvolver as funcionalidades e regras de negócio do sistema. E a Camada de Banco de Dados, que gerencia a persistência dos dados. Essa separação permite que as camadas sejam independentes e desacopladas, características positivas para o desenvolvimento e a manutenibilidade.

Figura 4 – Arquitetura de aplicações Web usando o padrão MVC



Fonte: [Sommerville \(2011\)](#)

Um exemplo de arquitetura em camadas é o padrão MVC (Modelo-Visão-Controlador), mostrado na Figura 4, onde há uma camada responsável pela interface do usuário, cha-

mado de "visão", outra camada responsável pelas requisições (e.g., HTTP, validação de dados), e uma última camada responsável pela lógica de negócios e banco de dados, chamada de "modelo".

O modelo Cliente-Servidor, por sua vez, representa uma implementação de distribuição da arquitetura em um ambiente de rede. Nesse modelo, o sistema é estruturado como um conjunto de serviços fornecidos por um ou mais servidores, que são acessados por múltiplos clientes. Tipicamente, a camada de interface (*frontend*) é o "cliente", enquanto as camadas lógicas e de banco de dados são "servidores". A comunicação entre cliente e servidor é estabelecida por meio de protocolos de comunicação, geralmente uma interface de programação de aplicações (API), seguindo o padrão RESTful.

2.3.6 UML - Linguagem de Modelagem Unificada

A Linguagem de Modelagem Unificada (UML) é uma linguagem gráfica para visualizar, especificar, construir e documentar os artefatos de sistemas de software. Segundo [Fowler \(2005\)](#), é uma abordagem que tem muito valor quando usada como ferramenta para facilitar a comunicação e compreensão entre os membros de uma equipe. Assim, não deve ser vista como um processo rígido a ser seguido, mas como uma caixa de utensílios que podem ser selecionados para aplicar a técnica mais adequada para resolver o problema em questão.

2.3.6.1 Técnicas Essenciais da UML

Há diversas técnicas e diagramas para muitos objetivos distintos, aqui estão as principais delas explicadas por uml:

- **Diagrama de Classes:** Utilizado para modelar a estrutura estática e o vocabulário fundamental do sistema, mostrando as classes e seus relacionamentos.
- **Diagrama de Sequência:** Essencial para visualizar as interações entre objetos ao longo do tempo, deixando claro quem chama quem e em que ordem.
- **Diagrama de Pacotes:** Usado para gerenciar a estrutura em larga escala, organizando as classes do sistema em grupos de alto nível.
- **Diagrama de Estados:** Fundamental para descrever o ciclo de vida de objetos que possuem comportamentos complexos e baseados em estados.

2.3.7 Testes de Software

Esta etapa do desenvolvimento de software é fundamental para a verificação de funcionalidades e defeitos do programa. Sendo assim, os testes podem ser usados para

dois objetivos: demonstrar que o software atende aos requisitos, podendo ser funcionais ou não funcionais, e descobrir se o software tem um comportamento errado ou inesperado, para encontrar defeitos no sistema (SOMMERVILLE, 2011). Assim, os testes podem ser classificados em alguns níveis de abstração, a explicação será expandida para alguns dos testes de desenvolvimento.

2.3.7.1 Testes Unitários

Segundo Sommerville (2011), são os testes mais simples de componentes, como funções ou classes de objetos. Esses testes são executados a partir de três partes que podem ser automatizadas, e, a partir delas, conseguimos testar a unidade de menor fração do programa.

- **Configuração:** É onde você instancia o caso de teste, com as entradas saídas ou configurações necessárias para o método ou objeto.
- **Chamada:** O objeto ou método que será testado deve ser chamado.
- **Afirmação:** É a comparação entre o resultado obtido com o resultado esperado. Aqui teremos a avaliação se o teste foi positivo ou se falhou.

2.3.7.2 Testes de Componente

Os componentes são diversos métodos ou objetos que funcionam em conjunto para executar uma funcionalidade específica do produto. Este tipo de teste tem o objetivo de validar este componente com o foco de analisar defeitos que surgem a partir de interações das unidades individuais (SOMMERVILLE, 2011). Esses erros de interface, segundo Sommerville (2011), podem ser classificados em três classes:

- **Mau uso de interface:** Ocorre quando um componente chamador utiliza a interface de outro componente de forma incorreta, como passar parâmetros de tipo ou em ordem errada.
- **Mau-entendimento de interface:** Acontece quando o componente chamador faz suposições incorretas sobre o comportamento do componente chamado, que não age como o esperado.
- **Erros de timing:** Ocorrem em sistemas de tempo real, quando o produtor e o consumidor de dados operam em velocidades diferentes, resultando no acesso a informações desatualizadas ou na leitura duplicada de um mesmo dado.

2.3.7.3 Testes de Sistema

Assim como o teste de componentes analisa a interação das partes unitárias do código, o teste de sistema analisa o comportamento entre as interações dos componentes do programa. O seu foco principal é validar o comportamento das versões emergentes do sistema, ou seja, testar funcionalidades que só se tornam aparentes quando os componentes são combinados.

2.3.7.4 Testes de *Release*

Os testes de *release* são realizados para validar uma entrega (*release*) do sistema, garantindo que ele atenda aos requisitos funcionais e não funcionais especificados. Segundo [Sommerville \(2011\)](#), esses testes são executados com a finalidade de mostrar ao cliente que o sistema está completo e funcional. Normalmente, é um teste de caixa-preta, ou seja, a validação é feita sem observar o código-fonte, focando apenas no comportamento do sistema.

2.3.8 Gestão e Configuração de Software - DevOps

Esta área de gestão e configuração de software tem o objetivo de estruturar as mudanças, versões, *releases* e processos de construção de sistemas. A evolução dessas práticas foi impulsionada pela maior agilidade das equipes e necessidade de colaboração contínua entre as pessoas e as entregas. A filosofia *DevOps* conta com práticas de desenvolvimento de software (Dev) com uma boa estrutura operacional (Ops), para garantir ciclos estruturados e encurtar o tempo entre as entregas. Há uma série de atividades definidas para esta faculdade, entretanto, vamos tratar apenas das principais para o desenvolvimento e ciclo de vida deste trabalho.

- **Controle de Versões:** Para gerenciar as diversas versões entre os colaboradores do projeto, e as várias versões do produto, foram construídos os sistemas de controle de versões. Esse gerenciamento é fundamental para apoiar o desenvolvimento independente e sem interferências entre os membros da equipe ou entre as diferentes versões do código-fonte ([SOMMERVILLE, 2011](#)). Atualmente, o Git é a ferramenta padrão do mercado, ele é usado a partir de uma cópia local do repositório (código) para cada desenvolvedor, incluindo o histórico de alterações e as ramificações do código-fonte. É uma ferramenta extremamente poderosa para a construção de versões paralelas à versão principal do sistema, e para facilitar a fusão (*merge*) de versões conflitantes em uma ramificação (*branch*) única.
- **Integração Contínua:** Esta é uma prática de desenvolvimento onde a equipe integra seu trabalho a um repositório compartilhado frequentemente. De preferencia

muitas vezes e várias vezes ao dia. Com o auxílio de ferramentas como o Git e GitHub (site para hospedagem de repositórios Git), essa integração fica ainda mais dinâmica, podendo ter processos (pipelines) automatizados, como a verificação por meio da construção (*build*) do sistema, ou a execução testes a cada incrementação de código no repositório, permitindo que os problemas sejam detectados o mais cedo possível.

3 Metodologia

Este capítulo descreve a abordagem metodológica empregada para a condução deste trabalho, e demonstra como a fundamentação teórica foi instanciada em ações práticas e aplicadas ao decorrer do desenvolvimento do projeto. Sendo assim, iniciamos com uma das partes mais importantes que é a metodologia de pesquisa, onde explica-se os métodos usados para a construção da pesquisa acadêmica.

3.1 Metodologia de Pesquisa

A estrutura da pesquisa foi construída com base nas classificações propostas por [Gerhardt e Silveira \(2009\)](#), que organiza o método científico quanto à abordagem do problema, à natureza dos objetivos e aos procedimentos técnicos empregados.

3.1.1 Quanto à Abordagem: Pesquisa Quantitativa

Segundo [Gerhardt e Silveira \(2009\)](#), a abordagem quantitativa se caracteriza pelo emprego de quantificação tanto na coleta quanto no tratamento das informações, utilizando técnicas objetivas ou procedimentos estatísticos para analisar os dados.

A escolha dessa abordagem justifica-se pela natureza da validação do software proposto. O objetivo central é mensurar objetivamente a acurácia e a viabilidade do sistema de notação em tempo real. Para isso, serão coletados dados numéricos, como a taxa de acerto das ações marcadas em comparação com um gabarito (*ground truth*), o tempo médio de registro por ação e a quantidade de erros de transcrição. As próprias estatísticas geradas pela ferramenta (e.g., percentual de conversão de ataques, eficiência de recepção) são, por definição, dados quantitativos.

3.1.2 Quanto à Natureza: Pesquisa Aplicada

A pesquisa aplicada, conforme descrito por [Gerhardt e Silveira \(2009\)](#), é motivada pela necessidade de resolver problemas concretos e práticos. Seu foco está na geração de conhecimentos para a intervenção na realidade.

Esta classificação é inerente ao propósito do projeto, pois o trabalho busca, além de desenvolver um conceito teórico, construir e validar uma ferramenta de software funcional com o objetivo direto de solucionar uma lacuna existente na comunidade do futevôlei.

3.1.3 Quanto aos Objetivos: Pesquisa Exploratória

O principal propósito da pesquisa exploratória é desenvolver, esclarecer e modificar conceitos e ideias, visando a formulação de problemas mais precisos ou hipóteses pesquisáveis para estudos posteriores (GERHARDT; SILVEIRA, 2009).

A adequação dessa classificação se dá pelo caráter inovador da solução no nicho do futevôlei. Embora a análise de desempenho exista em outros esportes, a aplicação de uma ferramenta de baixo custo com uma interface para a notação em tempo real no futevôlei é um campo pouco investigado. Este trabalho explora a viabilidade dessa nova abordagem, por isso, é exploratória por definição.

3.1.4 Quanto aos Procedimentos Técnicos

Os procedimentos técnicos referem-se aos métodos e técnicas utilizados para a coleta e análise dos dados. Este trabalho combina diferentes procedimentos de forma complementar.

- **Pesquisa Bibliográfica:** Realizada a partir de material já publicado, como livros, artigos científicos e teses, a pesquisa bibliográfica foi fundamental para a construção do referencial teórico. Por meio dela, foram investigados os sistemas de análise de desempenho existentes, os conceitos e tecnologias de software que fundamentam a ferramenta e a teoria sobre análise notacional no esporte.
- **Estudo de Caso e Pesquisa de Campo:** Segundo Gerhardt e Silveira (2009), o estudo de caso consiste no estudo profundo e exaustivo de um ou poucos objetos, permitindo seu amplo e detalhado conhecimento. Neste trabalho, o "caso" será o processo de utilização do software para a notação completa de uma partida de futevôlei. Este procedimento se caracteriza também como uma pesquisa de campo, pois ele busca reproduzir as condições e desafios da coleta de dados "no campo", ou seja, durante um jogo.

3.1.5 Metodologia de Desenvolvimento do Software

3.1.5.1 Ciclo de Vida

O ciclo de vida adotado para o desenvolvimento do software deste trabalho é um modelo híbrido, que combina o *framework* de gerenciamento de projetos SCRUM com práticas de engenharia de software da metodologia *Extreme Programming* (XP). O SCRUM é utilizado para estruturar o processo em ciclos iterativos, enquanto os princípios do XP, como a refatoração contínua e testes automatizados orientam as boas práticas para o desenvolvimento.

A seguir, detalha-se como o *framework* SCRUM foi adaptado para a realidade deste projeto.

- **Papéis:** Dada a composição da equipe, os papéis formais do SCRUM foram adaptados. A equipe de desenvolvimento conta apenas com o autor deste trabalho, que também atuou como *Product Owner* (PO). O especialista e Doutor na área da Ciência do Esporte atua como *Stakeholder*, fornecendo a visão do produto e validando os requisitos necessários e o orientador de Engenharia de Software atua como o Scrum Master, facilitando o processo, removendo impedimentos técnicos e incentivando a melhora contínua da equipe.
- **Artefatos:** O principal artefato utilizado é o *Backlog* do Produto. Este artefato centraliza todas as funcionalidades, requisitos, melhorias e correções desejadas para o sistema, sendo um documento dinâmico e priorizado continuamente. No início de cada ciclo, um conjunto de itens é selecionado para compor o *Backlog* da Sprint, representando a meta para aquele período.
- **Eventos:** O desenvolvimento é organizado a partir dos eventos listados abaixo e dividido por Sprints, que são ciclos de trabalho com duração fixa.
 - **Sprint:** Definiu-se um ciclo de duas semanas para cada Sprint, permitindo um ritmo constante de desenvolvimento e entrega.
 - **Sprint Planning:** No início de cada Sprint, reuni-se em uma reunião de planejamento para selecionar os itens de maior prioridade do *Backlog* do Produto e define os próximos passos.
 - **Comunicação Diária:** Em substituição ao *Daily Scrum*, foi adotada, quando necessária, uma comunicação assíncrona diária. Realizada por meio de mensagens para garantir o alinhamento ou resolução de impedimentos da equipe.
 - **Sprint Review:** Ao final de cada Sprint, uma reunião de revisão é conduzida com toda a equipe. Nela, o incremento do software desenvolvido é demonstrado, permitindo a validação das entregas, a coleta de *feedback* e a adaptação do *Backlog* do Produto para os ciclos seguintes.
 - **Sprint Retrospective:** Ao final de cada ciclo, é realizado uma reflexão sobre o processo, identificando pontos de melhoria no fluxo de trabalho, nas ferramentas e nas práticas de desenvolvimento para serem aplicados na próxima Sprint.

O projeto foi planejado com duas *releases* intermediárias e uma entrega final. Cada uma das entregas contam com um conjunto de *Sprints*. Essa estratégia permitiu a validação de versões mais robustas e integradas do produto em momentos chave do cronograma, consistente com a evolução constante esperada neste ciclo de vida.

3.1.5.2 Gerência de Configuração de Software

Esta seção é fundamental para estruturar o gerenciamento de versões do código e boas práticas de desenvolvimento. O gerenciamento facilita a integração de códigos conflitantes e possibilita a alteração no código de forma paralela, em ramos (*branches*) diferentes. Além disso, há a possibilidade de rastreamento das versões e releases entregues durante o ciclo de vida do produto (SOMMERVILLE, 2011). As ferramentas utilizadas neste trabalho são o Git e o GitHub.

- **Controle de Versões do Código:** Todo o código do projeto estará disponível em um repositório do GitHub. Todas as alterações e versões também estarão disponíveis nele. Para a organização e gerenciamento, há uma política de *branches* e *commits* que deve ser seguida.
 - **Política de *Branches*:** Seguindo o fluxo do GitHub Flow, a *branch* principal é chamada de **main** e sempre reflete a versão mais estável e testada do projeto (GitHub, 2025). Todo o trabalho, seja uma nova funcionalidade ou uma correção, é iniciado a partir dela. Para cada nova tarefa, uma *branch* com nome descritivo é criada (ex: **feat-reconhecimento-de-voz** ou **fix-cadastro**). Após a conclusão e validação, essa *branch* é mesclada de volta à **main** para ser implantada.
 - **Política de *Commits*:** Para a incrementação de código, segue-se a seguinte estrutura de comentários nos commits, baseado no padrão proposto por **Conventional Commits** (2025):
 - * **feat:** <mensagem> para novas funcionalidades ou melhorias no código.
 - * **fix:** <mensagem> para correção de bugs ou problemas.
 - * **style:** <mensagem> para ajustes que não têm impacto na execução do programa, como comentários, formatação ou lint.
 - * **docs:** <mensagem> para alterações na documentação.
 - * **refactor:** <mensagem> para refatoração de código que não altera a funcionalidade externa.
- **Integração Contínua (CI):** Para a realização do CI, o GitHub Actions será utilizado para criar uma *pipeline* automatizada. Essa *pipeline* executará a construção (*build*) do programa para verificar erros e rodará os testes previamente criados. A suíte de testes proposta para esta *pipeline* inclui testes unitários e de integração.

3.1.6 Engenharia de Requisitos

A Engenharia de Requisitos é o processo sistemático para definir, documentar e gerenciar os requisitos de um sistema de software. Seguindo as atividades propostas

por [Sommerville \(2011\)](#), este processo pode ser estruturado em: estudo de viabilidade, elicitação e análise, especificação e validação. Essas atividades foram integradas de forma adaptada para nortear este trabalho.

3.1.6.1 Estudo de Viabilidade

O estudo de viabilidade, que consiste em uma avaliação de alto nível para certificar que o projeto é viável, concluiu positivamente sobre a execução deste trabalho. Isso se deve pela natureza de pesquisa sem custos financeiros diretos, utilização de tecnologias de código aberto e infraestrutura computacional já existente. Além da capacidade técnica assegurada pela experiência da equipe de desenvolvimento e o envolvimento de um especialista da área da Ciência do Esporte.

3.1.6.2 Elicitação e Análise

A elicitação e análise é o processo de descoberta e compreensão dos requisitos a partir de diversas técnicas. Para este trabalho, foi essencial a colaboração com os *Stakeholders* para entender os problemas e identificar suas necessidades. Assim, foram empregadas duas técnicas principais:

- **Entrevistas:** Foram conduzidas entrevistas informais com o especialista da área de Ciência do Esporte. Atuando como o principal *stakeholder*, ele forneceu *insights* sobre as dinâmicas táticas do futevôlei, as métricas de desempenho mais relevantes e as funcionalidades que agregariam mais valor para o produto.
- **Etnografia:** Foi necessário compreender o contexto social e organizacional para pensar nos requisitos desse sistema de software. Por ser um esporte com baixo investimento financeiro, e tendo em conta a forma com que será utilizado, é primordial entender como isso impacta no funcionamento prático dele e seus requisitos.

3.1.6.3 Especificação

A especificação é a atividade de documentar os requisitos de forma clara e consistente. Após serem elicitados e analisados, os requisitos foram traduzidos em uma linguagem estruturada, servindo como a principal referência para o desenvolvimento. O principal artefato gerado nesta etapa foi o *Backlog* do Produto, fortalecendo o vínculo com o ciclo de vida SCRUM. Os requisitos foram categorizados em requisitos funcionais e requisitos não funcionais. O *Backlog* do Produto está demonstrado no capítulo de Solução Implementada.

3.1.6.4 Validação

A validação é o processo de garantir que os requisitos especificados estão completos, consistentes e se realmente atendem às necessidades e expectativas dos *stakeholders*. Além da verificação do primeiro documento gerado, a validação neste projeto também ocorreu de forma contínua e iterativa, uma prática consistente com a metodologia SCRUM. A validação acontece em cada evento de Sprint Review, realizada ao final de cada ciclo de desenvolvimento.

Ao apresentar um incremento de software, é possível validar na prática se os requisitos implementados permanecem alinhadas com a visão do produto, permitindo correções de curso de forma ágil a cada iteração das Sprints.

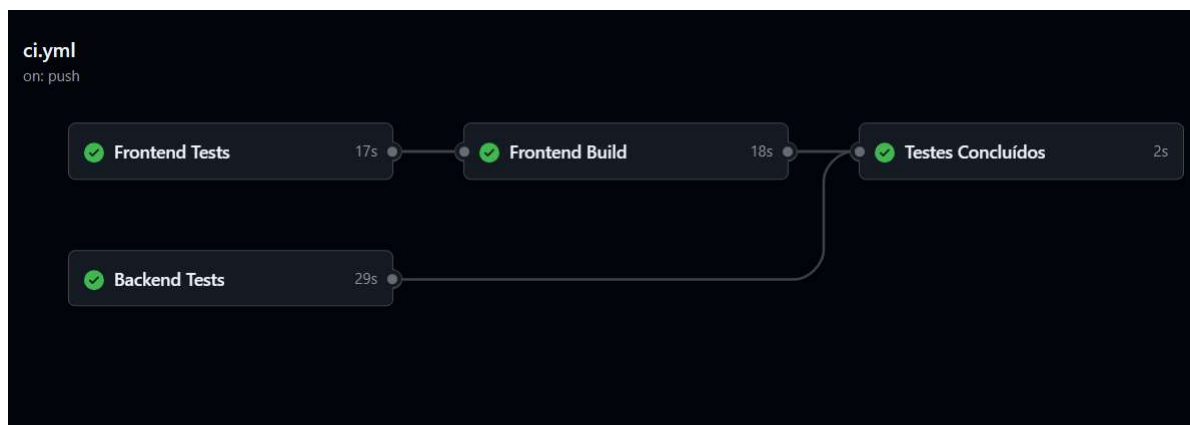
3.1.7 Metodologia de Avaliação e Validação

3.1.7.1 Plano de Testes

3.1.7.1.1 Testes de Desenvolvimento

A estratégia para realizar testes neste projeto segue os princípios do desenvolvimento ágil, focando em testes automatizados para uma rápida detecção de erros e falhas. O plano de testes contempla os três níveis de abstração: testes unitários, para validar os menores componentes de lógica, muito importantes no início do desenvolvimento; testes de componente, para verificar as interações entre os objetos que compõe a funcionalidade; e testes de sistema, usados quando o projeto está mais avançado para validar, inclusive, requisitos não funcionais.

Figura 5 – Pipeline de Integração Contínua (GitHub Actions)



Fonte: Elaborado pelo Autor (2025)

Para aplicar esta estratégia é fundamental o uso de Integração Contínua (CI), como mostra a Figura 5. Esta ferramenta está disponível pelo uso do Git e GitHub,

onde cada alteração do código é verificado pela execução de uma suíte de testes com todos os testes necessários para validar a nova mudança. Este processo é executado em um servidor de construção, que irá compilar a aplicação e rodar todos os testes pré-programados automaticamente. Caso falhe qualquer parte deste processo, a alteração é considerada "quebrada" e a equipe deve consertar os problemas para conseguir integrar o código à base do projeto. Isso mantém a base do código sempre estável e funcional.

3.1.7.1.2 Teste de *Release* e Desenho do Estudo de Caso

Para validar e testar a solução, além dos testes de desenvolvimento mostrados na seção anterior, devemos definir como será conduzido o estudo de caso e os testes de *releases*, também chamado de teste de validação. Para realizá-lo, precisamos de uma metodologia sistematizada que será seguida para guiar ou simular o cenário com as condições reais de uso, verificando sua viabilidade.

O estudo será realizado em um ambiente de rede local, gerada por um computador pessoal, que será acessada por um dispositivo móvel como um *tablet*. O servidor, representado pelo computador pessoal, é responsável pelo *backend* e banco de dados do serviço, enquanto o dispositivo é cliente nessa rede. Durante a realização do estudo, contará com as seguintes regras:

1. Configuração do servidor e cliente previamente feitas, conforme as instruções de uso.
2. As notações deverão ser realizadas em tempo real, ao vivo ou por vídeo (sem pausas, cortes ou alteração na velocidade de reprodução do vídeo).
3. Apenas 1 (um) operador (responsável pela notação no dispositivo móvel).

3.1.7.2 Plano de Métricas e Definição do Modelo GQM

Para que a avaliação da qualidade do software seja direcionada e objetiva, é fundamental estabelecer um plano de métricas estruturado. Para isso, este projeto adotará a abordagem GQM (*Goal-Question-Metrics*) desenvolvida por Basili, Caldiera e Rombach (1994) e fundamentada no referencial teórico. Esta metodologia garante que o projeto seja guiado por métricas fundamentadas em metas de alto nível, relevantes para que o objetivo final seja alcançado.

A seguir, é apresentado o modelo GQM que guiará o projeto durante todo seu ciclo de vida:

Objetivo:

Analisar a eficiência da sistematização do sistema de notação de futevôlei com o

propósito de avaliar sua viabilidade para uso em tempo real, da perspectiva do anotador, no ambiente de uma partida.

Questões e Métricas: Q1: Qual o tempo médio necessário para registrar uma ação completa (local da quadra + tipo de ação) durante uma partida?

M1.1: Tempo em segundos por registro de ação.

Q2: Qual a taxa de acerto da funcionalidade de transcrição de voz para as ações de jogo predefinidas?

M2.1: Percentual de transcrições de voz corretas (comandos de voz reconhecidos corretamente / total de comandos de voz).

Q3: Qual a curva de aprendizado do operador ao utilizar o sistema?

M3.1: Número de erros de operação por partida (contagem de ações que precisaram ser desfeitas ou corrigidas).

Q4: Qual o nível de satisfação do operador com a usabilidade e facilidade de manusear a ferramenta?

M4.1: Pontuação em uma escala Likert de 1 a 5 para perguntas sobre facilidade de uso e satisfação em um questionário.

Objetivo:

Analisar o desempenho de jogadores ou equipes de futevôlei com o propósito de levantar estatísticas descritivas sobre as ações de jogo, da perspectiva do treinador ou analista, no ambiente de uma partida.

Questões e Métricas:

Q1: Qual a qualidade do saque por jogador?

M1.1: Número de *Aces*(saque sem defesa) por jogador.

M1.2: Número de erros de saque por jogador.

M1.3: Percentual de aproveitamento do saque:

$$\frac{\text{Total de Saques} - \text{Erros de Saque}}{\text{Total de Saques}} \times 100$$

Q2: Qual a qualidade de ataque (e.g., Shark, Cabeça, Ombro) por jogador?

M2.1: Contagem de pontos de ataque por jogador.

M2.2: Contagem de erros de ataque por jogador.

M2.3: Percentual de aproveitamento do ataque:

$$\frac{\text{Total de Ataques} - \text{Erros de Ataque}}{\text{Total de Ataques}} \times 100$$

M2.4: Percentual de eficiência do ataque:

$$\frac{\text{Pontos de Ataque} - \text{Erros de Ataque}}{\text{Total de Tentativas de Ataque}} \times 100$$

Q3: Qual a eficácia defensiva de cada jogador (recepção e defesa)?

M3.1: Contagem de erros de recepção e defesa que resultaram em ponto para o adversário.

M3.2: Percentual de aproveitamento defensivo:

$$\frac{\text{Total de Defesas} - \text{Erros de Defesa}}{\text{Total de Defesas}} \times 100$$

Q4: Qual a distribuição espacial dos acertos e erros de ataque e saque na quadra?

M4.1: Mapeamento da direção do ataque ou saque (quadrante da quadra) para ações que resultaram em ponto.

M4.2: Mapeamento da direção do ataque ou saque (quadrante da quadra) para ações que resultaram em erro.

Com base no modelo GQM, o desenvolvimento do software é orientado para o registro das ações de jogo em tempo real, sob a perspectiva do anotador. O objetivo é que a ferramenta cumpra com a velocidade exigida pela dinâmica do futevôlei e facilite seu uso, minimizando ao máximo a carga cognitiva do operador. Para isso, deve contar com uma interface intuitiva e com o máximo de fluxos automatizados possíveis, para diminuir a interação humana sem perder dados relevantes da partida. Foram realizadas diversas adequações funcionais a partir de testes ao longo do desenvolvimento para atingir esse objetivo.

Também foram definidas métricas utilizadas para avaliar atletas e equipes durante uma partida, uma competição ou durante todo seu histórico de jogos. Elas são essenciais para definir quais estatísticas o software deve ser capaz de gerar a partir dos dados coletados.

4 Solução Implementada

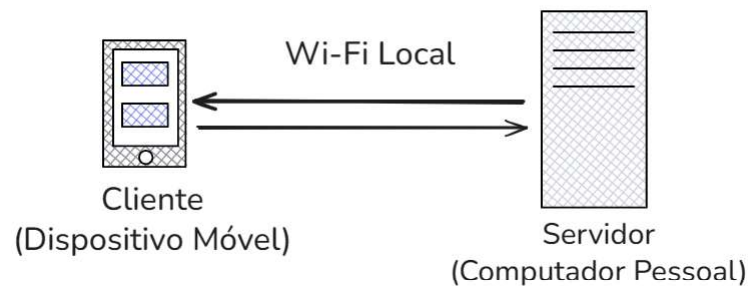
Este capítulo apresenta a implementação do sistema proposto a partir do problema levantado neste trabalho. As tecnologias são instanciadas de forma direta, com base no referencial teórico demonstrado e nas metodologias definidas.

4.1 Arquitetura da Solução

A estrutura arquitetural foi desenhada para atender aos critérios de marcação em tempo real, portabilidade, facilidade de uso, e usabilidade *offline*. Para isso, foi definido uma Arquitetura Cliente-Servidor e em Camadas. Ela é composta por três componentes que interagem entre si, formados por: um *frontend*(Cliente), um *backend*(Servidor) e um banco de dados.

A interação entre esses componentes é fundamental para o funcionamento offline do sistema, e está representada visualmente na Figura 6.

Figura 6 – Comunicação Cliente-Servidor Local



Fonte: Elaborado pelo Autor (2025)

Conforme ilustrado na Figura 6, o sistema realiza sua comunicação da seguinte forma:

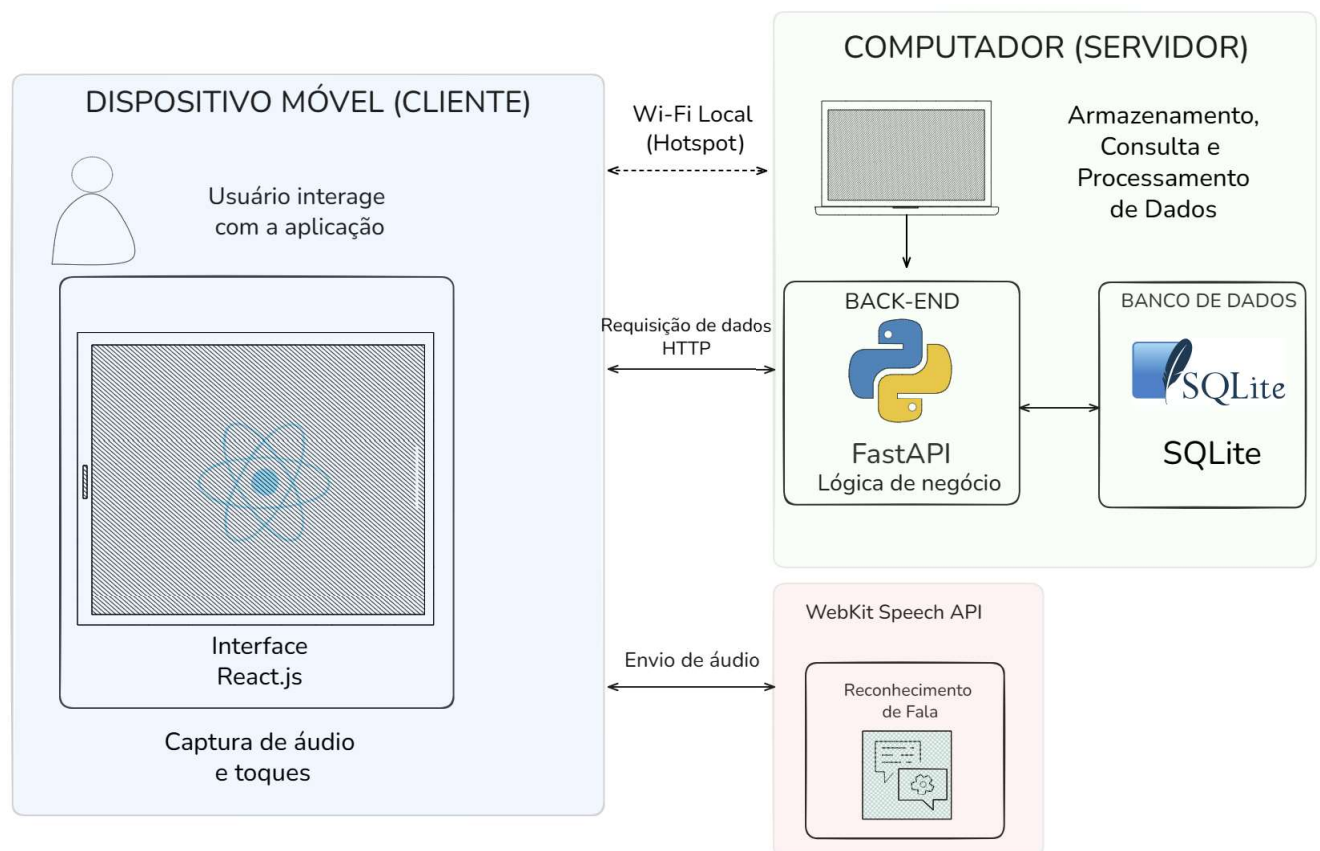
- **Cliente (Dispositivo Móvel):** É a interface com o usuário, executada em um navegador web de um *smartphone* ou *tablet*. Este componente é responsável por capturar todas as interações com o usuário, por exemplo os toques na tela, captura de áudio e exibições de dados.

- **Servidor (Computador Pessoal):** Atua como o núcleo do sistema, hospedado em um computador pessoal. Ele contém a lógica de negócio no Backend, realiza o processamento de áudio, gerencia o banco de dados e serve a aplicação web para o cliente.
- **Comunicação:** É estabelecida por meio de uma rede Wi-Fi local, criada pelo próprio computador servidor ao ser configurado como um *hotspot* móvel. Essa abordagem é fundamental para ser independente de qualquer infraestrutura de rede e hospedagem externa, ou da necessidade de acesso à internet, cumprindo com o requisito de portabilidade do sistema.

4.1.1 Descrição dos Componentes de Software

A seguir, cada componente da arquitetura é detalhado, apresentando suas responsabilidades, as tecnologias selecionadas e as justificativas para sua instanciação. A Figura 7 ilustra esses componentes da arquitetura geral do sistema.

Figura 7 – Componentes de Software da Aplicação



Fonte: Elaborado pelo Autor (2025)

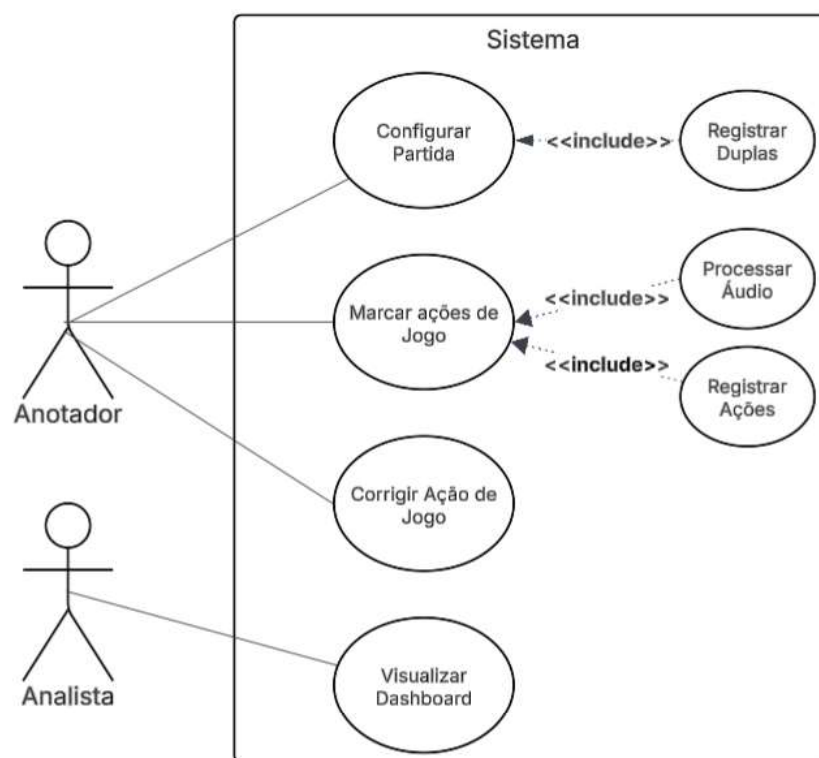
- Frontend (Cliente Web):
 - **Tecnologia:** React.js.
 - **Responsabilidades:** Renderizar a interface gráfica da quadra, capturar eventos de toque e áudio, enviar os dados ao backend, além de exibir graficamente o dashboard com as estatísticas de partidas.
 - **Justificativa:** O framework React.js foi escolhido por sua facilidade em criar interfaces responsivas e dinâmicas, permitindo a adaptabilidade da aplicação em diferentes tamanhos de tela.
- Backend (Servidor da Aplicação):
 - **Tecnologia:** Python com FastAPI.
 - **Responsabilidades:** Prover uma API RESTful para a comunicação com o frontend, orquestrar a lógica de negócio e gerenciar as operações no banco de dados.
 - **Justificativa:** Os três fatores principais foram: primeiramente, o Python é amplamente utilizado para ciência de dados no esporte; em segundo lugar, ele facilita evoluções futuras para inserir modelos de IA (como transcrição de fala ou visão computacional); e, por fim, é um framework com boa performance, que permite a construção de um serviço em tempo real.
- Módulo de Transcrição de Áudio:
 - **Responsabilidades:** Receber o áudio do frontend, processá-lo e retornar sua transcrição em formato de texto.
 - **Tecnologia:** Webkit Speech API.
 - **Justificativa:** Esta tecnologia foi selecionada por sua simplicidade na implementação e para validar a viabilidade do fluxo antes de implementar um modelo mais robusto.
- Banco de Dados
 - **Tecnologia:** Banco de Dados Relacional (SQLite).
 - **Responsabilidades:** Armazenar de forma persistente e estruturada todos os dados gerados pelo sistema, como informações de partidas, pontos e as ações detalhadas de cada ponto.
 - **Justificativa:** O banco de dados relacional foi escolhido por sua forma estruturada de armazenamento, que é adequada para o contexto dos tipos de dados neste trabalho. O SQLite é uma opção leve que prioriza a velocidade de escrita e leitura dos dados, essencial para o contexto em tempo real.

4.2 Requisitos de Software

4.2.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso ilustra as interações entre os atores (usuários) e as principais funcionalidades do sistema. Para este projeto, identificamos um ator principal: o Anotador, e um ator secundário: o Analista ou Estatístico. Ambos os atores podem ser desempenhados por uma mesma pessoa, como alguém da equipe técnica ou atleta de fora do campo. A seguir, é apresentada a descrição do caso de uso mostrado na Figura 8.

Figura 8 – Casos de Uso



Fonte: Elaborado pelo Autor (2025)

- **Atores principais:**

- **Anotador:** Usuário que interage diretamente com o sistema, anotando os eventos da partida em tempo real.
- **Analista/Estatístico:** Usuário que analisa os dados por meio das estatísticas geradas pelo sistema.

- **Casos de uso principais:**

1. **Configurar Partida:** O Anotador registra os detalhes da partida, como os jogadores de cada dupla e o nome das duplas. A data e hora também são registradas.

2. **Marcar Ações de Jogo:** O Anotador registra as ações de jogo, como saques, passes, ataques ou defesas. Deve ser feito por meio do toque na tela e do uso da fala para descrever a ação.
3. **Corrigir Ação de Jogo:** O Anotador pode corrigir uma ação da partida que foi registrada incorretamente.
4. **Visualizar o Dashboard:** O Analista/Estatístico pode acessar o dashboard para visualizar as estatísticas geradas da partida.

4.2.2 Backlog do Produto

O Backlog do Produto é uma lista ordenada e dinâmica de todas as funcionalidades, requisitos e melhorias desejadas para o software. É um artefato essencial para o desenvolvimento ágil. Os itens do backlog estão escritos na forma de Histórias de Usuário (*User Stories*), para manter o foco no valor entregue ao usuário final.

Os itens foram agrupados em Épicos para melhor organização e priorizados da seguinte forma:

- **Essencial:** Requisitos fundamentais para constituir o Mínimo Produto Viável (MVP).
- **Importante:** Funcionalidades que agregam valor significativo, mas não são críticas para o MVP.
- **Desejável:** Melhorias e funcionalidades que podem ser implementadas caso haja tempo e recurso. Mas não são prioritárias no desenvolvimento deste trabalho.

4.2.2.1 ÉPICO 1: Gestão de Partidas

Este épico cobre as funcionalidades básicas de criação e gerenciamento de partidas.

Tabela 3 – Histórias de Usuários para o Épico de Gestão de Partidas.

ID	Histórias de Usuários	Prioridade
US01	Como Anotador , eu quero criar uma nova partida , definindo um nome para ela, a data/hora e os nomes dos jogadores de cada dupla, para que eu possa iniciar uma nova sessão de notação com os dados corretos.	Essencial
US02	Como Anotador ou Técnico , eu quero visualizar uma lista de todas as partidas salvas , para que eu possa selecionar uma partida anterior para análise ou continuar uma notação.	Importante
US03	Como Anotador , eu quero excluir uma partida da lista, para manter meus registros organizados.	Desejável

Fonte: Elaborado pelo Autor (2025)

4.2.2.2 ÉPICO 2: Notação em Tempo Real

Este épico contém as principais funcionalidades do sistema, como interface e o processo de notação de ações.

Tabela 4 – Histórias de Usuários para o Épico de Notação em Tempo Real

ID	Histórias de Usuários	Prioridade
US04	Como Anotador , eu quero iniciar uma sessão de notação de uma partida existente, para começar a registrar os pontos do jogo.	Essencial
US05	Como Anotador , eu quero marcar em uma grade visual da quadra o local de destino de um saque ou ataque, para que a distribuição espacial das jogadas seja registrada.	Essencial
US06	Como Anotador , eu quero usar minha voz para registrar o tipo de ação (e.g., "cabeça", "peito", "coxa"), para que a notação seja rápida e eu possa manter o foco no jogo.	Importante
US07	Como Anotador , eu quero finalizar a marcação de um ponto, indicando a equipe vencedora e o motivo (e.g., ponto de ataque, erro de recepção), para que as estatísticas de eficiência sejam calculadas corretamente.	Importante
US08	Como Anotador , eu quero corrigir ou excluir a última ação registrada de forma rápida, caso eu cometa um erro durante a marcação de um rali.	Importante
US09	Como Anotador , eu quero selecionar o jogador que realiza uma ação, para que as estatísticas possam ser atribuídas corretamente.	Desejável
US10	Como Anotador , eu quero visualizar a sequência de ações do ponto atual na tela, para ter um feedback imediato e verificar a precisão da minha marcação.	Desejável
US11	Como Anotador , eu quero ter a opção de usar botões na tela como uma alternativa à voz para registrar o tipo de ação, para fins de comparação de usabilidade e uso em ambientes ruidosos.	Desejável

Fonte: Elaborado pelo Autor (2025)

4.2.2.3 ÉPICO 3: Visualização e Análise de Dados

Este épico contém as funcionalidades para a apresentação dos dados coletados em um dashboard útil para o Analista.

Tabela 5 – Histórias de Usuários para o Épico de Visualização e Análise de Dados

ID	Histórias de Usuários	Prioridade
US12	Como Anotador , eu quero ver o placar atual do jogo sendo atualizado em tempo real, para acompanhar o andamento da partida.	Essencial
US13	Como Analista , eu quero visualizar as estatísticas de aproveitamento de saque e ataque por equipe, para ter uma visão do desempenho ofensivo.	Essencial
US14	Como Analista , eu quero visualizar a quadra mostrando a distribuição espacial dos acertos e erros de ataque e saque, para identificar padrões táticos.	Essencial
US15	Como Analista , eu quero analisar as estatísticas individuais de eficiência de saque de cada jogador (Aces, erros, aproveitamento), para identificar pontos fortes e fracos.	Importante
US16	Como Analista , eu quero analisar o aproveitamento de ataque individual de cada jogador, detalhado por tipo de golpe (Shark, Cabeça, etc.), para orientar treinamentos específicos.	Importante
US17	Como Analista , eu quero analisar a eficácia defensiva de cada jogador , identificando a frequência de erros de recepção/defesa.	Importante
US18	Como Anotador , eu quero exportar os dados brutos de uma partida (e.g., para CSV), para poder realizar análises mais aprofundadas em outras ferramentas.	Importante
US19	Como Analista , eu quero filtrar as estatísticas do dashboard por jogador ou equipe , para focar a análise em um atleta específico.	Desejável

Fonte: Elaborado pelo Autor (2025)

4.2.2.4 ÉPICO 4: Requisitos Não Funcionais

Este épico descreve os requisitos não funcionais do sistema, como restrições de usabilidade e qualidade.

Tabela 6 – Requisitos Não Funcionais.

ID	Histórias de Usuários	Prioridade
RNF01	O sistema deve operar sem conexão à internet , conectado a uma rede local (Wi-Fi).	Essencial
RNF02	A interface de notação deve ser responsiva e otimizada para uso em dispositivos móveis (<i>tablets</i>).	Essencial
RNF03	Os dados de todas as partidas e ações devem ser armazenados de forma persistente em um banco de dados no servidor.	Essencial
RNF04	O tempo para a notação de uma ação completa deve ser suficientemente rápida para não prejudicar o fluxo de jogo, permitindo a notação em tempo real.	Importante
RNF05	O reconhecimento de voz deve possuir uma acurácia aceitável em um ambiente com ruído moderado, típico de uma quadra de futevôlei.	Desejável

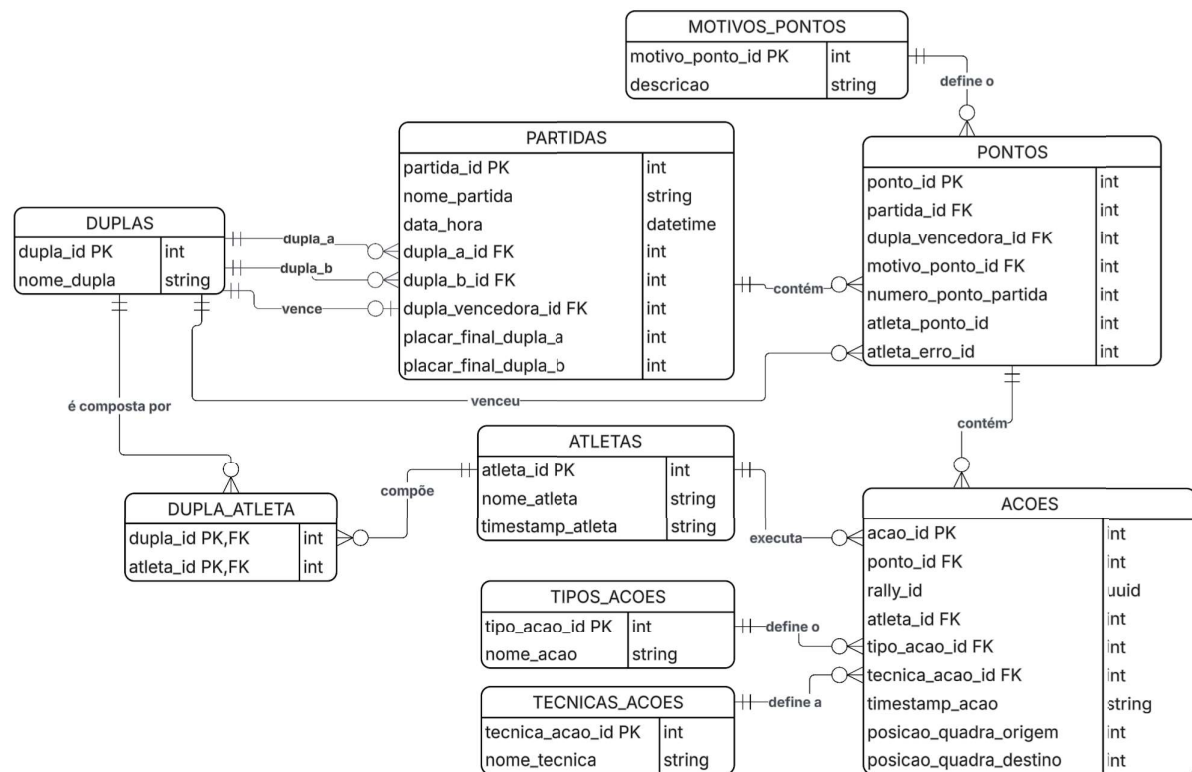
Fonte: Elaborado pelo Autor (2025)

4.2.3 Modelo de Dados: Diagrama Entidade-Relacionamento

Esta seção apresenta a base para a construção de um banco de dados relacional. Para isso, foi elaborado um Diagrama Entidade-Relacionamento (DER) que modela conceitualmente as principais entidades e seus relacionamentos dentro do sistema. Ele é fundamental para a construção de um banco de dados físico com integridade e consistência dos dados.

A figura 9 a seguir, apresenta o DER proposto a este trabalho.

Figura 9 – Diagrama Entidade-Relacionamento



Fonte: Elaborado pelo Autor (2025)

4.2.4 Apresentação da Aplicação

Para o desenvolvimento do sistema, foi usado um repositório remoto dentro do GitHub ¹, no perfil do autor do trabalho. O código é *open-source* e regido sob a licença *GNU General Public License versão 3.0* (GPL-3.0), para possibilitar o uso livre do sistema e garantir que a comunidade possa contribuir com melhorias a qualquer momento, seguindo as regras e políticas do repositório.

A aplicação será apresentada através da visão do usuário para a compreensão das funcionalidades e da usabilidade do sistema. As telas mostradas estão nas dimensões de um *tablet* comum, mas são redimensionadas conforme a tela do dispositivo, adaptando-se a telas de diversos tamanhos.

¹ Disponível em: <https://github.com/FelipeDireito/FTV-Scout>

Figura 10 – Tela Inicial

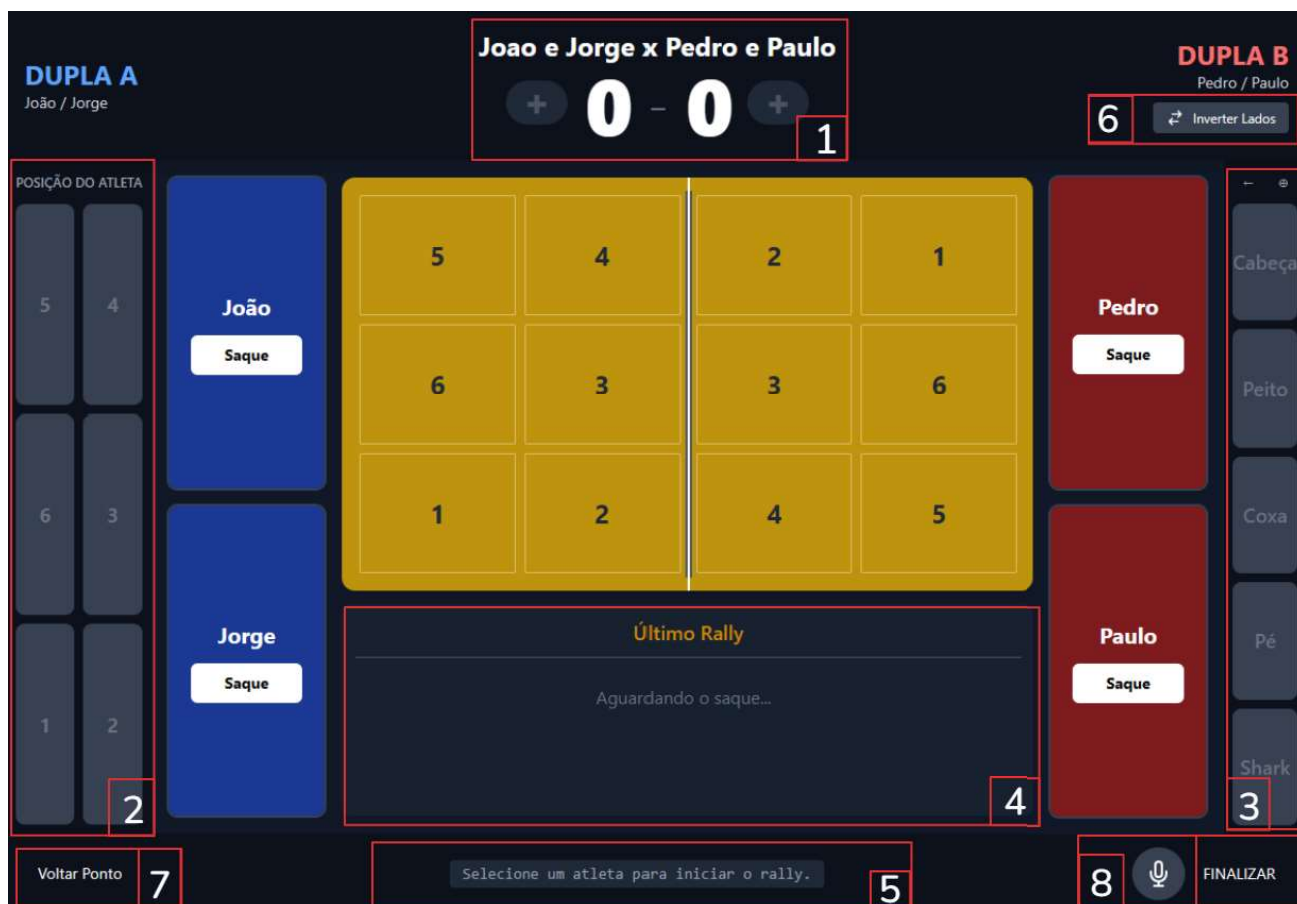


Fonte: Elaborado pelo Autor (2025)

A Figura 10 mostra a tela inicial da aplicação, onde o usuário pode criar uma nova partida ou ir para a análise de partidas já registradas. Ao lado direito, há as três últimas partidas registradas para facilitar o acesso. Dentro da análise de partidas, podemos selecionar uma partida específica ou visualizar o histórico completo de partidas de um atleta ou dupla.

A Figura 11 mostra a tela de notação da partida, onde as ações de jogo são coletadas para embasar todas as estatísticas geradas pelo sistema. Por isso, é a tela mais importante da aplicação, e, concomitantemente, a tela com mais detalhes e informações.

Figura 11 – Tela de Notação da Partida



Fonte: Elaborado pelo Autor (2025)

A partir da Figura 11, podemos observar os principais componentes da tela de notação, explicados a seguir e numerados de acordo com a imagem:

1. **Painel de Placar:** Seção para exibir o placar atual da partida. Ao lado de cada dupla, há botões para acrescentar pontos (+). Ao clicar, abre-se um *pop-up* para selecionar o motivo do ponto (ataque, saque, etc.).
2. **Posição do Atleta na Quadra:** Este componente tem a mesma função de selecionar a posição da ação na quadra (em amarelo). Entretanto, é facilitado o uso em tempo real, onde os polegares podem efetuar ações mais rapidamente em áreas próximas à borda do *tablet*. Os quadrantes são espelhados automaticamente de acordo com a equipe que está em posse da bola.
3. **Ações de Jogo:** Este espaço é destinado à marcação das ações de jogo. Ao selecionar um atleta, o botão é habilitado e podemos selecionar qual a técnica utilizada. Há dois módulos para a marcação, o simplificado mostrado na figura, e o módulo completo com as 11 técnicas possíveis do futevôlei.

4. **Logs de Rally:** Seção com o registro de todas as ações do *rally*. Assim que o *rally* termina, as ações ficam salvas e podem ser corrigidas com um toque em cima da ação desejada. Ao final do *rally* o motivo do ponto também é registrado.
5. **Área de Instruções:** Área com mensagens e instruções sobre a notação e o uso da aplicação. Informa os próximos passos e também as ações que estão sendo registradas.
6. **Inverter Lados:** Botão para inverter as duplas A e B de lado. Essencial para jogos em que as duplas trocam de lados para evitar o espelhamento manual do anotador.
7. **Voltar Ponto:** Botão para remover o último ponto da base de dados. Também subtrai o ponto do placar e as ações registradas. A interface atualiza para o último ponto registrado. Ao voltar o ponto, não serão consideradas as ações nas estatísticas do jogo.
8. **Microfone:** Botão para ativar ou desativar os comandos de voz. Por padrão, quando o *rally* inicia o microfone é ativado. A ação falada será detectada imediatamente após o toque no botão do jogador que a realizou. Só serão aceitas ações que já estão registradas no sistema, qualquer outra palavra será ignorada.
9. **Finalizar Partida:** Botão no canto inferior direito para encerrar o jogo. Mostra o histórico dos pontos com a dupla vencedora e o motivo de cada um deles antes de confirmar a finalização.

Após a finalização da partida, o usuário pode visualizar o resumo completo do jogo, conforme mostrado na Figura 12. Nele, estão disponíveis quatro seções em forma de abas para navegação: Resumo, Duplas, Individuais e Mapa de Ações. Cada seção é explicada a seguir.

Figura 12 – Resumo da Partida



Fonte: Elaborado pelo Autor (2025)

O resumo da partida compila os principais motivos de pontos e evidencia qual equipe foi vitoriosa. É um comparativo entre a pontuação de cada dupla em ataques, saques (aces), bloqueios e também os erros cometidos (ações que resultam em ponto para o adversário). Esta visão geral mostra, de forma visual, como foi distribuída a pontuação total e os fatores que levaram à vitória de uma equipe sobre a outra. Para isso, são utilizadas barras proporcionais ao número máximo de pontos em cada fundamento e cores para destacar a equipe mais efetiva em cada um deles.

Figura 13 – Estatísticas por Dupla



Fonte: Elaborado pelo Autor (2025)

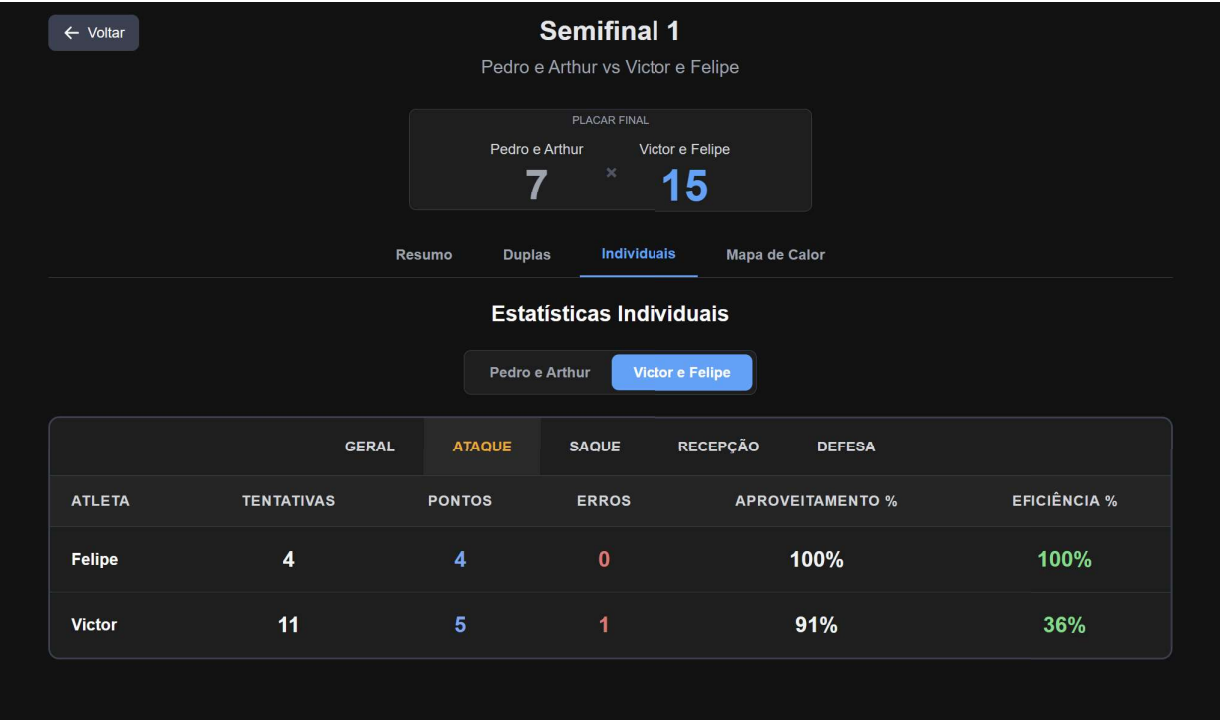
A aba de Duplas, Figura 13, apresenta as estatísticas agregadas por equipe, separada pelos fundamentos de ataque, saque, recepção e defesa. Além do valor absoluto de tentativas, pontos e erros, também são mostrados os seguintes indicadores de desempenho:

- **Aproveitamento:** Reflete a consistência da equipe, é a porcentagem de acertos, ou seja, ações que não resultaram em erro, sobre o total de tentativas.
- **Eficiência:** Indica o saldo de produtividade, obtido pela diferença entre pontos e erros, dividido pelo total de tentativas. Este indicador pode ser negativo, caso o número de erros supere o número de pontos feitos.

Os fundamentos de recepção e defesa não possuem o indicador de eficiência, por se tratar de ações que não resultam diretamente em pontos, mas sim na continuidade do *rally*. O importante nestes fundamentos é a redução de erros, já refletido no aproveitamento.

A aba de Individuais, Figura 14, apresenta as estatísticas detalhadas por atleta da dupla. As estatísticas mostradas são as mesmas da aba de Duplas, mas agora conseguimos comparar o desempenho individual de cada jogador da equipe.

Figura 14 – Estatísticas Individuais

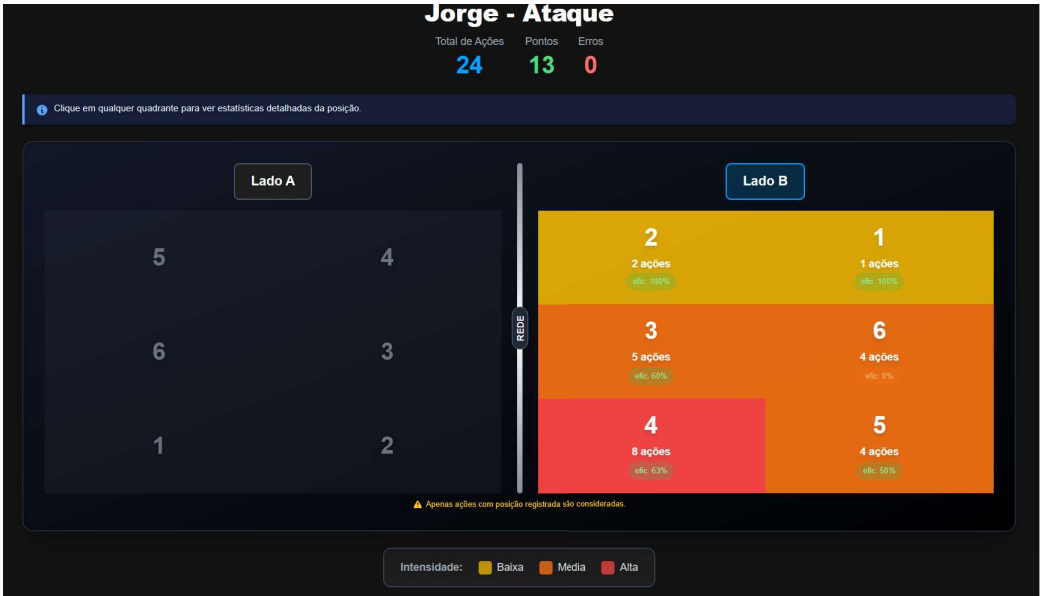


Fonte: Elaborado pelo Autor (2025)

A aba Mapa de Ações é assim chamada para facilitar a compreensão do usuário final, mas trata-se de um mapa coroplético, onde evidencia-se as ações realizadas em cada região da quadra. A quadra é dividida em lado A, à esquerda, e lado B, à direita, representando as duas equipes em jogo. Caso o usuário esteja observando uma partida específica, os lados serão correspondentes às equipes daquela partida. Ou seja, uma dupla estará sempre defendendo no lado A e atacando no lado B, enquanto a outra dupla estará na situação inversa. Caso o usuário esteja visualizando o histórico de um atleta ou dupla, os lados serão sempre normalizados para que o lado A seja o lado de defesa e o lado B o lado de ataque.

A Figura 15 demonstra a visualização do mapa coroplético com o atleta fictício Jorge no ataque. As áreas mais vermelhas indicam as regiões com maior concentração de ataques realizados, conforme a escala de intensidade abaixo do mapa. Em cada quadrante da quadra, são mostrados os números absolutos da ação realizada e a eficiência dela naquela região específica. Ao selecionar um quadrante, estas informações são detalhadas logo abaixo do mapa.

Figura 15 – Mapa de Ações - Ataque



Fonte: Elaborado pelo Autor (2025)

A Figura 16 também mostra mapa coroplético, porém, agora são as ações de recepção do atleta fictício Jorge. A interpretação é semelhante ao mapa de ataque, onde as áreas mais vermelhas indicam as regiões com maior concentração de recepções realizadas. Entretanto, como a recepção é um fundamento defensivo, o indicador mostrado em cada quadrante, além do valor absoluto de ações, é o aproveitamento da recepção na região específica.

Figura 16 – Mapa de Ações - Recepção



Fonte: Elaborado pelo Autor (2025)

Por fim, a última visualização mostrada do sistema é o histórico completo de partidas, que pode ser filtrado por atleta ou dupla. A figura 17 mostra o histórico de um atleta, onde estão listadas todas as dez últimas partidas do atleta, com informações como nome da partida, adversários, placar, resultado final e estatísticas resumidas sobre a pontuação do atleta na partida. É possível ir para a análise detalhada da partida através do botão em "Análise" à direita de cada linha.

Figura 17 – Histórico de Partidas de um Atleta



Fonte: Elaborado pelo Autor (2025)

4.2.5 Cronograma de Desenvolvimento

O cronograma de desenvolvimento é uma parte essencial para o planejamento de um projeto de software. Aqui define-se o tempo estimado e as entregas previstas para a construção do produto. Este desenvolvimento foi realizado em duas *releases* e uma entrega final.

Vale ressaltar que a metodologia SCRUM/XP, utilizada neste trabalho, permite que hajam alterações ao decorrer do desenvolvimento, então não é um processo rígido, mas sim um guia que foi usado para metas de progressão do projeto.

Tabela 7 – Cronograma Estimado de Desenvolvimento.

Sprint	Período	Objetivo Principal
1	18/08 - 01/09	Configuração e desenvolvimento da estrutura inicial do back-end.
2	02/09 - 16/09	Desenvolvimento da interface (<i>frontend</i>) inicial de gestão de partidas e conexão com o back-end.
3	17/09 - 01/10	Implementação da tela de notação em tempo real (interface e marcação de zona).
Release 1 (Primeiro MVP funcional para notação completa de um ponto)		
4	02/10 - 15/10	Desenvolvimento das estatísticas por equipe e por atleta.
5	16/10 - 30/10	Refinamento da usabilidade da notação, adicionando <i>feedback</i> ao anotador.
Release 2 (Ferramenta de notação robusta e com dashboard inicial)		
6	31/10 - 14/11	Testes de validação final com o estudo de caso, correção de bugs.
7	15/11 - 24/11	Implementação de funcionalidades desejáveis e Ajustes na documentação
Entrega Final (Versão completa do software e documentação)		

5 Considerações Finais

5.1 Considerações sobre os Objetivos

Conforme descrito na Seção 1.2.2, este trabalho foi desenvolvido com base em um objetivo geral e seis objetivos específicos. Esta seção é destinada à sumarização da conclusão de cada um dos objetivos definidos.

O **objetivo específico 1** foi alcançado por meio da revisão bibliográfica. A Tabela 1 e a Seção 2.1.3 apresentam as principais ações de jogo do futevôlei, que serviram de base para a definição das notações coletadas pela aplicação.

O **objetivo específico 2** foi atingido a partir das métricas definidas no GQM para o futevôlei, conforme a Seção 3.1.7.2, onde foi definido quais são os indicadores de desempenho para realizar as análises táticas e técnicas no sistema.

O **objetivo específico 3** foi concluído com a definição da arquitetura do sistema, apresentada na Seção 4.1.1 (Figura 7). A Seção detalha os componentes, suas responsabilidades e a justificativa da escolha tecnológica de cada um.

O **objetivo específico 4** foi alcançado pela implementação da visualização das estatísticas e dados geradas pela notação no sistema (Seção 4.2.4). O mapa coroplético contempla uma visualização gráfica das ações de jogo, auxiliando a análise tática e técnica da equipe e dos jogadores.

O **objetivo específico 5** foi realizado com o desenvolvimento da aplicação de software, detalhada ao decorrer do capítulo 4. O capítulo apresenta as informações técnicas, definições de projeto e as principais telas e funcionalidades do sistema. A solução é *open-source* e gratuita, disponibilizada no GitHub do autor ¹.

O **objetivo específico 6** foi contemplado pela aplicação da solução em cenários práticos. A validação foi realizada com a coleta de dados de partidas oficiais de futevôlei gravadas.

Sendo assim, o **objetivo geral** foi cumprido. Foi entregue uma ferramenta de software (OE5) que sistematiza as notações (OE1, OE6) e gera subsídios para a análise de desempenho (OE2, OE4) por meio de uma arquitetura viável (OE3). A disponibilização do sistema como *open-source* cumpre o propósito de auxiliar a análise de desempenho para todas as equipes e entusiastas do esporte. Portanto, a solução proposta atende ao objetivo geral e aos objetivos específicos definidos.

¹ Disponível em: <https://github.com/FelipeDireito/FTV-Scout>

5.2 Resultados

Neste trabalho, foi apresentado o desenvolvimento de uma solução de software para a notação e análise de desempenho no futevôlei que responde à questão principal de pesquisa proposta. A solução foi desenvolvida com base no referencial teórico da ciência do esporte, e uma arquitetura de software condizente com as necessidades do produto. Para a resolução do problema, foi definida uma diferença substancial para com os outros sistemas existentes: a necessidade de ter uma marcação em tempo real em um esporte ainda jovem e com baixo investimento financeiro e pouco pesquisado.

O uso de comandos de voz para notação das ações de jogo, proposto neste trabalho, não se mostrou efetivo durante os testes de campo. A velocidade da fala e a coordenação motora necessária para usar a voz e o toque são difíceis de serem conciliados. Entretanto, devido à automatização de fluxos no *software* e à simplicidade de uso de apenas um ou dois toques na tela para a marcação das ações de jogo durante o *rally*, o sistema torna-se rápido e eficiente para o uso em tempo real apenas utilizando o *touch*.

Com a solução implementada, é possível coletar os dados em tempo real e de forma gratuita, cumprindo com o propósito de incentivar a análise e coleta de dados no futevôlei. Quanto aos dados anotados, estes consistem, em sua totalidade, na contagem de números inteiros representando a frequência de eventos do jogo, definidos a partir de um instrumento de observação do futevôlei validado cientificamente. A base de dados permite a realização de análises estatísticas descritivas pertinentes — como média, desvio-padrão, mediana, máximos e mínimos — para comparação de atletas e duplas. Do ponto de vista inferencial, testes de hipóteses podem ser conduzidos para comparar as frequências obtidas em quaisquer dos casos mencionados, por exemplo, utilizando o teste Qui-quadrado de independência. Implementações futuras poderão contemplar esses avanços e favorecer o uso do sistema para fins de pesquisa.

5.3 Trabalhos Futuros

Há evoluções e trabalhos futuros objetivamente visíveis a partir do desenvolvimento realizado até aqui. O desenvolvimento de mais funcionalidades para a análise de desempenho, como gráficos e uma visualização de "ação por ação" pode ser implementado com base no modelo atual de dados. Esta visualização permitiria um detalhamento de cada ponto jogado, que pode ser útil para análises mais profundas.

A extração dos dados coletados em formato como *CSV* pode ser implementada também a partir da arquitetura atual. Essa funcionalidade permitiria que os dados fossem analisados em outras ferramentas, assim como poderia servir de insumo para analistas e pesquisadores externos ao sistema.

Uma evolução com custo adicional, seria a publicação do *software* em servidores na nuvem, eliminando a necessidade de configuração local do servidor pelo usuário. Isso facilitaria o uso para qualquer pessoa, sem a necessidade de conhecimentos técnicos prévios, entretanto, seria necessário criar um sistema de autenticação, e possivelmente uma criptografia de dados.

Por fim, uma evolução gananciosa possível é a adaptação do fluxo de notação manual atual para um fluxo automatizado, com o uso de inteligência artificial e técnicas de visão computacional. Seria necessário um modelo treinado para realizar o reconhecimento das ações de jogo, como saques, ataques, defesas, entre outras. Isso tornaria o processo de notação mais ágil e automatizado, diminuindo a carga cognitiva do anotador do sistema.

5.4 Fragilidades Externas da Aplicação

É possível identificar algumas limitações e dependências que são visíveis no produto:

1. **Complexidade da Configuração Inicial:** A configuração local do servidor, embora funcione bem para eliminar custos, faz-se necessário que o usuário tenha um conhecimento básico sobre o uso do software e sua configuração inicial. Este problema pode ser mitigado com a publicação do sistema em servidores na nuvem, a arquitetura já está preparada para isso.
2. **Coordenação Motora:** Mesmo com o sistema otimizado e fluído, a necessidade de coordenação motora para o uso do sistema em tempo real pode ser um desafio para alguns usuários. Uma das alterações implementadas foi a possibilidade de simplificar os tipos de técnica da ação, que pode ser reduzido de 11 para apenas 5 técnicas, diminuindo o erro na marcação. Recomenda-se o uso completo das técnicas apenas em notações pós-jogo.

5.5 Conclusões

A solução de *software* desenvolvida e apresentada neste trabalho preenche a lacuna identificada na coleta de dados no futevôlei ao estabelecer uma padronização e estruturação das informações para todas as partidas. O valor científico é prático: com dados organizados, pode-se realizar análises e, a partir de uma base robusta com quantidade significativa de notações, pode-se determinar um "padrão ouro" de desempenho. Isso pode ser utilizado para identificar os atletas com melhores índices em cada técnica e ação de jogo, e evidenciar os atletas de elite do esporte. Isso também gera uma base comparativa para os atletas que estão procurando evoluir seu desempenho.

Além do valor agregado pelo trabalho, o projeto amplia as expectativas para melhorias futuras. Por ser uma ferramenta de código aberto, ele pode ser modificado por desenvolvedores e pela comunidade acadêmica. O objetivo final é que o projeto estimule o aprofundamento de pesquisas notacionais e novos estudos, para dar continuidade ao incentivo à coleta de dados e o uso de métricas dentro do futevôlei.

Referências

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. *The Goal Question Metric Approach*. College Park, MD, USA, 1994. v. 1, 528–532 p. Citado 2 vezes nas páginas 30 e 42.

BORGES, L. C.; SENATORE, J. V.; LAMAS, L. Tactical-technical performance in footvolley: validation of an observational instrument. v. 30, p. e10240082–e10240082, 2024. ISSN 1980-6574. Disponível em: <<https://www.periodicos.rc.biblioteca.unesp.br/index.php/motriz/article/view/18357>>. Citado 4 vezes nas páginas 13, 15, 18 e 19.

Conventional Commits. *Conventional Commits Specification*. 2025. Acesso em: 13 nov. 2025. Disponível em: <<https://www.conventionalcommits.org/en/v1.0.0/>>. Citado na página 39.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados*. 6. ed. São Paulo: Addison-Wesley, 2011. ISBN 978-85-7936-160-2. Citado na página 27.

FOWLER, M. *UML Essencial: Um Breve Guia para a Linguagem Padrão de Modelagem de Objetos*. 3ª ed.. ed. Porto Alegre: Bookman, 2005. Citado na página 32.

GABIN, B. et al. Lince: Multiplatform sport analysis software. v. 46, p. 4692–4694, 2012. ISSN 18770428. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S1877042812020563>>. Citado na página 21.

GALIN, D. *Software quality assurance*. [S.l.]: Pearson Education Limited, 2004. ISBN 978-0-201-70945-2. Citado na página 28.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa*. Porto Alegre: Editora da UFRGS, 2009. ISBN 9788575165219. Citado 2 vezes nas páginas 36 e 37.

GIATISIS, G. Performance indicators in women's volleyball olympics and world championships (2014–2021). v. 18, n. 4, p. 1266–1276, 2023. ISSN 1747-9541. Publisher: SAGE Publications. Disponível em: <<https://doi.org/10.1177/17479541221106378>>. Citado na página 20.

GitHub. *GitHub Flow*. 2025. Acesso em: 13 nov. 2025. Disponível em: <<https://docs.github.com/en/get-started/using-github/github-flow>>. Citado na página 39.

GÓMEZ-RUANO, M. Current approaches to performance analysis in sport. v. 14, n. 51, p. 3–4, 2018. Citado na página 13.

International Organization for Standardization. *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE*. Geneva, CH, 2014. Citado na página 28.

LIMA, V. d. S. et al. Status of match analysis research in indoor and beach volleyball: A bibliometric analysis. v. 50, p. 838–848, 2023. ISSN 1988-2041. Disponível em: <<https://recyt.fecyt.es/index.php/retos/article/view/100039>>. Citado na página 13.

MANIFESTO for Agile Software Development. 2001. Acesso em: 23 jun. 2025. Disponível em: <<https://agilemanifesto.org/>>. Citado na página 23.

MORGULEV, E.; AZAR, O. H.; LIDOR, R. Sports analytics and the big-data era. v. 5, n. 4, p. 213–222, 2018. ISSN 2364-4168. Disponível em: <<https://doi.org/10.1007/s41060-017-0093-7>>. Citado 3 vezes nas páginas 17, 18 e 20.

RAMOS, A. et al. How competitive performance data can inform the training process? an action-research study based on the constraint-led approach. v. 55, n. 2, p. 337–348, 2023. ISSN 1848638X, 13311441. Disponível em: <<https://hrcak.srce.hr/311307>>. Citado 2 vezes nas páginas 15 e 17.

RANGEL, W. et al. Integrated evaluation of team strategy, training practices and game performance of a basketball team. v. 18, n. 1, p. 197–206, 2023. ISSN 1747-9541, 2048-397X. Disponível em: <<https://journals.sagepub.com/doi/10.1177/17479541221076621>>. Citado na página 13.

Scrum.org. *Scrum Framework Poster*. 2025. Acesso em: 9 jul. 2025. Disponível em: <<https://www.scrum.org/resources/scrum-framework-poster>>. Citado na página 24.

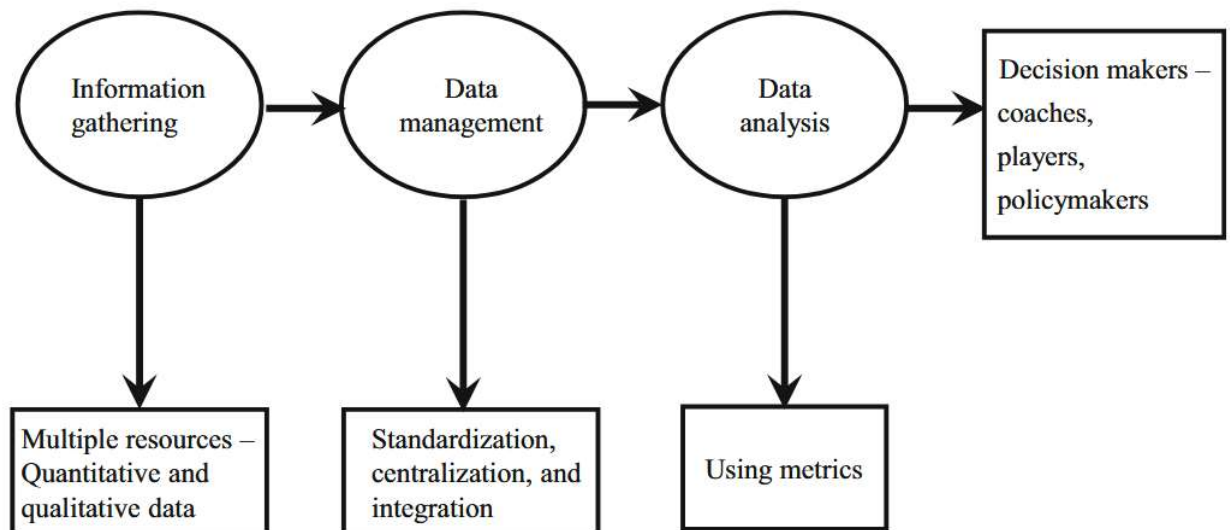
SILVA, M.; LACERDA, D.; JOÃO, P. V. Match analysis of discrimination skills according to the setter defence zone position in high level volleyball. *International Journal of Performance Analysis in Sport*, v. 14, n. 2, p. 463–472, 2014. Disponível em: <<https://doi.org/10.1080/24748668.2014.11868735>>. Citado na página 20.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 978-0-321-31379-9, 0-321-31379-8. Citado 7 vezes nas páginas 26, 29, 31, 33, 34, 39 e 40.

Anexos

ANEXO A – Figura Original

Figura A.1 – Sports Analysis Framework



Fonte: Morgulev, Azar e Lidor (2018)

ANEXO B – Tabela Original

Tabela B.1 – Categorias e variáveis

Game actions		
Serve	— Player involved — Serve technique — Execution zone	— Destination zone — Ball trajectory* — Serve result
Reception/defense	— Player involved — Reception/Defense technique	— Execution zone — Destination zone — Reception/Defense result
Set	— Player involved — Set technique	— Execution zone — Destination zone — Set result
1st touch correction	— Player involved — 1st touch technique	— Execution zone — Destination zone — 1st touch result
Attack	— Player involved — Attack technique — Execution zone	— Destination zone — Attack result
Block	— Player involved — Block technique	— Execution zone — Destination zone — Block result

Fonte: Borges, Senatore e Lamas (2024)